A Related Works

A.1 Imitation Learning and Multi-agent Imitation Learning

Imitation learning (IL) recovers expert policies from demonstrations without requiring reward function information. IL methods fall into two main categories: Behavior Cloning (BC) and Inverse Reinforcement Learning (IRL) (Ng & Russell, 2000; Abbeel & Ng, 2004). BC aims to mimic the expert by maximizing state-action pair likelihoods in demonstrations. However, BC often faces covariate shift issues due to not considering policy distribution in the environment (Ross & Bagnell, 2010; Ross et al., 2011). In contrast, IRL first recovers a reward function, and then optimizes the policy using reinforcement learning, not supervised learning like BC. Recent methods, based on adversarial frameworks, excel in data efficiency compared to traditional IRL approaches (Ho & Ermon, 2016; Finn et al., 2016; Fu et al., 2017).

In multi-agent scenarios, Le et al. (2017) develop a latent coordination model for players in cooperative games with distinct roles. However, many multi-agent scenarios involve agents who need cooperation or specific roles (e.g., self-driving). Bhattacharyya et al. (2018) extend generative adversarial imitation learning for multi-agent problems using parameter sharing but do not model agent interactions. Interaction Network (Battaglia et al., 2016) simulates physical object relations with binary links, while CommNet (Sukhbaatar et al., 2016) learns dynamic agent communication but does not explicitly characterize agent action dependencies. Some researchers propose using graph techniques or attention mechanisms to infer multi-agent relationships. For instance, Kipf et al. (2018) use graph neural networks (GNN) to infer agent relationship types. Hoshen (2017) introduce attention mechanisms into multi-agent predictive modeling.

Additionally, Li et al. (2020) combine generative models and attention mechanisms to capture the behavior generation process of multi-agent systems. These works address agent relationship reasoning rather than explicitly capturing agent action dependencies. Most existing works primarily focus on agent behavior prediction but offer limited insights into agent behavior interdependence.

A.2 Distribution Matching in Multi-agent Imitation Learning

Ho & Ermon (2016) originally introduced the concept of adversarial distribution matching as a methodology for executing imitation learning within the single-agent framework (the GAIL algorithm). Building upon this foundation, Song et al. (2018) extended the applicability of GAIL to the multi-agent environment, albeit with specific nuanced adaptations. Their analytical framework posits independent imitation learning as an endeavor to seek a Nash equilibrium, assuming a unique equilibrium point. In pursuing this line of inquiry, MAGAIL harnesses recent advancements in single-agent GAIL convergence theory, as elucidated by Guan et al. (2021), to elucidate the process of convergence towards the joint expert policy. Furthermore, Wang et al. (2021) undertake a comprehensive exploration of Multi-Agent Reinforcement Learning (MARL) by employing copula functions to explicitly capture the interdependence between the marginal policies of individual agents in the context of multi-agent imitation learning. Additionally, a pair of complementary studies by Durugkar et al. (2020) and Radke et al. (2022) establish the efficacy of balancing individual preferences, such as aligning with the state-action visitation distribution of specific strategies, in conjunction with the shared task reward. These approaches can accelerate progress on the shared task. Wang et al. (2023) view multi-agent imitation learning as a decentralized distribution matching problem and combine the distribution matching reward with the task reward. In contrast to the aforementioned body of research, the primary objective of the present paper diverges from the conventional study of imitation learning. Instead, we focus enhancing cooperative task performance through distribution matching techniques that model inter-agent dependencies.

B Proofs of the Theoretical Results

B.1 Proof of Lemma 2

Lemma 2. *(Multi-agent Advantage Decomposition (Kuba et al., 2021)). Let $i^{1:N}$ be a permutation of N agents. For any state $\mathbf{s} \in \mathcal{S}$ and joint actions $\mathbf{a} = \mathbf{a}^{1:N} \in \mathcal{A}$, the following equation*

holds for any subset of N agents and any permutation of their labels: $\mathbf{A}_{\pi_\theta}^{k+1:N}(s, \mathbf{a}^{1:k}, \mathbf{a}^{k+1:N}) = \sum_{j=k+1}^{N} A_{\pi_\theta}^{j}(\mathbf{s}, \mathbf{a}^{1:j-1}, a^j)$*, where* $k = 0, \ldots, N-1$.

Proof. By the definition of the multi-agent advantage, we have:

$$\begin{aligned}&\mathbf{A}_{\pi_\theta}^{k+1,\ldots,N}(\mathbf{s}, \mathbf{a}^{1:k}, \mathbf{a}^{k+1:N}) \\ =&\mathbf{Q}_{\pi_\theta}^{1:k,k+1:N}(\mathbf{s}, \mathbf{a}^{1:k,k+1:N}) - \mathbf{Q}_{\pi_\theta}^{1:k}(\mathbf{s}, \mathbf{a}^{1:k})\end{aligned} \tag{7}$$

which can be written as a telescoping sum:

$$\begin{aligned}&\mathbf{Q}_{\pi_\theta}^{1:k,k+1:N}(\mathbf{s}, \mathbf{a}^{1:k,k+1:N}) - \mathbf{Q}_{\pi_\theta}^{1:k}(\mathbf{s}, \mathbf{a}^{1:k}) \\ =& \sum_{j=k+1}^{N} \left[\mathbf{Q}_{\pi_\theta}^{1:j}(\mathbf{s}, \mathbf{a}^{1:j}) - \mathbf{Q}_{\pi_\theta}^{1:j-1}(\mathbf{s}, \mathbf{a}^{1:j-1})\right] \\ =& \sum_{j=k+1}^{N} A_{\pi_\theta}^{j}(\mathbf{s}, \mathbf{a}^{1:j-1}, a^j)\end{aligned} \tag{8}$$

So the Multi-agent Advantage Decomposition lemma is proved. ■

B.2 Proofs of Theorems 1 and 3

Theorem 1. *Let R_ω^{cdr} be the cumulative discounted individual reward given by the discriminator parameterized by w, and let V_ϕ be the joint value function parameterized by ϕ. Let $\mathcal{D}^\pi$ denote the total data collected by π. Then increasing the joint advantage variance $\sum_{k=1}^{N} \mathrm{Var}[A_{\pi_\theta}^{k}(\mathbf{s}, a^k)]$ of multi-agent policy gradients is equivalent to solving a bi-level joint value loss optimization problem that is related to the joint reward function and the joint value function:*

$$\begin{aligned}&\max_{w} \sum_{k=1}^{N} \mathbb{E}_{\mathbf{s}, a^k \in \mathcal{D}^{\pi^k}} [R_\omega^{cdr}(\mathbf{s}, a^k) - V_{\phi^*}(\mathbf{s})]^2, \\ &\text{s. t. } \phi^*(\mathbf{s}) = \min_{\phi} \sum_{k=1}^{N} \mathbb{E}_{\mathbf{s}, a^k \in \mathcal{D}^{\pi_k}} [R_\omega^{cