Boosting Offline MARL under Imbalanced Datasets via Compositional Diffusion Models

Anonymous Author(s)
Submission Id: 574

ABSTRACT

Offline multi-agent reinforcement learning (MARL) is hampered by agent-quality imbalance in datasets, where the entanglement of expert and suboptimal behaviors from heterogeneous behavior policies inhibits effective policy learning. Conventional offline MARL methods overfit to these suboptimal behaviors, leading to significant performance degradation. A promising solution is data augmentation using generative models like diffusion model, which can generate balanced, high-quality trajectories to enrich the dataset. However, existing methods usually adopt a standard diffusion process, conditioning generation solely on team-level signals such as global return. This coarse guidance lacks active, fine-grained, agent-level control, limiting the diffusion model's ability to produce high-quality cooperative behaviors that generalize beyond the dataset. To address this, we propose **Co**mpositional **D**iffusion for **I**mbalanced Datasets (CODI), a novel framework that leverages large language models (LLMs) and diffusion models to generate balanced, high-quality trajectories. CODI first distills an agent quality labeler from an LLM to annotate the dataset. It then employs a conditional diffusion model that generates trajectory segments based on not only return-to-go but also fine-grained agent quality labels. Crucially, to effective compose scattered high-quality behaviors and enable generalization, CODI decomposes the target team quality into in-distribution agent-level labels for compositional diffusion generation. These generated segments are subsequently stitched into complete trajectories, augmenting the dataset. Extensive evaluation on challenging imbalanced datasets, where only a single agent is an expert, shows that CODI successfully mitigates data imbalance and facilitates the learning of strong cooperative policies, recovering 63% of the performance achieved with a balanced expert dataset and substantially outperforming baseline methods.

KEYWORDS

Multi-agent systems, Reinforcement learning, Diffusion models

ACM Reference Format:
Anonymous Author(s). 2026. Boosting Offline MARL under Imbalanced Datasets via Compositional Diffusion Models. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages.

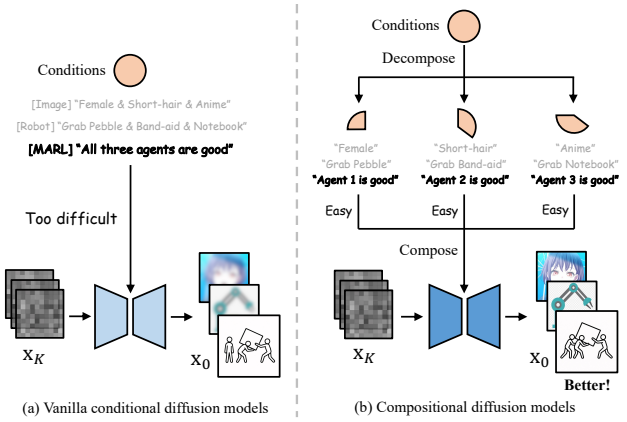
1 INTRODUCTION

Recently, cooperative multi-agent reinforcement learning (MARL) has emerged as a powerful paradigm for solving complex tasks involving multiple interacting agents [24], with applications spanning autonomous driving [42], large language models (LLMs) [7], and embodied intelligence [5]. While online MARL algorithms have shown impressive results in simulated environments [20, 28, 34, 38], their reliance on extensive environment interaction poses significant challenges in real-world deployment where trial-and-error learning is prohibitively expensive, risky, or ethically constrained. Offline MARL [6] addresses this limitation by learning policies directly from static datasets, eliminating the need for environment interaction during training. This paradigm has gained considerable attention recently, with promising methods like OMAR [25] and OMIGA [36], and applications such as domain calibration [13].

However, the performance of offline MARL is inherently bounded by the quality of the learned datasets. While these methods achieve strong performance when trained on all-expert data, their performance degrades significantly when applied to suboptimal datasets, particularly those exhibiting quality imbalance across agents [39]. In real-world multi-agent scenarios, coordination data is often collected by agents with heterogeneous capabilities, such as a football game with human players of varying levels. This results in trajectories where expert-level behaviors from some agents are entangled with the suboptimal actions of others. Such agent-quality imbalance poses a fundamental challenge for offline MARL, as algorithms may fail to leverage the scattered high-quality individual behaviors and instead overfit to the weak coordination, eventually learning suboptimal cooperative policies. One promising solution is data augmentation, generating high-quality and balanced coordination data to enrich the datasets before learning. Single-agent-based methods, BATS [1] and MBTS [9], stitch existing trajectories but cannot synthesize novel coordination. Utilizing generative models like diffusion models [37], MADiff [46] and MADiTS [39] set high target returns as the only instruction to steer the generation of better coordination. Nonetheless, this coarse guidance lacks active, fine-grained, agent-level control, limiting the diffusion model's ability to learn about the inherent coordination patterns and produce high-quality cooperative behaviors that generalize beyond the dataset.

To address this challenge, we propose **Co**mpositional **D**iffusion for **I**mbalanced Datasets (CODI), a novel framework that leverages LLMs and diffusion models to generate high-quality, balanced trajectories. CODI first distills an LLM's coordination knowledge into an agent quality labeler, which efficiently annotates the dataset with fine-grained quality signals. A conditional diffusion model is then trained to generate trajectory segments based on both return-to-go (RTG) and these quality labels, ensuring the output is not only high-return but also balanced across agents. However, a critical out-of-distribution (OOD) issue arises: since the original dataset is dominated by severely imbalanced trajectories, conditioning the model on the desired "all-expert" quality label presents a situation rarely seen during training. This challenge of generating samples under OOD conditions is a fundamental problem for conditional

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.



(a) Vanilla conditional diffusion models (b) Compositional diffusion models

Figure 1: A comparison of vanilla and compositional diffusion models. Compositional diffusion produces higher-quality samples, a capability we leverage for offline MARL.

diffusion models, commonly observed in domains like image generation [19], robotics [35, 45], and multi-agent world models [41]. To address this, compositional diffusion techniques have been developed, which excel at generating novel, high-quality outputs by combining in-distribution concepts (Figure 1). Inspired by this, CODI novelly decomposes the OOD target of team quality into a set of in-distribution, agent-level quality labels. This decomposition enables effective compositional generation, composing scattered high-quality behaviors of individual agents from the dataset to form novel and balanced cooperation that generalize beyond the original data. The generated high-quality segments are then stitched into complete, high-quality cooperative trajectories for policy learning.

We conduct experiments on challenging imbalanced datasets where only a single agent is an expert across four MARL tasks, including Cooperative Navigation (CN) and World from the Multi-Agent Particle Environment (MPE) [20], and two combat scenarios in StarCraft Multi-Agent Challenge (SMAC) [29] and the more challenging SMACv2 [4]. CODI successfully generates trajectories with high fidelity, quality, and agent imbalance, enabling the learning of strong cooperative policies. Our method recovers an average of 63% of the performance gap between the imbalanced dataset and the balanced expert policy, substantially outperforming baseline methods. These results highlight the capability of CODI to boost offline MARL under imbalanced datasets.

2 BACKGROUND

2.1 Offline MARL

Fully cooperative multi-agent decision-making problems can be formalized by a Dec-POMDP [23], defined as

$$\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, \Omega, O, R, \rho, \gamma \rangle.$$

In this formulation, $\mathcal{N} = \{1, \cdots, n\}$ denotes the agent set containing n agents, $\mathcal{S}$ is the global state space, $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^n$ indicates the joint action space, where $\mathcal{A}^i$ is the action space available to agent i. P is the state transition dynamics, Ω is the observation space, $O : \mathcal{S} \times \mathcal{A} \to \Omega$ is the observation function, R is the reward function, ρ is the initial state distribution, and γ is the discount factor for future rewards. At each time step t, every agent i receives a local observation $o_t^i \in \Omega$ and selects an action $a_t^i \in \mathcal{A}^i$, forming the joint action $\boldsymbol{a}_t$. The environment then transitions from s_t to the next state s_{t+1} according to transition function $P(s_{t+1}|s_t, \boldsymbol{a})$, while providing a collective reward $r_t = R(s_t, \boldsymbol{a}, s_{t+1})$. The optimization target for agent policies $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_n)$ is formulated as $\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}) = \mathbb{E}_{s_0 \sim \rho, \boldsymbol{a}_t \sim \boldsymbol{\pi}(\cdot|o_t^{1:n}), s_{t+1} \sim P(\cdot|s_t, \boldsymbol{a}_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$. In the offline MARL scenario, agents are restricted from environment interaction and must instead learn from a fixed dataset $\mathcal{D} = \{\tau\}$ with trajectories $\tau = (\boldsymbol{o}_0, \boldsymbol{a}_0, r_0, \ldots, \boldsymbol{o}_T, \boldsymbol{a}_T, r_T)$ collected by unknown behavioral policies $\boldsymbol{\pi}_\beta$. In this work, we focus on learning from severely *imbalanced* datasets collected by $\boldsymbol{\pi}_\beta$ comprising a single expert agent alongside multiple sub-optimal (e.g., random) agents.

2.2 Denoising Diffusion Probabilistic Models

Given offline learning tasks like offline MARL, Denoising Diffusion Probabilistic Models (DDPMs) [10, 31, 32] serve as a class of generative models that can learn the underlying data distribution $p(\mathbf{x})$ by reversing a predefined forward noising process. The core idea is to first gradually corrupt a data sample with noise until it becomes indistinguishable from pure Gaussian noise, and then to train a neural network to learn the reverse process, thereby allowing the generation of new samples from noise. The forward process adds Gaussian noise to a data sample $\mathbf{x}_0$, producing noisy latents $\mathbf{x}_1, \ldots, \mathbf{x}_K$. The reverse process aims to recover the data structure from noise. It starts by sampling $\mathbf{x}_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively denoises it to produce a sequence $\mathbf{x}_{K-1}, \ldots, \mathbf{x}_0$. Since the true reverse distribution $q(\mathbf{x}_{k-1}|\mathbf{x}_k)$ is intractable, a neural network parameterized by θ is trained to approximate it. In the DDPM formulation, this network $\epsilon_\theta(\mathbf{x}_k, k)$ is tasked with predicting the noise component ϵ that was added to $\mathbf{x}_0$ to obtain $\mathbf{x}_k$. The training objective is a simplified mean-squared error loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{k, \mathbf{x}_0, \epsilon} \left[\|\epsilon - \epsilon_\theta(\mathbf{x}_k, k)\|^2 \right], \tag{1}$$

where k is uniformly sampled from $\{1, \ldots, K\}$, $\mathbf{x}_0$ is a training sample, and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Once the model is trained, new samples are generated through an iterative sampling procedure from $k = K$ down to $k = 1$. A common sampling step, derived from the reverse process, computes $\mathbf{x}_{k-1}$ by subtracting the predicted noise and then injecting new stochastic noise with a variance σ_k^2:

$$\mathbf{x}_{k-1} = \mathbf{x}_k - \epsilon_\theta(\mathbf{x}_k, k) + \mathcal{N}(\mathbf{0}, \sigma_k^2 \mathbf{I}). \tag{2}$$

For conditional generation of the form $p(\mathbf{x}|\mathbf{y})$, where $\mathbf{y}$ is an auxiliary input such as a class label or a text prompt, the model is adapted to become $\epsilon_\theta(\mathbf{x}_k, k|\mathbf{y})$. An effective technique for achieving high-quality conditional generation is classifier-free guidance [32]. During sampling, the noise prediction is adjusted as follows:

$$\hat{\epsilon} = \epsilon_\theta(\mathbf{x}_k, k) + w \left(\epsilon_\theta(\mathbf{x}_k, k|\mathbf{y}) - \epsilon_\theta(\mathbf{x}_k, k) \right), \tag{3}$$

where w is the guidance scale. This guided estimate $\hat{\epsilon}$ effectively pushes the sampling process towards regions of the data space where the condition $\mathbf{y}$ is satisfied. The guided estimate $\hat{\epsilon}$ is then used in place of $\epsilon_\theta(\mathbf{x}_k, k)$ in the sampling update (Eq. (2)).

Owing to their functional similarity to energy-based models (EBMs) [3], diffusion models offer a framework for compositional

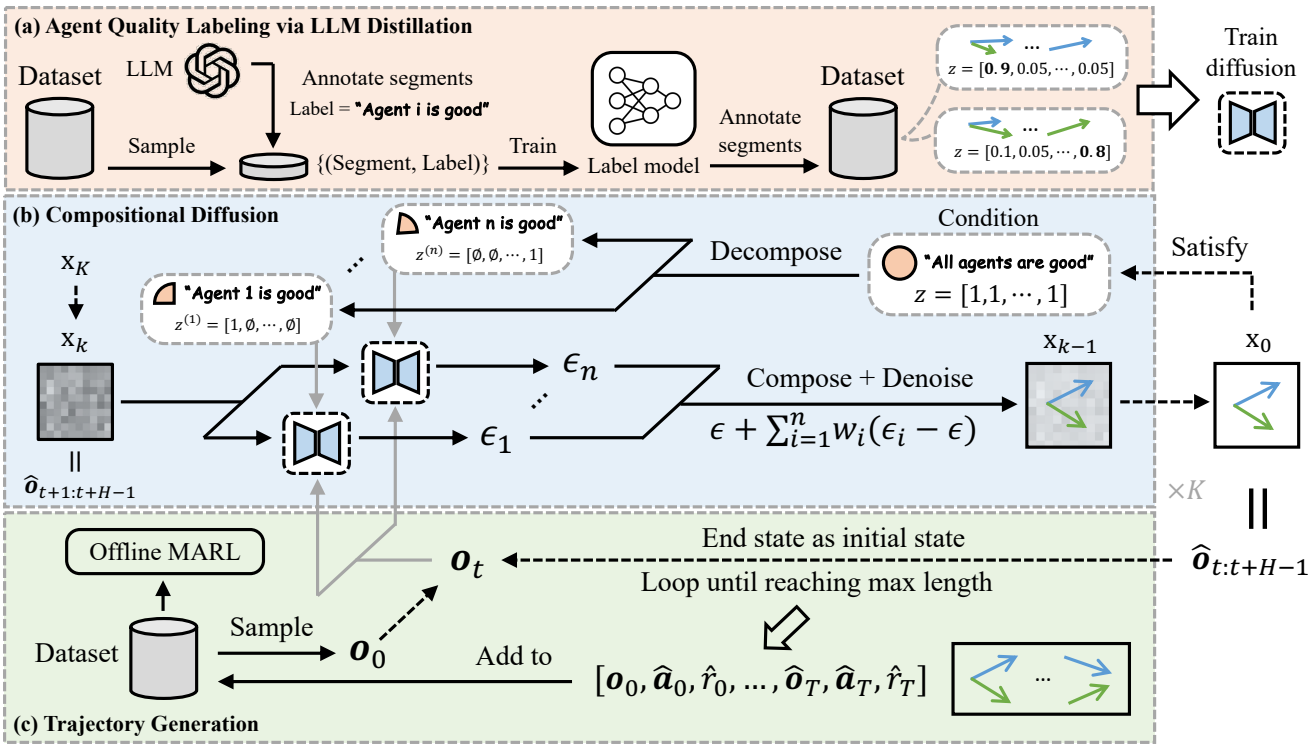


Figure 2: The overall workflow of CODI. (a) Agent quality labeling via LLM distillation. An LLM is distilled into a label model to annotate agent quality for diffusion training. (b) Compositional diffusion. A compositional diffusion model is conditioned on quality labels to generate balanced, high-quality trajectory segments. (c) Trajectory generation. The generated segments are stitched into complete cooperative trajectories.

generation. This allows for the generation of samples that simultaneously satisfy multiple concepts or conditions $\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(n)}$, i.e., modeling $p(\mathbf{x}|\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(n)})$. A straightforward yet powerful compositional rule is to form a linear combination of the guidance signals from each individual condition:

$$\hat{\epsilon}_{\text{comp}} = \epsilon_\theta(\mathbf{x}_k, k) + \sum_{i=1}^{n} w_i \left[\epsilon_\theta\left(\mathbf{x}_k, k|\mathbf{y}^{(i)}\right) - \epsilon_\theta(\mathbf{x}_k, k)\right]. \quad (4)$$

This compositional approach, where each condition $\mathbf{y}^{(i)}$ is associated with its own guidance weight w_i, has been empirically shown in recent research [19] to significantly outperform the naive baseline of simply concatenating all conditions into a single input $\mathbf{y} = (\mathbf{y}^{(1)}, \cdots, \mathbf{y}^{(n)})$. It provides finer control over the influence of each concept during the generation process.

3 METHODS

This section introduces CODI (Figure 2), a novel data augmentation framework that generates high-quality, balanced cooperative trajectories from imbalanced offline datasets. The framework operates through three stages: First, it trains an agent quality labeler via LLM knowledge distillation, producing fine-grained quality annotations across the dataset (Section 3.1). These quality labels then serve as conditions for training a compositional diffusion model, enabling it to better capture coordination patterns and generate trajectory segments with high and balanced agent qualities (Section 3.2). Finally, generated segments are stitched into complete trajectories for data augmentation (Section 3.3).

3.1 Agent Quality Labeling via LLM Distillation

To effectively learn from datasets with imbalanced agent qualities, CODI first establishes a robust quality assessment mechanism that accurately evaluates agent performance within trajectories.

Specifically, the framework begins by leveraging a LLM[1] as an expert annotator to identify high-performing agents within trajectory segments. Consider an offline dataset $\mathcal{D} = \{\tau_j\}_{j=1}^m$ with m trajectory segments, where each segment $\tau_j = (\boldsymbol{o}_t, \boldsymbol{a}_t, r_t, \ldots, \boldsymbol{o}_{t+H-1}, \boldsymbol{a}_{t+H-1}, r_{t+H-1})$ consists of H consecutive steps of joint observations, joint actions, and rewards. For each segment τ_j, CODI converts it into a natural language description $\tau_j^{\texttt{text}}$ via a simple handcrafted function (e.g., "the position of agent 1 is {the first two dimensions of o_t^1}"). This textual feature is then incorporated into a structured prompt $[\texttt{desc}, \tau_j^{\texttt{text}}, \texttt{inst}]$, where `desc` provides basic information of environment, agents, and task, and `inst` is a concise instruction such as "*select the best agent*". Utilizing its rich knowledge and reasoning ability, the LLM processes this prompt to output

[1]We use the `gpt-4o-mini` model in our work

detailed quality assessments for all agents, more interpretable than prior black-box methods. Finally, it annotates index $y_j \in \{1, \cdots, n\}$ indicating the best-performing agent in the segment.

However, given the substantial volume of trajectory segments in practical applications, directly employing the LLM for comprehensive annotation would be prohibitively expensive. To address this challenge, CODI employs a knowledge distillation strategy: first, a randomly selected subset of trajectory segments (less than 10% of the dataset in our experiments) is annotated using the LLM as an expert oracle. These high-quality annotations $\mathcal{D}_{\text{label}} = \{(\tau_j, y_j)\}$ then serve as training data for a compact quality labeling model f_{label} that learns to replicate the LLM's assessment capabilities.

Training f_{label} constitutes a supervised classification task. The network architecture employs a Gated Recurrent Unit (GRU) [2] to capture temporal dependencies within trajectory segments, followed by a multi-layer perceptron (MLP) that produces final classification logits: $\boldsymbol{z}_j = (z_j^1, \cdots, z_j^n) = f_{\text{label}}(\tau_j)$, where $z_j^i \in [0, 1]$ represents the predicted probability that agent i is the best-performing agent in trajectory segment τ_j, and $\sum_{i=1}^n z_j^i = 1$. The model is optimized using the standard cross-entropy loss:

$$\mathcal{L}_{\text{label}} = \mathbb{E}_{\tau_j \sim \mathcal{D}_{\text{label}}} \left[-\sum_{i=1}^{n} \mathbb{I}\{y_j = i\} \log z_j^i \right], \tag{5}$$

where $\mathbb{I}$ is the indicator function. It is worth noting that the formulation of $\boldsymbol{z}_j$ and the loss function can be flexibly adapted to accommodate different data characteristics, such as multiple high-performing agents. In this work, we adopt the standard classification setup with a single best agent for simplicity. Once trained, this distilled model efficiently annotates the entire dataset, providing reliable quality labels for subsequent training stages while dramatically reducing computational costs. More details about agent quality labeling are provided in Appendix A.1 and Appendix E.

3.2 Compositional Diffusion for Offline MARL

To actively guide the generation of cooperative trajectories towards both high returns and balanced performance, we train a diffusion model conditioned on not only return-to-go (RTG) values, but also agent quality labels obtained through LLM distillation.

Formally, consider trajectory segments $\tau = (\boldsymbol{o}, \boldsymbol{a}, r)_{t:t+H-1}$, we define diffusion samples as observation sequences $\mathbf{x} = \boldsymbol{o}_{t:t+H-1}$, and conditional generation as $p_\theta(\mathbf{x}|\hat{R}, \boldsymbol{z})$, where $\hat{R} = \sum_{t'=t}^{T} r_{t'}$ is the RTG value from the segment's starting point to trajectory termination, and $\boldsymbol{z} = f_{\text{label}}(\tau)$ is the predicted quality label vector. The training objective for our conditional diffusion model adapts the standard DDPM loss from Eq. (1):

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{k, \mathbf{x}_0, \epsilon} \left[\|\epsilon - \epsilon_\theta(\mathbf{x}_k, k|\hat{R}, \boldsymbol{z})\|^2 \right]. \tag{6}$$

During training, we apply dropout regularization to the label dimensions to enhance model robustness. Specifically, for each training sample, we randomly mask out individual dimensions of the label vector $\boldsymbol{z}$ with probability $p_{\text{drop}} = 0.25$, forcing the model to learn robust conditional generation that can handle incomplete or partial label information, which is important for the subsequent compositional generation phase.

After training, the intended generation step is to sample from $p_\theta(\mathbf{x}|\hat{R}_{\text{gen}}, \boldsymbol{z}_{\text{gen}})$, where $\hat{R}_{\text{gen}}$ is a high RTG value, and $\boldsymbol{z}_{\text{gen}} = \mathbf{1}_n = [1, \cdots, 1]$ is the target condition for balanced, high-quality performance across all n agents. However, a direct application fails because the condition $\boldsymbol{z}_{\text{gen}}$ is severely out-of-distribution (OOD). The original imbalanced dataset could be dominated by trajectories where only a small subset of agents are experts (e.g., $\boldsymbol{z} = [1, 0, \cdots, 0]$), making the "all-expert" condition $\mathbf{1}_n$ virtually unseen during training. To mitigate this problem, we innovatively leverage the compositional diffusion technique [19] by decomposing the joint condition into a set of simpler, in-distribution conditions:

$$p_\theta(\mathbf{x}|\hat{R}_{\text{gen}}, \boldsymbol{z}_{\text{gen}}) = p_\theta(\mathbf{x}|\hat{R}_{\text{gen}}, \boldsymbol{z}_{\text{gen}}^{(1)}, \cdots, \boldsymbol{z}_{\text{gen}}^{(n)}), \tag{7}$$

where $\boldsymbol{z}_{\text{gen}}^{(i)}$ is a vector with $z^i = 1$ and all other dimensions $z^{j \neq i}$ masked, representing the concept "agent i performs well", a condition that is abundantly present in the imbalanced dataset (e.g., segments where at least one agent is good). Also, label dropout during training makes the model familiar with partial label information. This approach allows us to compose the scattered, high-quality behaviors of individual agents from the dataset, yielding novel team-level cooperative behaviors. During generation, we compose these agent-level individual concepts following Eq. (4):

$$\hat{\epsilon}_{\text{comp}}^{\text{CODI}} = \epsilon_\theta(\mathbf{x}_k, k) + \sum_{i=1}^{n} w_i \left[\epsilon_\theta\left(\mathbf{x}_k, k|\hat{R}_{\text{gen}}, \boldsymbol{z}_{\text{gen}}^{(i)}\right) - \epsilon_\theta(\mathbf{x}_k, k) \right], \tag{8}$$

maximizing the probability of $\mathbf{x}_{k-1}$ satisfying all concepts from the perspective of energy-based models (EBMs) [3]. In this work, we set equal weights w_i for balanced composition, while the framework allows adaptive adjustment to emphasize specific agents when needed. This approach effectively steers the generation towards the target concept $\boldsymbol{z}_{\text{gen}} = \mathbf{1}_n$ ("All agents perform well") by leveraging only in-distribution, single-agent-expert conditions during the denoising process. The final generated segments with high return and balanced quality, are subsequently stitched into complete trajectories to augment the offline dataset for policy learning.

3.3 Pipeline for Trajectory Generation

Finally, CODI employs the trained compositional diffusion model to generate complete trajectories, following a stitching pipeline analogous to [39]. During training, we optimize the diffusion model, along with three auxiliary models $f_{\text{inv}}, f_{\text{fwd}}, f_{\text{rwd}}$ that learn the inverse dynamics, transitions, and rewards:

$$\begin{aligned} \mathcal{L}_{\text{total}} = \mathcal{L}_{\text{diff}}(\theta) + \mathbb{E}_{\mathcal{D}}[\|\boldsymbol{a}_t - f_{\text{inv}}(\boldsymbol{o}_t, \boldsymbol{o}_{t+1})\|^2 \\ + \|\boldsymbol{o}_{t+1} - f_{\text{fwd}}(\boldsymbol{o}_t, \boldsymbol{a}_t)\|^2 + \|r_t - f_{\text{rwd}}(\boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1})\|^2]. \end{aligned} \tag{9}$$

To generate a trajectory, we first sample an initial joint observation $\boldsymbol{o}_0$ from $\mathcal{D}$, then initialize a noisy sequence $\mathbf{x}_K = [\boldsymbol{o}_0, \tilde{\boldsymbol{x}}_1, \cdots, \tilde{\boldsymbol{x}}_{H-1}]$ where future steps are Gaussian noise. This sequence is denoised over K steps following Eq. (2):

$$\mathbf{x}_{k-1} = \mathbf{x}_k - \hat{\epsilon}_{\text{comp}}^{\text{CODI}} + \mathcal{N}(\mathbf{0}, \sigma_k^2 \mathbf{I}), \quad \text{for } k = K, \cdots, 1, \tag{10}$$

where $\hat{\epsilon}_{\text{comp}}^{\text{CODI}}$ is the compositional denoising output defined in Eq. (8). The initial observation $\boldsymbol{o}_0$ remains fixed as conditioning, yielding $\mathbf{x}_0 = [\boldsymbol{o}_0, \hat{\boldsymbol{o}}_1, \cdots, \hat{\boldsymbol{o}}_{H-1}]$. Each generated segment is validated for dynamic consistency: for every consecutive pair $(\hat{\boldsymbol{o}}_t, \hat{\boldsymbol{o}}_{t+1})$, we check if $|\hat{\boldsymbol{o}}_{t+1} - f_{\text{fwd}}(\hat{\boldsymbol{o}}_t, f_{\text{inv}}(\hat{\boldsymbol{o}}_t, \hat{\boldsymbol{o}}_{t+1}))| \leq \delta$. If valid, actions and rewards are imputed as $\hat{\boldsymbol{a}}_t = f_{\text{inv}}(\hat{\boldsymbol{o}}_t, \hat{\boldsymbol{o}}_{t+1})$ and $\hat{r}_t = f_{\text{rwd}}(\hat{\boldsymbol{o}}_t, \hat{\boldsymbol{a}}_t, \hat{\boldsymbol{o}}_{t+1})$, and the RTG $\hat{R}$ is updated. The last observation $\hat{\boldsymbol{o}}_{H-1}$ then initializes

(a) MPE: CN (b) MPE: World (c) SMAC: 3m (d) SMACv2: Zerg_3v4

Figure 3: Cooperative MARL tasks used in this paper. (a) Cooperative Navigation (CN) in Multi-Agent Particle Environment (MPE) [20]. (b) World in MPE. (c) 3m in StarCraft Multi-Agent Challenge (SMAC) [29]. (d) Zerg_3v4 in SMACv2 [4].

the next segment. If the consistency check fails, we employ an iterative resampling strategy: the noise input to the diffusion model is resampled to regenerate the inconsistent segment. This resampling process repeats until either a dynamically consistent segment is obtained or a maximum resampling threshold is reached. Upon exceeding the threshold, the current trajectory segment is abandoned, and we reinitialize the process by sampling a new initial observation $\boldsymbol{o}_0$ from the dataset to start a fresh generation attempt. This generation–stitching process repeats until trajectories reach a target length, and can be parallelized for efficiency. The generated trajectories are added to $\mathcal{D}$, enriching data quality and coordination balance to boost downstream offline MARL. Pseudocode and hyperparameters of the pipeline are provided in Appendix A.2.

4 EXPERIMENTS

We conduct extensive experiments to evaluate the effectiveness of CODI, addressing the following key questions: (1) Can CODI significantly improve the performance of offline MARL algorithms when learning from severely imbalanced datasets (Section 4.2) ? (2) How does CODI operate in detail, and to what extent does compositional diffusion contribute (Section 4.3) ? (3) How does CODI scale as the number of agents increases (Section 4.4) ?

4.1 Experimental Setup

Cooperative Multi-Agent Environments. We conduct experiments on four classic cooperative multi-agent tasks (Figure 3). The first two are from the Multi-Agent Particle Environment (MPE) [20]: **Cooperative Navigation (CN)**, where three agents cover landmarks while avoiding collisions, and **World**, a complex predator-prey scenario requiring high-level coordination among three agents. The other two are combat scenarios from StarCraft: **3m** from the original StarCraft Multi-Agent Challenge (SMAC) [29], where three allied marines battle three enemy marines, and **Zerg_3v4** from the more challenging SMACv2 [4], which presents a more challenging asymmetric 3v4 battle with randomized start positions, demanding robust strategies. Further details are in Appendix D.

Imbalanced Datasets Collection. To investigate the impact of severe agent-quality imbalance, we construct datasets where only one agent in each trajectory exhibits expert-level performance. For each environment, we begin by training a proficient, balanced joint behavior policy using QMIX [28]. To create the imbalanced datasets, we then collect 20k trajectories per environment under the following protocol: in each episode, exactly one randomly selected agent executes the expert policy, while all other agents are constrained to take random actions. This setup effectively simulates realistic scenarios of sub-optimal coordination commonly encountered during open-environment data collection. To assess the effectiveness of data augmentation methods, we combine the original 20k imbalanced trajectories with an additional 20k trajectories generated by each augmentation method, forming datasets of 40k trajectories, which are then used to train agnostic offline MARL algorithms under identical conditions.

Data Augmentation Baselines and Offline MARL Methods. We compare CODI against several strong data augmentation baselines. The first is the **Original** dataset, which uses no augmentation and serves as a reference. To enhance the dataset with higher-return trajectories, single-agent-based method **MBTS** [9] stitches existing trajectory segments using predicted joint actions. Going beyond action generation, **MADiff** [46] employs a diffusion model conditioned on high returns to synthesize joint observation sequences. For more balanced trajectory generation, **MADiTS** [39] iteratively uses integrated gradients [33] to identify underperforming agents and then re-denoising their sub-trajectories. To actively improve balance, **CODI** $_{\text{w/o Com.}}$ is a variant of CODI that conditions the diffusion model on agent-quality labels but disables compositional diffusion, testing the necessity of this mechanism. **CODI** $_{\text{w Pri.}}$ replaces the predicted labels with privileged information, ground-truth one-hot labels indicating the expert agent, isolating the effect of LLM-distillation. More details are provided in Appendix B.

To evaluate the effect of each augmented dataset, we employ four representative offline MARL algorithms. **Behavior Cloning (BC)** [26] provides a direct measure of data quality by imitating the entire dataset. To mitigate the impact of low-quality data, we then use **50%BC** [44], which enhances BC by selectively cloning only the top 50% of trajectories by return. For more advanced evaluation, we use **OMAR** [25], which addresses non-concavity in the value function by hybridizing first-order policy gradients with zeroth-order optimization. Finally, we include **OMIGA** [36], which combines

Table 1: Data augmentation results (mean ± std) on four offline MARL methods across four tasks. Returns are normalized as $\frac{R-R_{\text{lo}}}{R_{\text{hi}}-R_{\text{lo}}}$, where R_{lo} is the average return of the learned imbalanced dataset, and R_{hi} is the return of the balanced expert behavior policy. The best result in each column, excluding the privileged method CODI $_{\text{w Pri.}}$ (denoted in gray), is highlighted in bold.

Envs	Algs	Original	MBTS	MADiff	MADiTS	CODI $_{\text{w/o Com.}}$	CODI $_{\text{w Pri.}}$	CODI
CN	BC	$0.21_{\pm 0.06}$	$-1.34_{\pm 0.05}$	$0.33_{\pm 0.02}$	$0.33_{\pm 0.02}$	$0.36_{\pm 0.04}$	$0.59_{\pm 0.09}$	$\mathbf{0.58_{\pm 0.09}}$
	50%BC	$-0.02_{\pm 0.03}$	$-0.02_{\pm 0.03}$	$-0.25_{\pm 0.06}$	$-0.36_{\pm 0.13}$	$0.11_{\pm 0.14}$	$0.56_{\pm 0.07}$	$\mathbf{0.43_{\pm 0.05}}$
	OMAR	$-4.06_{\pm 1.20}$	$-0.27_{\pm 0.48}$	$0.26_{\pm 0.36}$	$-0.10_{\pm 0.95}$	$0.35_{\pm 0.43}$	$0.76_{\pm 0.36}$	$\mathbf{0.50_{\pm 0.83}}$
	OMIGA	$0.66_{\pm 0.19}$	$0.84_{\pm 0.07}$	$0.93_{\pm 0.07}$	$0.92_{\pm 0.01}$	$0.94_{\pm 0.04}$	$1.01_{\pm 0.06}$	$\mathbf{0.96_{\pm 0.04}}$
World	BC	$0.43_{\pm 0.15}$	$-0.49_{\pm 0.09}$	$-0.18_{\pm 0.24}$	$-0.25_{\pm 0.05}$	$0.31_{\pm 0.18}$	$0.51_{\pm 0.11}$	$\mathbf{0.61_{\pm 0.16}}$
	50%BC	$0.06_{\pm 0.26}$	$-0.49_{\pm 0.19}$	$-0.50_{\pm 0.10}$	$-0.52_{\pm 0.03}$	$0.00_{\pm 0.11}$	$0.27_{\pm 0.10}$	$\mathbf{0.38_{\pm 0.21}}$
	OMAR	$-0.88_{\pm 0.71}$	$-1.55_{\pm 0.11}$	$-1.02_{\pm 0.47}$	$-0.92_{\pm 0.49}$	$-0.93_{\pm 0.37}$	$-0.43_{\pm 0.38}$	$\mathbf{-0.40_{\pm 0.64}}$
	OMIGA	$0.66_{\pm 0.10}$	$-3.36_{\pm 0.81}$	$-3.60_{\pm 0.17}$	$-3.46_{\pm 0.27}$	$-2.42_{\pm 1.10}$	$-2.07_{\pm 1.73}$	$\mathbf{0.93_{\pm 0.14}}$
3m	BC	$0.52_{\pm 0.25}$	$0.22_{\pm 0.00}$	$0.45_{\pm 0.11}$	$0.57_{\pm 0.22}$	$0.61_{\pm 0.28}$	$0.78_{\pm 0.08}$	$\mathbf{0.75_{\pm 0.03}}$
	50%BC	$0.28_{\pm 0.02}$	$0.28_{\pm 0.02}$	$0.48_{\pm 0.02}$	$0.36_{\pm 0.14}$	$0.37_{\pm 0.07}$	$0.49_{\pm 0.12}$	$\mathbf{0.58_{\pm 0.09}}$
	OMAR	$0.24_{\pm 0.05}$	$0.08_{\pm 0.10}$	$0.45_{\pm 0.15}$	$0.66_{\pm 0.23}$	$0.63_{\pm 0.29}$	$0.95_{\pm 0.04}$	$\mathbf{0.72_{\pm 0.09}}$
	OMIGA	$0.36_{\pm 0.04}$	$0.34_{\pm 0.03}$	$0.28_{\pm 0.09}$	$0.37_{\pm 0.03}$	$0.26_{\pm 0.05}$	$0.44_{\pm 0.07}$	$\mathbf{0.40_{\pm 0.10}}$
Zerg_3v4	BC	$0.74_{\pm 0.00}$	$0.69_{\pm 0.68}$	$0.96_{\pm 0.31}$	$0.87_{\pm 0.19}$	$1.01_{\pm 0.35}$	$0.95_{\pm 0.19}$	$\mathbf{1.07_{\pm 0.12}}$
	50%BC	$1.11_{\pm 0.20}$	$0.93_{\pm 0.06}$	$0.87_{\pm 0.35}$	$0.96_{\pm 0.10}$	$1.10_{\pm 0.10}$	$1.23_{\pm 0.16}$	$\mathbf{1.25_{\pm 0.07}}$
	OMAR	$0.62_{\pm 0.08}$	$0.32_{\pm 0.07}$	$0.63_{\pm 0.06}$	$0.59_{\pm 0.06}$	$0.55_{\pm 0.10}$	$0.68_{\pm 0.28}$	$\mathbf{0.65_{\pm 0.10}}$
	OMIGA	$0.39_{\pm 0.12}$	$0.42_{\pm 0.17}$	$\mathbf{0.63_{\pm 0.05}}$	$0.59_{\pm 0.24}$	$0.56_{\pm 0.14}$	$0.46_{\pm 0.15}$	$0.62_{\pm 0.40}$
Average		0.08	-0.21	0.05	0.04	0.24	0.45	**0.63**

multi-agent value decomposition with implicit local regularization for stable off-policy learning. All methods are evaluated over three random seeds. More details are provided in Appendix C.

4.2 Competitive Results

In this section, we present the comprehensive overall data augmentation results of CODI, its ablations, and the baseline methods, on four offline MARL methods across four distinct tasks. To ensure clarity given varying return scales across different tasks, we report normalized returns calculated as $\frac{R-R_{\text{lo}}}{R_{\text{hi}}-R_{\text{lo}}}$, where R_{lo} is the average return of the learned imbalanced dataset, and R_{hi} is the return of the balanced expert behavior policy. A higher normalized return indicates performance closer to the expert policy. These normalized metrics allow for a more equitable comparison across different environments and dataset configurations. As shown in Table 1, learning from the Original dataset yields an average return similar to the dataset quality itself (improving by only 0.08), demonstrating the clear necessity of data augmentation for performance gains under imbalanced settings. MBTS performs even worse than the original dataset, indicating that in the presence of severely imbalanced data, merely stitching existing trajectory segments via predicted joint actions without actively generating new high-quality segments is fundamentally insufficient. In contrast, MADiff employs diffusion to generate entirely new joint observation sequences, and MADiTS further incorporates a re-denoising process targeting underperforming agents. While these methods achieve improvements with certain offline algorithms on specific tasks (e.g., BC and OMIGA on CN), their average performance remains largely stagnant, suggesting that the standard diffusion architecture requires substantial modification to address severe imbalance effectively.

The variant CODI $_{\text{w/o Com.}}$, which conditions the diffusion model on agent-quality labels but disables compositional diffusion, achieves a noticeable performance uplift. This confirms that injecting coordination information via conditioning is beneficial. To mitigate the OOD issue, the full CODI method integrates compositional diffusion, substantially enhancing the model's capability to generate balanced, high-quality data under severe imbalance. Overall, CODI recovers 63% of the performance achieved by the balanced expert policy. Notably, on the challenging Zerg_3v4 task, CODI even surpasses the expert policy's performance when evaluated with BC and 50%BC, underscoring its effectiveness. Furthermore, CODI outperforms CODI $_{\text{w Pri.}}$, which utilizes privileged ground-truth labels. This result indicates that the LLM-distilled agent-quality labels are more flexible and robust than raw one-hot annotations, by capturing richer, more nuanced quality assessments. More results are provided in Appendix F.

4.3 Case Study

To provide deeper insight into the functionality of CODI, we present a detailed case study analyzing the learning and the trajectory generation process in the CN task. This analysis aims to elucidate how our method transforms suboptimal, imbalanced demonstrations into high-quality cooperative trajectories through its combination of LLM reasoning and trajectory generation capabilities.

First, we visualize and compare the original imbalanced trajectories with those generated by our method. As shown in Figure 4(a), a real imbalanced trajectory (top) exhibits a typical failure mode commonly encountered in offline multi-agent datasets: only the blue agent (Agent 2) demonstrates purposeful navigation towards the landmark with a clear strategy, while the other two agents

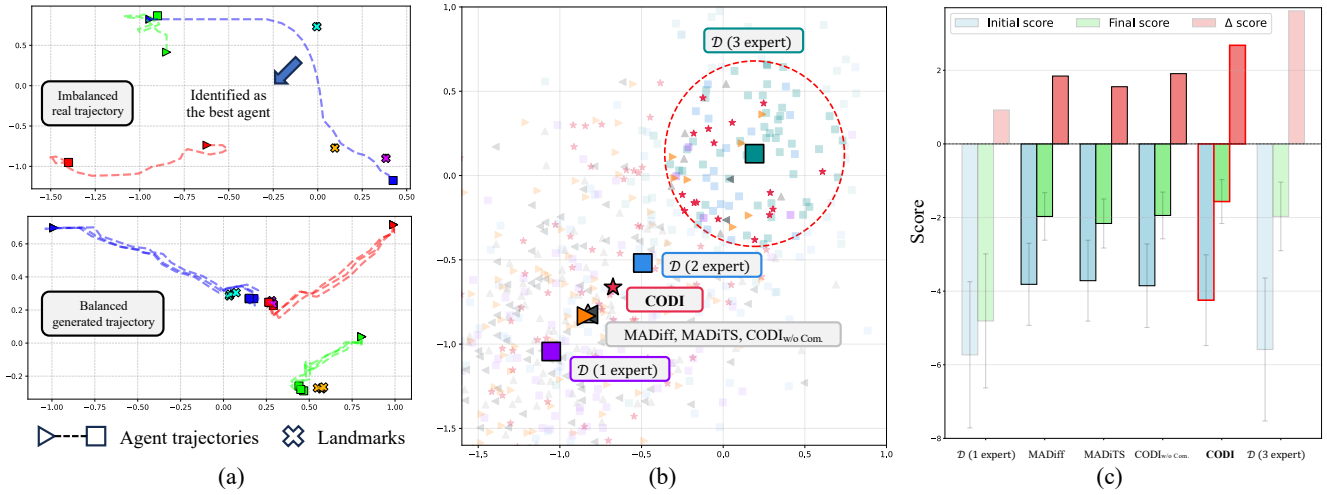


Figure 4: A case study on the CN task. (a) Trajectory visualization: a real imbalanced trajectory (top) with only one agent approaching the landmarks versus a balanced CODI-generated trajectory (bottom) where all agents navigate to the landmarks. (b) Trajectory embeddings: CODI brings the mean embedding (red star) closer to the balanced real dataset (green square), and yields more better trajectories (red circle). (c) Score improvement: CODI yields the highest improvement across methods.

exhibit sub-optimal, uncoordinated movements characterized by oscillatory behavior and inefficient path planning. This lack of cooperation and the presence of significant performance disparity among agents pose a fundamental challenge for offline MARL algorithms, as they may learn to emulate these suboptimal behaviors rather than discovering truly coordinated strategies. To address this challenge, CODI first employs a LLM reasoning process. Given a segment of this imbalanced trajectory, the LLM module processes the structured prompt, and outputs detailed analysis:

> **Prompt**: You are an expert cooperative-navigation coach. This is a joint trajectory segment with T=8 timesteps and N=3 agents. ··· Agent 2 Timeline: t=0: action=no-op; nearest landmark id=1; distance to nearest landmark=0.407; ··· t=7: action=move up; nearest landmark id=1; distance to nearest landmark=0.615;
> FINAL INSTRUCTION: ··· Output the index of the most expert-like agent in format ···
> **Output**: ··· Agent 0 and Agent 1 both allowed their distances to increase overall, with Agent 0 showing poor performance by increasing its distance consistently ··· Agent 2 demonstrates the most expert-like behavior by achieving the greatest overall decrease in distance to its nearest landmark in the early timesteps. It successfully moves significantly closer at first (from 0.407 to 0.200), showing a purposeful strategy ···

And it correctly annotating the blue agent as the best one for subsequent distillation. Compared with the real trajectory, the trajectory generated by CODI (bottom) showcases efficient, coordinated navigation among all three agents, providing an intuitive demonstration of its capability. To verify the fidelity of the generated trajectory, we render it from the observations of each agent, confirming strong consistency despite minor perceptual differences.

Moving beyond individual examples, we perform a quantitative distributional analysis. We derive embeddings for the observation-action sequences of all real and generated trajectories using contrastive learning [11], as shown in Figure 4(b). The feature extractor is trained on datasets collected by 1, 2, and 3 expert agents, with their embeddings forming corresponding reference clusters. For visual clarity, each cluster displays 100 randomly sampled trajectories. While baseline data augmentation methods produce embeddings clustered near the region of severe imbalance (purple square), CODI successfully shifts the distribution towards the balanced, expert domain. The closer proximity of CODI's mean embedding (red star) to better datasets (blue and green squares) indicates a systematic improvement in trajectory quality. More importantly, the dense clustering of CODI's samples within the expert region (red circle) demonstrates its ability to frequently generate high-quality trajectories, directly addressing the core limitation of sparse positive examples in imbalanced datasets. This distributional shift is crucial for effective policy learning, as it expands the coverage of from low-quality to high-quality data.

Beyond trajectory-level features, we directly measure the cooperative improvement brought by the augmented data. We define a CN score as the negative distance between agents and landmarks at a given step. Figure 4(c) shows the average initial (blue bars) and final (green bars) scores for different datasets, as well as the improvement (red bars). CODI achieves the most significant score improvement, second only to the balanced expert dataset. Notably, the initial scores of CODI-generated trajectories are lower than those of the baselines, yet they achieve the highest final scores. This provides a key insight into CODI's superiority: even when sampling from challenging initial states, our method, empowered by its label conditioning and compositional diffusion, can generate trajectories with valid performance gains. In contrast, baselines lack this strong generalization capability, generating segments often

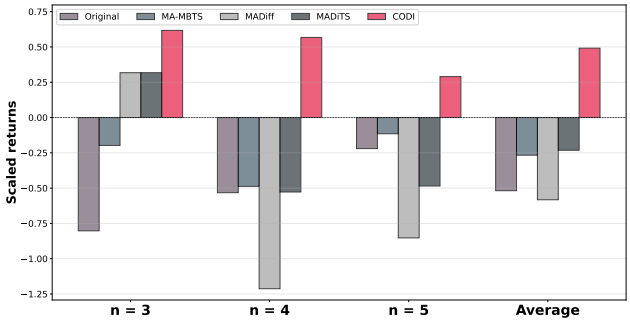


Figure 5: Offline MARL performance improvement achieved by data augmentation methods under varying numbers of agents in the CN environment.

fail to pass dynamics validation and are discarded, resulting in a final dataset with limited coverage in high-improvement regions. This case study highlights the effectiveness of CODI in generating high-quality, cooperative trajectories, which ultimately boosts the performance of agnostic offline MARL algorithms.

4.4 The Impact of Agent Numbers

A key factor of our study is understanding how the number of agents affects performance. To this end, we extend our experiments to CN environments with 3, 4, and 5 agents. In each configuration, we maintain the same severe imbalance protocol where exactly one randomly selected agent executes the expert policy while all others take random actions. As shown in Figure 5, the performance of baseline methods reveals significant scalability limitations. Both the Original dataset and MA-MBTS yield negative scaled returns across all team sizes. While MADiff and MADiTS achieve positive scaled returns in the 3-agent setting, their performance degrades substantially, becoming negative as the number of agents increases to 4 and 5. This trend underscores the poor scalability of standard diffusion architectures in this context. As the number of agents grows, the coordination space expands exponentially, and these methods struggle to generate effective, balanced joint trajectories without specialized mechanisms to address severe imbalance. In contrast, our proposed CODI demonstrates consistent and robust scalability. Although the performance improvement modestly decreases as the team size increases from 3 to 5 agents, CODI remains the only method to consistently achieve positive scaled returns across all configurations. This result highlights the importance of leveraging LLM-distilled agent-quality labels and the compositional diffusion process, to effectively preserve the ability to generate high-quality data even as coordination complexity grows substantially.

5 RELATED WORK

5.1 Offline MARL

Offline MARL [6, 40] learns cooperative policies from static datasets. A common approach uses policy constraints to address distribution shift. For example, MABCQ [12] applies value deviation and transfer normalization in a decentralized framework. CFCQL [30] uses per-agent conservative regularization, while OMAR [25] combines policy gradients with zeroth-order optimization to escape local optima. OMIGA [36] transforms global value regularization into implicit local constraints. We evaluate OMAR and OMIGA as advanced baselines. Currently, diffusion models have been explored to improve sample efficiency and model complex dynamics [43]. A key limitation of existing methods is their sensitivity to dataset quality, particularly under agent trajectory imbalances, highlighting the need for robust data augmentation.

5.2 Data Augmentation for Offline RL

Data augmentation addresses limited dataset quality in Offline RL. Some methods perform trajectory stitching to create near-optimal trajectories from sub-optimal ones. MBTS [9] uses a learned model and value function, while BATS [1] plans with an environment model. DiffStitch [15] employs diffusion models to bridge trajectory segments. Multi-agent extensions like MADiff [46] and MADiTS [39] generate trajectories conditioned on high returns but often fail to synthesize coordinated behaviors under data imbalance, underscoring the need for specialized multi-agent augmentation.

5.3 LLMs for Agent Quality Labeling in MARL

To provide data augmentation methods with additional agent quality information, a promising approach is to utilize LLMs [8, 21]. Recent works have explored leveraging the rich semantic and decision-making knowledge of LLMs to address credit assignment in RL. LaRe [27] generates symbolic latent rewards, LCA [17] produces agent-specific rewards, and LLM-MCA [22] decomposes global rewards numerically. DPM [14] uses LLM preferences for trajectories and agent contributions, while SemDiv [16] verifies generated agent behaviors. QLLM [18] focuses on efficient LLM quantization. These works demonstrate a growing trend of using LLMs for their semantic knowledge and heuristic evaluation to label agent quality.

6 FINAL REMARKS

In this work, we propose CODI, a novel framework that leverages LLMs and compositional diffusion to address the critical challenge of agent-quality imbalance in offline MARL. By distilling LLM-derived coordination knowledge into fine-grained quality labels and employing a conditional diffusion model guided by both return-to-go and these compositional labels, CODI effectively generates high-quality, balanced trajectories that generalize beyond the severely imbalanced datasets. Empirical results on challenging datasets with severe quality imbalance demonstrate that CODI significantly recovers the performance gap to expert policies, substantially outperforming existing baselines. While CODI demonstrates strong performance, its current formulation assumes that improved individual behaviors compositionally enhance team performance. This assumption may not hold in environments with strong social dilemmas, where individual and collective interests can conflict. Extending our approach to such non-monotonic settings remains an important future direction. Also, the integration of more advanced generative models and external knowledge from LLMs offers a promising path for further enhancing coordination synthesis. Future work will focus on extending this approach to more complex multi-agent systems, such as those involving embodied agents in open-world scenarios [5].

REFERENCES

[1] Ian Char, Viraj Mehta, Adam Villaflor, John M Dolan, and Jeff Schneider. 2021. BATS: Best Action Trajectory Stitching. In *Advances in Neural Information Processing Systems*.

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS Deep Learning and Representation Learning Workshop*.

[3] Yilun Du, Shuang Li, and Igor Mordatch. 2020. Compositional Visual Generation with Energy Based Models. In *Advances in Neural Information Processing Systems*.

[4] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Nicolaus Foerster, and Shimon Whiteson. 2023. SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning. In *NeurIPS Datasets and Benchmarks Track*.

[5] Zhaohan Feng, Ruiqi Xue, Lei Yuan, Yang Yu, Ning Ding, Meiqin Liu, Bingzhao Gao, Jian Sun, Xinhu Zheng, and Gang Wang. 2025. Multi-agent Embodied AI: Advances and Future Directions. *preprint arXiv:2505.05108* (2025).

[6] Claude Formanek, Asad Jeewa, Jonathan P. Shock, and Arnu Pretorius. 2023. Off-the-Grid MARL: Datasets and Baselines for Offline Multi-Agent Reinforcement Learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 2442–2444.

[7] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large Language Model Based Multi-agents: A Survey of Progress and Challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*. 8048–8057.

[8] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large Language Model Based Multi-agents: A Survey of Progress and Challenges. In *International Joint Conference on Artificial Intelligence*, Vol. 56. 8048–8057.

[9] Charles A Hepburn and Giovanni Montana. 2022. Model-based Trajectory Stitching for Improved Offline Reinforcement Learning. In *Offline RL Workshop at Neural Information Processing Systems*.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*. 6840–6851.

[11] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2021. A Survey on Contrastive Self-supervised Learning. *preprint arXiv:2011.00362* (2021).

[12] Jiechuan Jiang and Zongqing Lu. 2023. Offline Decentralized Multi-Agent Reinforcement Learning. In *European Conference on Artificial Intelligence*, Vol. 26. 1148–1155.

[13] Tao Jiang, Lei Yuan, Lihe Li, Cong Guan, Zongzhang Zhang, and Yang Yu. 2024. Multi-Agent Domain Calibration with a Handful of Offline Data. In *Advances in Neural Information Processing Systems*. 69607–69636.

[14] Sehyeok Kang, Yongsik Lee, Minu Kim, Jihwan Oh, Song Chong, and Se-Young Yun. 2025. DPM: Dual Preferences-based Multi-Agent Reinforcement Learning. https://openreview.net/forum?id=VzuPnoSKQ1

[15] Guanghe Li, Yixiang Shan, Zhengbang Zhu, Ting Long, and Weinan Zhang. 2024. DiffStitch: Boosting Offline Reinforcement Learning with Diffusion-based Trajectory Stitching. In *International Conference on Machine Learning*, Vol. 41. 28597–28609.

[16] Lihe Li, Lei Yuan, Pengsen Liu, Tao Jiang, and Yang Yu. 2025. LLM-Assisted Semantically Diverse Teammate Generation for Efficient Multi-agent Coordination. In *Proceedings of the Forty-second International Conference on Machine Learning*.

[17] Muhan Lin, Shuyang Shi, Yue Guo, Vaishnav Tadiparthi, Behdad Chalaki, Ehsan Moradi Pari, Simon Stepputtis, Woojun Kim, Joseph Campbell, and Katia Sycara. 2025. Speaking the Language of Teamwork: LLM-Guided Credit Assignment in Multi-Agent Reinforcement Learning. *preprint arXiv:2502.03723* (2025).

[18] Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. 2024. QLLM: Accurate and Efficient Low-Bitwidth Quantization for Large Language Models. In *International Conference on Learning Representations*.

[19] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. 2022. Compositional Visual Generation with Composable Diffusion Models. In *European Conference on Computer Vision*. 423–439.

[20] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*. 6379–6390.

[21] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2025. Large Language Models: A Survey. *preprint arXiv:2402.06196* (2025).

[22] Kartik Nagpal, Dayi Dong, and Negar Mehr. 2025. Leveraging Large Language Models for Effective and Explainable Multi-Agent Credit Assignment. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*. 1501–1510.

[23] Frans A Oliehoek, Christopher Amato, et al. 2016. *A Concise Introduction to Decentralized POMDPs*. Vol. 1. Springer.

[24] Afshin Oroojlooy and Davood Hajinezhad. 2023. A Review of Cooperative Multi-agent Deep Reinforcement Learning. *Applied Intelligence* 53, 11 (2023), 13677–13722.

[25] Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. 2022. Plan Better Amid Conservatism: Offline Multi-Agent Reinforcement Learning with Actor Rectification. In *International Conference on Machine Learning*. 17221–17237.

[26] Dean A Pomerleau. 1988. Alvinn: An Autonomous Land Cehicle in a Neural Network. In *Advances in Neural Information Processing Systems*. 305–313.

[27] Yun Qu, Yuhang Jiang, Boyuan Wang, Yixiu Mao, Cheems Wang, Chang Liu, and Xiangyang Ji. 2025. Latent Reward: LLM-Empowered Credit Assignment in Episodic Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 47. 20095–20103.

[28] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-agent Reinforcement Learning. In *International Conference on Machine Learning*. 4295–4304.

[29] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *AAMAS*. 2186–2188.

[30] Jianzhun Shao, Yun Qu, Chen Chen, Hongchang Zhang, and Xiangyang Ji. 2023. Counterfactual Conservative Q Learning for Offline Multi-agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 36. 77290 – 77312.

[31] Yang Song and Stefano Ermon. 2019. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*. 11918–11930.

[32] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*.

[33] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *International Conference on Machine Learning*. 3319–3328.

[34] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *International Conference on Learning Representations*.

[35] Lirui Wang, Jialiang Zhao, Yilun Du, Edward H. Adelson, and Russ Tedrake. 2024. PoCo: Policy Composition from and for Heterogeneous Robot Learning. In *Robotics: Science and Systems*.

[36] Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. 2023. Offline Multi-Agent Reinforcement Learning with Implicit Global-to-Local Value Regularization. In *Advances in Neural Information Processing Systems*, Vol. 36. 52413–52429.

[37] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2024. Diffusion Models: A Comprehensive Survey of Methods and Applications. *ACM Comput. Surv.* 56, 4 (2024), 105:1–105:39.

[38] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*, Vol. 35. 24611–24624.

[39] Lei Yuan, Yuqi Bian, Lihe Li, Ziqian Zhang, Cong Guan, and Yang Yu. 2025. Efficient Multi-agent Offline Coordination via Diffusion-based Trajectory Stitching. In *International Conference on Learning Representations*.

[40] Lei Yuan, Ziqian Zhang, Lihe Li, Cong Guan, and Yang Yu. 2023. A Survey of Progress on Cooperative Multi-agent Reinforcement Learning in Open Environment. *Science China Information Sciences (SCIS)* (2023).

[41] Hongxin Zhang, Zeyuan Wang, Qiushi Lyu, Zheyuan Zhang, Sunli Chen, Tianmin Shu, Behzad Dariush, Kwonjoon Lee, Yilun Du, and Chuang Gan. 2025. COMBO: Compositional World Models for Embodied Multi-Agent Cooperation. In *International Conference on Learning Representations*.

[42] Ruiqi Zhang, Jing Hou, Florian Walter, Shangding Gu, Jiayi Guan, Florian Röhrbein, Yali Du, Panpan Cai, Guang Chen, and Alois Knoll. 2024. Multi-Agent Reinforcement Learning for Autonomous Driving: A Survey. *preprint arXiv:2408.09675* (2024).

[43] Yang Zhang, Xinran Li, Jianing Ye, Delin Qu, Shuang Qiu, Chongjie Zhang, Xiu Li, and Chenjia Bai. 2025. Revisiting Multi-Agent World Modeling from a Diffusion-Inspired Perspective. In *Advances in Neural Information Processing Systems*.

[44] Zhilong Zhang, Yihao Sun, Junyin Ye, Tianshuo Liu, Jiaji Zhang, and Yang Yu. 2024. Flow to Better: Offline Preference-based Reinforcement Learning via Preferred Trajectory Generation. In *International Conference on Learning Representations*.

[45] Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. 2024. RoboDreamer: Learning Compositional World Models for Robot Imagination. In *Forty-first International Conference on Machine Learning*, Vol. 41. 61885 – 61896.

[46] Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. 2024. MADiff: Offline Multi-agent Learning with Diffusion Models. In *Advances in Neural Information Processing Systems*.

Appendix: Boosting Offline MARL under Imbalanced Datasets via Compositional Diffusion Models

Anonymous Author(s)
Submission Id: 574

ACM Reference Format:
Anonymous Author(s). 2026. Appendix: Boosting Offline MARL under Imbalanced Datasets via Compositional Diffusion Models. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages.

A IMPLEMENTATION DETAILS OF CODI

A.1 Agent Quality Labeling via LLM Distillation

To efficiently obtain fine-grained agent-quality annotations from large offline datasets, we employ an LLM-based distillation framework that transfers the reasoning ability of a powerful language model into a lightweight label model. This process consists of two stages: (i) annotating a subset of trajectory segments using an LLM, and (ii) training a neural classifier to imitate these annotations for large-scale automatic labeling.

LLM annotation. We use the GPT-4o-mini model to generate quality labels for multi-agent trajectory segments. Each input to the LLM is a structured prompt $[\texttt{desc}, \tau_j^{\texttt{text}}, \texttt{inst}]$, where $\tau_j^{\texttt{text}}$ is the natural language description corresponding to the segment $\tau_j \in \mathcal{D}$, `desc` basic information of environment, agents, and task, and `inst` a concise instruction for the LLM to select the best-performing agent (see Appendix E for more prompt details). The LLM is accessed through the OpenAI Python interface by creating chat completions with the prompt as a user message, using the default temperature setting to maintain moderate response diversity. From the returned text, we extract the index of the agent identified as the best-performing and convert it into a one-hot label vector.

To construct the labeling subset, we randomly sample a small proportion of trajectory segments from the offline datasets. Each segment $\tau_j = (\boldsymbol{o}_t, \boldsymbol{a}_t, r_t, \ldots, \boldsymbol{o}_{t+H-1}, \boldsymbol{a}_{t+H-1}, r_{t+H-1})$ contains H consecutive time steps of joint observations, joint actions, and rewards. Segments may have different valid lengths with the invalid timesteps padded by zero, and may partially overlap with each other, which ensures sufficient coverage and generalization. By estimation, roughly 10% or fewer of the total data is used for annotation, with larger samples adopted only for the CN4 and CN5 tasks to account for their increased number of agents.

Label model training. The distilled label model f_{label} takes as input a joint multi-agent trajectory segment and predicts the probability that each agent exhibits the most expert-like behavior. For efficiency and generalization, the model shares a compact structure and parameterization across agents: a single-layer GRU encoder followed by a two-layer MLP head applied independently to each agent's temporal features. The resulting logits are normalized by a softmax function across agents to produce a probability distribution over expert identities. The model is trained as a supervised classifier using the cross-entropy loss:

$$\mathcal{L}_{\text{label}} = \mathbb{E}_{\tau_j \sim \mathcal{D}_{\text{label}}}\Big[-\sum_{i=1}^{n} \mathbb{I}\{y_j = i\} \log z_j^i \Big], \tag{1}$$

where y_j denotes the LLM-provided label for segment τ_j and $\boldsymbol{z}_j = (z_j^1, ..., z_j^n) = f_{\text{label}}(\tau_j)$ the predicted probabilities for all n agents. Training employs the Adam [3] optimizer with early stopping on a held-out validation split.

Empirically, using the label model's predictions as diffusion model conditions yields performance comparable to, or even exceeding, the use of ground-truth quality labels, owing to the LLM's ability to capture the agents' temporal patterns associated with high-quality behaviors.

A.2 Pseudocode and Hyperparameters

We provide pseudocode of CODI and hyperparameters of different settings in this section.

Algorithm 1 CODI Training Phase

1: **Input:** Offline dataset $\mathcal{D} = \{\tau_j\}_{j=1}^m$, LLM oracle
2: **Output:** Trained models f_{label}, ϵ_θ, f_{inv}, f_{fwd}, f_{rwd}
3: **Step 1: Agent Quality Labeling**
4: Sample subset $\mathcal{D}_{\text{subset}} \subset \mathcal{D}$ (e.g., $|\mathcal{D}_{\text{subset}}| < 0.1|\mathcal{D}|$)
5: Initialize labeled dataset $\mathcal{D}_{\text{label}} \leftarrow \emptyset$
6: **for** each $\tau_j \in \mathcal{D}_{\text{subset}}$ **do**
7: Construct prompt: $[\texttt{desc}, \tau_j^{\texttt{text}}, \texttt{inst}]$ (See Appendix E)
8: Get LLM annotation: $y_j \leftarrow \text{LLM}(\text{prompt})$
9: Store labeled data: $\mathcal{D}_{\text{label}} \leftarrow \mathcal{D}_{\text{label}} \cup \{(\tau_j, y_j)\}$
10: **end for**
11: Train quality labeler f_{label} with $\mathcal{D}_{\text{label}}$ via Eq. (1)
12: **Step 2: Compositional Diffusion Training**
13: **repeat**
14: Sample trajectory segment $\tau \sim \mathcal{D}$
15: Extract $\mathbf{x} = \boldsymbol{o}_{t:t+H-1}$, $\hat{R}$, $\boldsymbol{z} = f_{\text{label}}(\tau)$
16: Sample noise level $k \sim \text{Uniform}(1, K)$, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
17: Compute diffusion loss: $\mathcal{L}_{\text{diff}} = \|\epsilon - \epsilon_\theta(\mathbf{x}_k, k|\hat{R}, \boldsymbol{z})\|^2$
18: Compute dynamics losses:
19: $\mathcal{L}_{\text{inv}} = \|\boldsymbol{a}_t - f_{\text{inv}}(\boldsymbol{o}_t, \boldsymbol{o}_{t+1})\|^2$
20: $\mathcal{L}_{\text{fwd}} = \|\boldsymbol{o}_{t+1} - f_{\text{fwd}}(\boldsymbol{o}_t, \boldsymbol{a}_t)\|^2$
21: $\mathcal{L}_{\text{rwd}} = \|r_t - f_{\text{rwd}}(\boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1})\|^2$
22: Update all models: $\theta \leftarrow \theta - \nabla_\theta(\mathcal{L}_{\text{diff}} + \mathcal{L}_{\text{inv}} + \mathcal{L}_{\text{fwd}} + \mathcal{L}_{\text{rwd}})$
23: **until** convergence

Algorithm 1 outlines the complete training pipeline. In Step 1 (Lines 3-10), we first sample a subset $\mathcal{D}_{\text{subset}}$ from the offline dataset $\mathcal{D}$ and initialize the labeled dataset $\mathcal{D}_{\text{label}}$ as empty. For each trajectory τ_j in the subset, we convert it to text format, construct a

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.

prompt containing task description and annotation instructions, query the LLM oracle to obtain quality labels y_j, and store the annotated data in $\mathcal{D}_{\text{label}}$. Using this labeled data, we train a quality labeler f_{label} to predict trajectory quality features. In Step 2 (Lines 12-22), we perform compositional diffusion training. In each iteration, we sample a trajectory segment from $\mathcal{D}$, extract the observation sequence $\mathbf{x}$, return $\hat{R}$, and quality features $\boldsymbol{z}$. We then sample a noise level k and random noise ϵ, and compute the diffusion loss $\mathcal{L}_{\text{diff}}$, which trains the noise prediction network $\epsilon\theta$ to denoise the observation sequence conditioned on both the return and quality features. Simultaneously, we compute dynamics model losses: inverse dynamics loss $\mathcal{L}_{\text{inv}}$, forward dynamics loss $\mathcal{L}_{\text{fwd}}$, and reward prediction loss $\mathcal{L}_{\text{rwd}}$. Finally, we update all model parameters by jointly optimizing these loss functions.

Algorithm 2 details the trajectory generation process. The algorithm begins by sampling an initial observation from the dataset and initializes the trajectory with a target return-to-go. The core generation occurs in segments of length H (Lines 7-35). The generation process consists of three key components. First, in the compositional denoising phase (Lines 13-18), the algorithm initializes a noisy segment and performs iterative denoising from noise level K down to 1. The critical innovation is the compositional noise prediction $\hat{\epsilon}^{\text{CODI}}_{\text{comp}}$, which blends unconditional and conditional denoising signals using multiple quality conditions $\boldsymbol{z}^{(i)}_{\text{gen}}$ weighted by w_i. Second, each generated segment undergoes rigorous dynamics consistency checking in the verification loop (Lines 20-30). The loop uses the trained inverse dynamics model f_{inv} to infer actions between consecutive observations, then verifies that the forward dynamics model f_{fwd} can accurately predict the next observation. Segments failing this consistency check with deviation exceeding δ are resampled up to M times. Third, for valid segments (Lines 32-35), the algorithm reconstructs the complete transition sequence using f_{inv} and f_{rwd} to generate actions and rewards respectively. The return-to-go is updated recursively, and generation continues autoregressively from the last observation. The final generated trajectories augment the offline dataset for subsequent multi-agent reinforcement learning, enabling policy improvement through synthetic high-quality data.

Key hyperparameters in our implementation are listed here for completeness. For agent quality labeling via LLM distillation, the label model is trained using the Adam optimizer with learning rate 2×10^{-4}. For most tasks, 1200 batches each containing 32 trajectory segments of length $H = 8$ are sampled from the dataset $\mathcal{D}$ and relabeled with the LLM; GRU hidden size and MLP dropout rate are respectively set to 128 and 0.1. For the CN4 and CN5 tasks, a larger sample of 6400 batches is employed, and GRU hidden size and MLP dropout rate are respectively set to 256 and 0.0. In all tasks, the label model is trained for up to 150k gradient steps with a 10% validation split for early stopping. For training diffusion models and running offline MARL methods, we following the settings of established frameworks, as discussed in the next section. For the generation process, we set the RTG values as -60 for CN3, -120 for CN4, -210 for CN5, -70 for World, 9 for 3m, and 7 for Zerg_3v4. The return distribution of given datasets are provided in Section F.1. The reconstruction thresholds are set to 0.01 for CN tasks, 0.03 for World and Zerg_3v4, and 0.075 for 3m. Other hyperparameters are

Algorithm 2 CODI Generation Phase

1: **Input:** Trained models ϵ_θ, f_{inv}, f_{fwd}, f_{rwd}, dataset $\mathcal{D}$
2: **Parameters:** Target RTG $\hat{R}_{\text{gen}}$, segment length H, max resamples M ($M = |\mathcal{D}|$ in our work)
3: **Output:** Generated trajectories $\mathcal{D}_{\text{gen}}$
4: **while** more trajectories needed **do**
5: Sample $\boldsymbol{o}_0 \sim \mathcal{D}$; Initialize $\tau_{\text{gen}} \leftarrow [\boldsymbol{o}_0]$, $\hat{R} \leftarrow \hat{R}_{\text{gen}}$
6: **while** trajectory length < target **do**
7: Initialize segment $\mathbf{x}_K = [\boldsymbol{o}_0, \tilde{\boldsymbol{x}}_1, \cdots, \tilde{\boldsymbol{x}}_{H-1}]$ with noise
8: $m \leftarrow 0$; valid $\leftarrow$ False
9: **while** $m < M$ AND NOT valid **do**
10: **for** $k = K, \ldots, 1$ **do**
11: Compute noise $\hat{\epsilon}^{\text{CODI}}_{\text{comp}} =$
12: $\epsilon_\theta(\mathbf{x}_k, k) + \sum_{i=1}^{n} w_i \left[\epsilon_\theta\left(\mathbf{x}_k, k | \hat{R}, \boldsymbol{z}^{(i)}_{\text{gen}}\right) - \epsilon_\theta(\mathbf{x}_k, k)\right]$
13: $\mathbf{x}_{k-1} = \mathbf{x}_k - \hat{\epsilon}^{\text{CODI}}_{\text{comp}} + \mathcal{N}(\mathbf{0}, \sigma_k^2 \mathbf{I})$
14: **end for**
15: Extract $\mathbf{x}_0 = [\boldsymbol{o}_0, \hat{\boldsymbol{o}}_1, \cdots, \hat{\boldsymbol{o}}_{H-1}]$
16: **Consistency Check**: valid $\leftarrow$ True
17: **for** $t = 0, \ldots, H-2$ **do**
18: $\hat{\boldsymbol{a}}_t \leftarrow f_{\text{inv}}(\hat{\boldsymbol{o}}_t, \hat{\boldsymbol{o}}_{t+1})$; $\hat{\boldsymbol{o}}^{\text{pred}}_{t+1} \leftarrow f_{\text{fwd}}(\hat{\boldsymbol{o}}_t, \hat{\boldsymbol{a}}_t)$
19: **if** $|\hat{\boldsymbol{o}}_{t+1} - \hat{\boldsymbol{o}}^{\text{pred}}_{t+1}| > \delta$ **then**
20: valid $\leftarrow$ False; $m \leftarrow m + 1$; **break**
21: **end if**
22: **end for**
23: **end while**
24: **if** valid **then**
25: **for** $t = 0, \ldots, H-2$ **do**
26: $\hat{\boldsymbol{a}}_t \leftarrow f_{\text{inv}}(\hat{\boldsymbol{o}}_t, \hat{\boldsymbol{o}}_{t+1})$; $\hat{r}_t \leftarrow f_{\text{rwd}}(\hat{\boldsymbol{o}}_t, \hat{\boldsymbol{a}}_t, \hat{\boldsymbol{o}}_{t+1})$
27: $\tau_{\text{gen}} \leftarrow \tau_{\text{gen}} \cup \{(\hat{\boldsymbol{o}}_t, \hat{\boldsymbol{a}}_t, \hat{r}_t)\}$
28: **end for**
29: $\hat{R} \leftarrow \hat{R} - \sum_{t=0}^{H-2} \hat{r}_t$; $\boldsymbol{o}_0 \leftarrow \hat{\boldsymbol{o}}_{H-1}$
30: **else**
31: **break**
32: **end if**
33: **end while**
34: **if** τ_{gen} reaches target length **then**
35: $\mathcal{D}_{\text{gen}} \leftarrow \mathcal{D}_{\text{gen}} \cup \{\tau_{\text{gen}}\}$
36: **end if**
37: **end while**
38: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{\text{gen}}$ {Augment dataset for offline MARL}

consistent with the MADiTS framework. The running time of CODI is almost the same with MADiTS, with about 30 hours for training and 3 hours for generation on 80 TFLOPS GPUs. Also, a single run of CODI incurs a cost of less than $0.10 for OpenAI APIs and less than $50 for the full project.

B IMPLEMENTATION DETAILS OF BASELINES

We conduct comprehensive comparisons between CODI and several strong data augmentation approaches to thoroughly evaluate its effectiveness in addressing agent-quality imbalance in offline MARL. All experiments are implemented on top of the publicly

available codebases of MBTS[1], MADiff[2] and MADiTS[3], maintaining consistency in architectural choices and hyperparameter settings.

The baseline methods encompass a diverse range of augmentation strategies. First, the **Original** dataset serves as our fundamental reference point, representing the raw imbalanced data without any augmentation, which helps quantify the intrinsic challenges posed by behavioral heterogeneity. For single-agent augmentation techniques adapted to multi-agent settings, we implement **MBTS** [2], which operates by identifying promising state pairs across different trajectories and predicting joint actions to connect them, effectively creating new transitions that potentially yield higher returns while maintaining environmental plausibility. Moving to diffusion-based generative approaches, **MADiff** [12] represents a foundational diffusion framework that generates complete joint trajectories conditioned solely on team-level returns. This method employs an attention-based architecture to model complex multi-agent interactions but lacks explicit mechanisms for addressing agent-level quality imbalance. Building upon this, **MADiTS** [10] introduces an iterative refinement process that identifies underperforming agents using integrated gradients [8] and selectively re-denoises their sub-trajectories while preserving high-quality behaviors from other agents, offering a more targeted approach to quality improvement. To systematically evaluate the components of our proposed framework, we include two carefully designed variants of CODI. The first, **CODI** $_{\textbf{w/o Com.}}$, retains the agent-quality conditioning but disables the compositional diffusion mechanism, thereby testing whether fine-grained conditioning alone suffices or the explicit decomposition and composition of team quality into agent-level labels is indeed necessary for effective generalization. The second variant, **CODI** $_{\textbf{w Pri.}}$, replaces the LLM-distilled quality labels with privileged ground-truth information in the form of one-hot expert indicators, allowing us to isolate the impact of label quality and assess the performance of our approach when perfect agent-quality annotations are available. Across all diffusion-based methods, we maintain consistent experimental settings including the same number of denoising steps, identical guidance scales for conditioning, and so on. This rigorous experimental design ensures that observed performance differences can be attributed to the methodological innovations rather than implementation variations.

C IMPLEMENTATION DETAILS OF OFFLINE MARL METHODS

To comprehensively evaluate the effect of each augmented dataset, we employ four representative offline Multi-Agent Reinforcement Learning (MARL) algorithms that span a spectrum of methodological approaches. **Behavior Cloning (BC)** [6] serves as a fundamental baseline, providing a direct measure of data quality by learning a policy that directly imitates the state-action distributions present in the entire dataset. Its performance is highly correlated with the inherent quality and coherency of the dataset, making it a crucial first indicator. To mitigate the potential negative impact of low-quality or suboptimal trajectories within the dataset, we then utilize **50%BC** [11], an enhanced variant that selectively clones only the top 50% of trajectories ranked by their empirical returns. When applied to an augmented dataset, 50%BC first combines the original dataset and the augmented dataset, each containing 20k trajectories, into a pooled set of 40k trajectories, and then selects the top 50% with the highest returns for policy learning. This method tests the dataset's ability to support performance improvements when the learning process is biased towards its highest-returning segments. For a more advanced evaluation that assesses the dataset's suitability for complex value-based optimization, we employ **OMAR** [5]. This algorithm directly addresses a key identified challenge in offline MARL: the non-concavity of the multi-agent value function, which makes policy gradient improvements highly prone to converging to poor local optima. OMAR mitigates this by innovatively combining first-order policy gradients with zeroth-order optimization (specifically, using an evolutionary strategy), which enables a more effective and robust optimization of the conservative value functions over the actor parameters, thereby better escaping local optima. Finally, we include **OMIGA** [9], a state-of-the-art method that elegantly bridges the gap between centralized training with decentralized execution (CTDE) and offline learning constraints. OMIGA's core contribution is its implicit global-to-local value regularization, which provides a principled framework to convert global-level value regularization (aimed at preventing over-estimation on out-of-distribution actions) into equivalent implicit local value regularizations for individual agents. This approach simultaneously enables efficient in-sample learning, integrating multi-agent value decomposition with the necessary offline policy constraints.

All experiments are conducted using the implementations from the publicly available OffPymarl[4], ensuring consistency and reproducibility. We utilize the default hyperparameters as provided by the framework without any task-specific modifications, maintaining an identical evaluation protocol across all datasets to ensure a fair and unbiased comparison.

D COOPERATIVE MULTI-AGENT ENVIRONMENTS

We conduct experiments on four classic cooperative multi-agent tasks (Figure 1), selected to represent a spectrum of challenges in offline MARL. The first two are from the Multi-Agent Particle Environment (MPE) [4], which provides a lightweight, continuous 2D world ideal for rapid prototyping and analysis. The first MPE task is **Cooperative Navigation (CN)**, where three agents must learn to cover three distinct landmarks in a confined space. The core challenge here is for the agents to develop an implicit assignment strategy, dividing the landmarks amongst themselves without direct communication. The action space for MPE agents is discrete, allowing them to choose from five actions: to remain stationary or to apply an acceleration force in each of the four cardinal directions (up, down, left, right). The reward structure is shared, providing a negative penalty for the distance between agents and landmarks. The second MPE task, **World**, is a more complex predator-prey scenario that requires high-level, sequential coordination. Here, three slower cooperative agents must collaborate to capture three faster "prey" agents. We set the prey as random policies for simplicity. This task tests the agents' ability to perform temporally extended

[1]https://github.com/CharlesHepburn1/Model-Based-Trajectory-Stitching
[2]https://github.com/zbzhu99/madiff
[3]https://openreview.net/attachment?id=EpnZEzYDUT&name=supplementary_material
[4]https://github.com/zzq-bot/offline-marl-framework-offpymarl

(a) MPE: CN (b) MPE: World (c) SMAC: 3m (d) SMACv2: Zerg_3v4

Figure 1: Cooperative MARL tasks used in this paper. (a) Cooperative Navigation (CN) in Multi-Agent Particle Environment (MPE) [4]. (b) World in MPE. (c) 3m in StarCraft Multi-Agent Challenge (SMAC) [7]. (d) Zerg_3v4 in SMACv2 [1].

coordination, where individual actions are only meaningful in the context of the team's joint strategy.

The other two tasks are combat scenarios from the StarCraft Multi-Agent Challenge, which present partially observable, discrete-action problems with rich state spaces. The **3m** scenario from the original SMAC [7] is a foundational symmetric battle where three allied marines engage three enemy marines controlled by a built-in AI. The reward function in SMAC is all positive and shaped to guide learning: agents receive a small, positive reward for damaging enemy units' hit points (HP), a medium-sized reward for killing an enemy, and a large, team-wide reward for winning the battle by eliminating all enemies. The primary challenge in 3m is for the agents to learn the "focus fire" strategy-concentrating their attack on a single enemy unit to eliminate threats efficiently rather than spreading damage. This requires agents to overcome the credit assignment problem, correctly attributing the success of a kill to the agents that contributed, and requires robust decentralized execution under partial observability. To evaluate performance in a more difficult setting, we also include **Zerg_3v4** from the more challenging SMACv2 [1]. This scenario presents an asymmetric 3v4 battle where the allied team controls three Zerglings against four enemy Marines. The asymmetry in numbers creates an initial disadvantage for the learning agents. Furthermore, SMACv2 introduces randomized start positions for every episode, a critical feature that prevents agents from memorizing a single, brittle strategy and instead demands the learning of robust, generalizable policies that can adapt to dynamic initial conditions. Success in Zerg_3v4 requires sophisticated micro-tactics, such as kiting (hit-and-run maneuvers), flanking, and dynamic target selection, making it a suitable testbed for our experiments.

E PROMPT ENGINEERING

Task-specific prompts are designed to guide the LLM to select the best-performing agent in each given trajectory segment τ_j, thus deciding the quality label. Each prompt comprises (1) desc, a concise description of the environment, agents and task, (2)τ_j^{text}, a natural-language summary of τ_j converted from numerical states and actions, and (3) inst, an instruction requesting the identification of the best-performing agent. desc includes heuristic priorities to help the LLM identify typical high-quality behaviors, and inst instructs it to derive a brief analysis before given its decision, so that the priorities are taken into consideration.

E.1 CN

You are an expert cooperative-navigation coach for multi-agent teams. This is a joint trajectory segment with T=8 timesteps and N=3 agents. Important context: among these agents, EXACTLY ONE is following a true expert strategy; the OTHER agents are random policies that at each step pick an action uniformly at random (each action equally likely).

Task: pick EXACTLY ONE agent (0-based index) that you judge to be the most expert-like. IMPORTANT: The last line of your OUTPUT should be STRICTLY and ONLY one index in the format: [k] (k must be between 0 and 2).

Action legend: 0 = no-op 1 = move left 2 = move right 3 = move down 4 = move up

IMPORTANT NOTES (CRITICAL): - Δd = delta of the distance to nearest landmark. * If Δd < 0 => agent moved CLOSER compared with previous step => THIS IS GOOD. * If Δd > 0 => agent moved AWAY compared with previous step => THIS IS BAD. - The following BEHAVIOR FEATURES (in PRIORITY ORDER) can be evidence of an expert-like agent: 1. if the distance to the nearest landmark is already reasonably small, keeps it so small consistently 2. if the distance is large, achieves largest DECREASEMENT of DISTANCE THROUGHOUT TIMESTEPS 3. if the distance is large, chooses a series of purposeful actions that reduce the distance gradually / smoothly 4. doesn't change favorite / nearest landmark frequently 5. avoids colliding with teammates - IF TIES APPEAR, BREAK THEM BY STRICTLY FOLLOWING THE PRIORITY ORDER ABOVE.

Context: team total reward over this segment ≈ {}.

Agent 0 - navigation report. Scorecard: Favorite landmark index (most often nearest): {} (fraction {}) throughout timesteps, DISTANCE DECREASES by {} Collision steps observed: {} Short summary: collisions {}; favorite landmark {}. Timeline: - t=0: action=no-op; Δd={}; nearest landmark id={}; distance to nearest

landmark={}; ... - t=7: action=move right; Δd={}; nearest landmark id={}; distance to nearest landmark={};

Agent 1 - navigation report. Scorecard: Favorite landmark index (most often nearest): {} (fraction {}) throughout timesteps, DISTANCE DECREASES by {} Collision steps observed: {} Short summary: collisions {}; favorite landmark {}. Timeline: - t=0: action=move left; Δd={}; nearest landmark id={}; distance to nearest landmark={}; ... - t=7: action=move right; Δd={}; nearest landmark id={}; distance to nearest landmark={};

Agent 2 - navigation report. Scorecard: Favorite landmark index (most often nearest):{}(fraction {}) throughout timesteps, DISTANCE INCREASES by {} Collision steps observed: {} Short summary: collisions {}; favorite landmark {}. Timeline: - t=0: action=move left; Δd={}; nearest landmark id={}; distance to nearest landmark={}; ... - t=7: action=move down; Δd={}; nearest landmark id={}; distance to nearest landmark={};

Team-level stats: Fraction of valid steps with >1 agents focusing same landmark (overlap): {}

FINAL INSTRUCTION: Derive your analysis in no more than 100 words based on the information above. The last line of your output should be ONLY the index of the most expert-like agent, in the format [k].

Output:

Agent 0 demonstrates the most expert-like behavior with a consistent decrease in distance to the nearest landmark throughout all timesteps ({} in total). It maintained its focus on landmark 1 without collisions. Agent 1 and Agent 2 both showed significant issues: Agent 1 only slightly decreased its distance ({}) and had multiple collisions ({}), while Agent 2 actually increased its distance ({}) and also had {} collisions. Thus, Agent 0 stands out clearly as the best performer.

[0]

E.2 World

Tactical analyst summary for a hunting scenario. This trajectory segment has T=8 timesteps and total_agents (N)=3 (3 adversaries + 3 good).

Task: identify EXACTLY ONE adversary index (from the adversary list above) that is MOST expert-like. Output format constraint: the very last line of your RESPONSE MUST be exactly one index in the form: [k] (k must be one of the adversary indices).

Action legend: 0 = no-op 1 = move left 2 = move right 3 = move down 4 = move up

Key interpretation rule

(Δ d = current_distance - previous_distance):

* Δ d < 0 => moved CLOSER to its nearest target since previous step => POSITIVE evidence of hunting skill. * Δ d > 0 => moved AWAY since previous step => NEGATIVE evidence.

Primary evidence priority (use this order to decide): 1) sustained DECREASE in distance to targets across the segment (large negative overall Δ). 2) high fraction of timesteps with Δ d < 0 (consistent closing). 3) participation in multi-adversary close events (shows tactical coordination). 4) smoothness: smaller Δ d variance suggests purposeful approach rather than jittery/random moves. 5) consistency in preferring the same target (favorite target fraction).

If ties occur, break them by (1) higher priority above, then (2) lower Δ d variance, then (3) higher participation in coordinated events.

Adversary 0 report: Metrics: Avg nearest-good distance: {} Favorite target (good agent): {} (fraction {}) Distance trend across segment: INCREASES by {} (farther) Fraction of steps with distance DECREASE (good steps): {} Avg improvement magnitude when improving: {} Δd standard deviation (smoothness): {} Times within close radius ({}): {} Participation in multi-adversary close events: {} Short timeline (per-step): - t=0: action=move up; Δd={}; nearest_good={}; distance={}; rel={} ... - t=7: action=move right; Δd={}; nearest_good={}; distance={}; rel={}

Adversary 1 report: Metrics: ... Short timeline (per-step): - t=0: action=move down; Δd={}; nearest_good={}; distance={}; rel={} ... - t=7: action=move right; Δd={}; nearest_good={}; distance={}; rel={}

Adversary 2 report: Metrics: ... Short timeline (per-step): - t=0: action=no-op; Δd={}; nearest_good={}; distance={}; rel={} ... - t=7: action=move up; Δd={}; nearest_good={}; distance={}; rel={}

Team-level statistics: Fraction of timesteps with >=2 adversaries simultaneously very close to the same target: {}

FINAL INSTRUCTION: In no more than 100 words, give a brief rationale (objective, tactical) for your choice, then on the FINAL LINE output ONLY the chosen adversary index k, in format [k].

Output:

Adversary 2 is identified as the most expert-like in this hunting scenario due to the significant decrease in average distance to its target, improving by {} overall. It also exhibits a high fraction of {} of timesteps with distance decrease, indicating consistent efforts to close in on its target. Additionally, despite not participating in multi-adversary events, its smoothness with a low Δd standard deviation of {} highlights a purposeful approach. These factors combined demonstrate superior hunting tactics.

[2]

E.3 3m

You are an expert StarCraft II micro-management coach. This is a joint trajectory segment with T=8 timesteps and N=3 friendly agents.

Task: pick EXACTLY ONE agent (0-based index) that you judge to be the most expert-like. IMPORTANT: The last line of your OUTPUT should be STRICTLY and ONLY one index in the format: [k] (k must be between 0 and 2).

Action legend: 0 = no-op (only available for dead agents) 1 = stop 2-5 = move (cardinal directions) >=6 = attack(enemy_id-6)

Primary signals (in order): 1) Timely engagement: attacking when enemies are attackable and causing HP drops. 2) Purposeful positioning: moves that reduce distance to enemies. 3) Team coordination: focusing same enemy as teammates when appropriate. 4) Survivability: staying alive and keeping health/shield.

Tie-break (if needed): higher attack efficiency → earlier first attack → higher remaining health → better team overlap.

Context: team total reward over this segment ≈ {}

Per-agent reports (commentator voice):

Agent 0 - consistent engager. Scorecard: First attack time: {}; Total attacks: {}; Damage-causing attacks (observed HP drops): {}; Attack efficiency (damage-causing / total attacks): {}; Moves that reduced distance to enemies / total moves: {}/{} (effectiveness {}); Steps with enemy visible: {}/8 (≈{}); Team overlap on attacks (fraction): {}; Health: start {} → end {}, min {}. Short summary: Attacked {} times with {} attacks that coincided with observed enemy HP drops; moved closer to enemies on {} moves; saw enemies in {}/{} steps. Timeline (key events; we show actions, important HP changes, visibility & qualitative distances): - t=0: stop; own HP {} ... - t=7: own HP dropped {}→{} DEAD.

Agent 1 - mostly passive. Scorecard: ... Timeline (key events; we show actions, important HP changes, visibility & qualitative distances): - t=0: stop; own HP {} ... - t=7: move; saw enemies (yes); own HP {}

Agent 2 - vulnerable (low HP / died). Scorecard: ... Timeline (key events; we show actions, important HP changes, visibility & qualitative distances): - t=0: move (moved closer to nearest enemy); saw enemies (yes); own HP {} ... - t=7: DEAD.

Decision hints (for your deliberation): - Prefer agents that attack when enemies are attackable and whose attacks coincide with enemy HP drops. - Prefer moves that reduce distance to enemies (purposeful approach). - Prefer agents that contribute to team focus (overlap) rather than lone, ineffective attacks.

FINAL INSTRUCTION: Derive your analysis in no more than 100 words based on the information above. The last line of your output should be ONLY the index of the most expert-like agent, in the format [k] without any other words.

Output:

Agent 0 stands out as the most expert-like, demonstrating consistent engagement by attacking three times, causing HP drops on two of those occasions. They effectively closed the distance to enemies on every move and displayed situational awareness by seeing enemies in {} out of 8 steps. Although Agent 0 ultimately died, their contribution through damage dealt and timely actions shows superior skill compared to Agents 1 and 2, who exhibited no attacks and less proactive positioning.

[0]

E.4 Zerg_3v4

You are an expert StarCraft II micro-management coach analyzing a Zerg vs Zerg battle. This is a joint trajectory segment with T=8 timesteps and N=3 friendly Zerg agents. Environment: SMAC-v2 10gen_zerg (3 friendly Zerg units vs 4 enemy Zerg units).

Task: pick EXACTLY ONE agent (0-based index) that you judge to be the most expert-like. IMPORTANT: The last line of your OUTPUT should be STRICTLY and ONLY one index in the format: [k] (k must be between 0 and 2).

Zerg Unit Types: - Zergling: Fast melee attacker, low health, high damage up close - Hydralisk: Ranged attacker, medium health, good dps - Baneling: Suicide unit, explodes for area damage

Action legend: 0 = no-op (only available for dead agents) 1 = stop 2-5 = move (cardinal directions) >=6 = attack(enemy_id-6) (targets enemy 0-3)

Primary signals (in order): 1) Unit-type appropriate tactics: Zerglings close distance, Hydralisks maintain range, Banelings seek multi-kills 2) Timely engagement: attacking when enemies are attackable and causing HP drops 3) Target priority: focusing vulnerable enemies or high-value targets first 4) Survivability: avoiding unnecessary damage while dealing damage 5) Team coordination: focusing same enemy as teammates when appropriate Tie-break (if needed): higher attack efficiency → better target selection → higher remaining health → earlier first attack.

Context: team total reward over this segment ≈ {}.

Per-agent reports:

Agent 0 (Hydralisk) - Hydralisk. Metrics: Unit type: Hydralisk; First attack time: never; Total attacks: {}; Damage-causing attacks: {}; Attack efficiency: {}; Movement effectiveness: {}; Enemy visibility: {}/{} ({}); Team coordination: {}; Health: {}→{} (min {}); Most focused on enemy_{} ({} attacks). Summary: Hydralisk with {} attacks ({} caused damage); {} purposeful moves; saw enemies {}% of time. Timeline: - t=0: move; saw enemies (yes); own HP {} ... - t=7: move; saw enemies (yes); own HP {}

Agent 1 (Hydralisk) - Hydralisk. Metrics: ... Timeline: - t=0: move (moved away from nearest enemy); saw enemies (yes); own HP {} ... - t=7: move; saw enemies (yes); own HP {}

Agent 2 (Hydralisk) - Hydralisk. Metrics: ... Timeline: - t=0: move; saw enemies (yes); own HP {} ... - t=7: own HP dropped {}→{} DEAD.

Decision guidance (Zerg-specific): - Zerglings: Should aggressively close distance and engage - Hydralisks: Should maintain optimal range and deal consistent damage - Banelings: Should seek high-value multi-kills or finish key targets - All: Should coordinate focus fire when advantageous

FINAL: Provide brief analysis (≤100 words). Then the last line of your OUTPUT should ONLY be the index k of the agent chosen as expert, written in the form [k] .

Output:

In the current Zerg vs Zerg scenario, Agent 2 (Hydralisk) demonstrates the most competent behavior by engaging an enemy with an attack, even though it resulted in its demise. Despite being ineffective in damage caused, the decision to attack shows an understanding of unit interaction, which is critical in micro-management context. Agents 0 and 1 failed to engage despite having opportunities, showcasing poor tactical execution overall. Therefore, based on the provided metrics, Agent 2 exemplifies the most expert-like play in this situation.

[2]

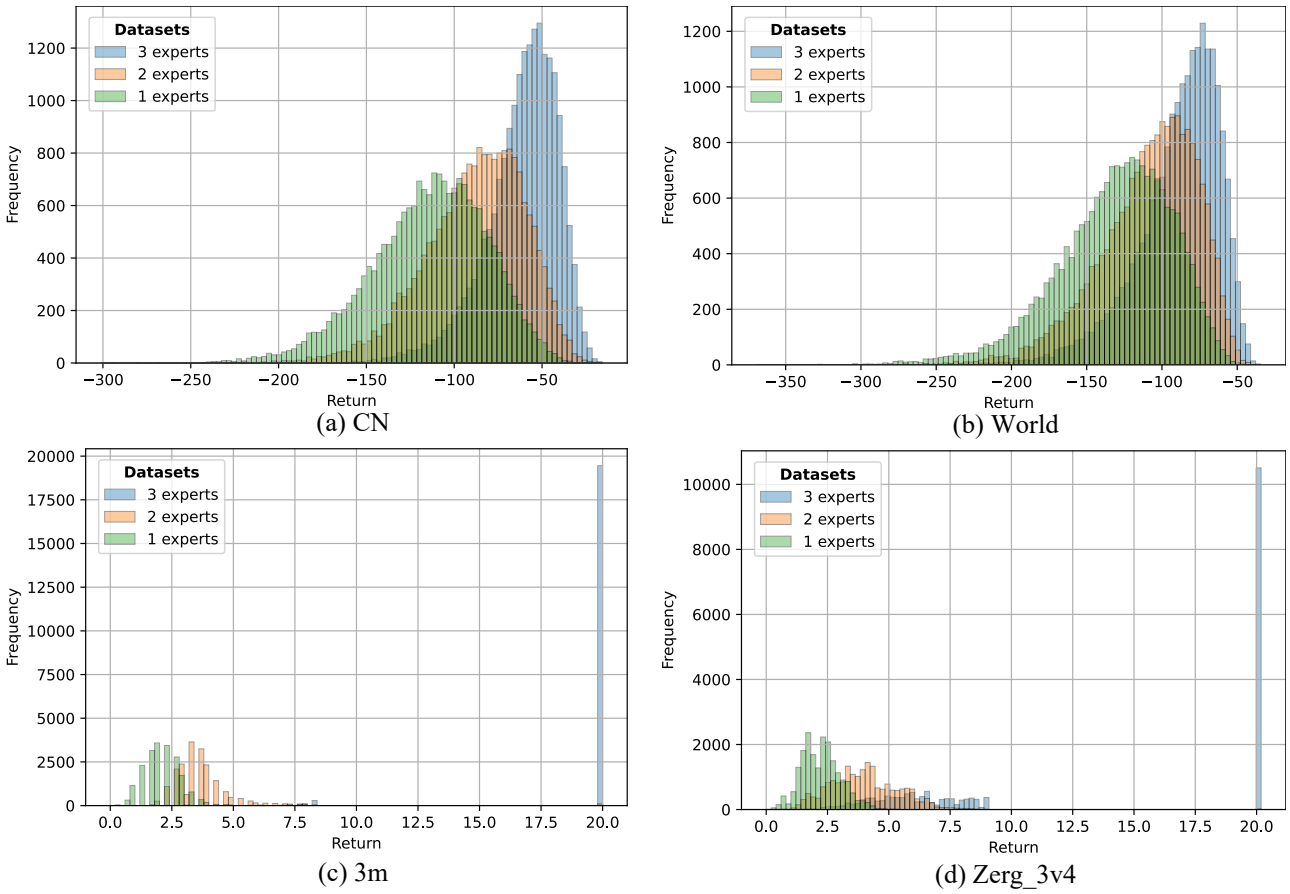


Figure 2: Return distributions of the original datasets used in the main experiments. For each environment, we show collected datasets containing exactly $n = 1, 2, 3$ expert agents (green, orange, and blue respectively), each comprising 20k trajectories. Datasets with $n = 3$ expert agents are used for evaluation. As expected, the average return increases with the number of experts. The average returns of the $n = 1$ and $n = 3$ datasets correspond to R_{lo} and R_{hi} used for normalization in the main experiments.

F MORE EXPERIMENT RESULTS

In this section, we report the return distributions of the datasets and the learning curves corresponding to the main results.

F.1 Return Distribution of Datasets

As shown in Figure 2, we present the return distributions of the original datasets used in the four main experimental tasks. For each task, we collect datasets containing exactly $n = 1, 2, 3$ expert agents (represented by green, orange, and blue, respectively), with each dataset comprising 20k trajectories. Datasets with $n = 3$ expert agents are used for evaluation. As anticipated, the average return of the dataset increases with the number of expert agents. The average returns of the $n = 1$ and $n = 3$ datasets correspond precisely to the R_{lo} and R_{hi} values used as references for normalization in the main experiments. Regarding the datasets generated by the data augmentation algorithms, we observed that the return distributions of the datasets generated by diffusion-based methods are very similar and largely indistinguishable. The mean returns of these generated datasets all closely align with the specified target return values. To conserve space, we choose not to visualize them here. However, as emphasized in the main text, although the baseline methods generate datasets with return distributions similar to CODI, CODI achieves these returns by generating trajectories that exhibit greater improvement from more challenging initial states, which accounts for its performance superiority.

F.2 Learning Curves of Main Results

As shown in Figure 3, we present the complete learning curves corresponding to the results in the main experimental table. The x-axis represents the training iterations of the offline MARL algorithms, and the y-axis shows the unnormalized episode returns. The final values of these curves, after normalization, yield the results reported in the main experimental table. For clarity, the curves are smoothed, which may cause minor visual discrepancies in performance comparisons between methods. It can be observed that across most dataset and offline MARL algorithm combinations, CODI (red star) and CODI $_{\text{w Pri.}}$ (purple circle) achieve the most stable and superior performance, demonstrating the effectiveness of our proposed method.

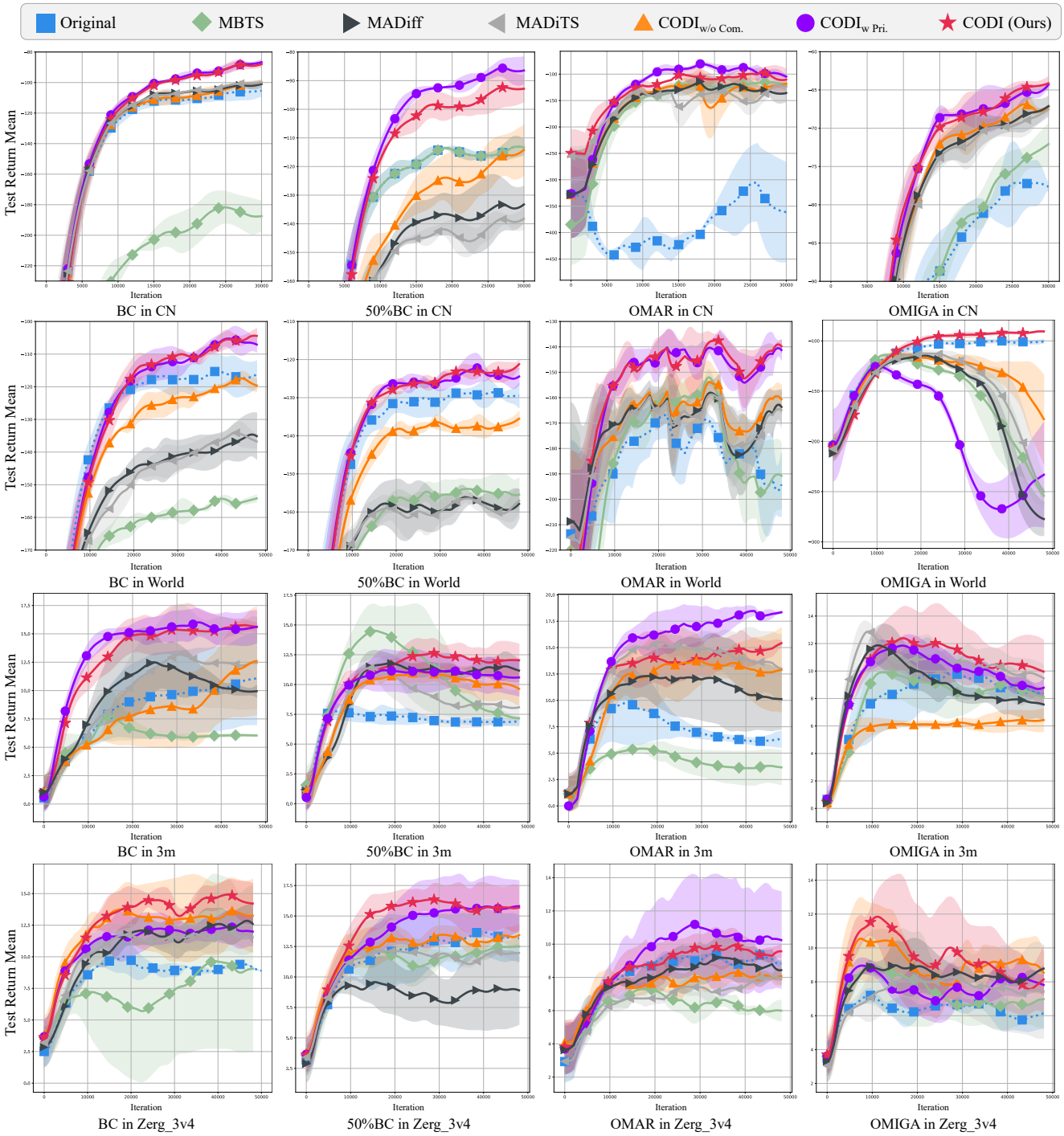


Figure 3: Complete learning curves corresponding to the main experimental results. The x-axis represents the training steps of the offline MARL algorithms, and the y-axis shows the unnormalized episode returns. The final values of these curves, when normalized, yield the results presented in Table 1 of the main text. For clarity, the curves are smoothed. Across most dataset and offline MARL algorithm combinations, CODI and CODI $_{\text{w Pri.}}$ achieve the most stable and superior performance, demonstrating the effectiveness of our approach.

REFERENCES

[1] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Nicolaus Foerster, and Shimon Whiteson. 2023. SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning. In *NeurIPS Datasets and Benchmarks Track*.

[2] Charles A Hepburn and Giovanni Montana. 2022. Model-based Trajectory Stitching for Improved Offline Reinforcement Learning. In *Offline RL Workshop at Neural Information Processing Systems*.

[3] Diederik P. Kingma and Jimmy Ba. 2025. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, Yoshua Bengio and Yann LeCun (Eds.).

[4] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*. 6379–6390.

[5] Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. 2022. Plan Better Amid Conservatism: Offline Multi-Agent Reinforcement Learning with Actor Rectification. In *International Conference on Machine Learning*. 17221–17237.

[6] Dean A Pomerleau. 1988. Alvinn: An Autonomous Land Cehicle in a Neural Network. In *Advances in Neural Information Processing Systems*. 305–313.

[7] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *AAMAS*. 2186–2188.

[8] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *International Conference on Machine Learning*. 3319–3328.

[9] Xiangsen Wang, Haoran Xu, Yinan Zheng, and Xianyuan Zhan. 2023. Offline Multi-Agent Reinforcement Learning with Implicit Global-to-Local Value Regularization. In *Advances in Neural Information Processing Systems*, Vol. 36. 52413–52429.

[10] Lei Yuan, Yuqi Bian, Lihe Li, Ziqian Zhang, Cong Guan, and Yang Yu. 2025. Efficient Multi-agent Offline Coordination via Diffusion-based Trajectory Stitching. In *International Conference on Learning Representations*.

[11] Zhilong Zhang, Yihao Sun, Junyin Ye, Tianshuo Liu, Jiaji Zhang, and Yang Yu. 2024. Flow to Better: Offline Preference-based Reinforcement Learning via Preferred Trajectory Generation. In *International Conference on Learning Representations*.

[12] Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. 2024. MADiff: Offline Multi-agent Learning with Diffusion Models. In *Advances in Neural Information Processing Systems*.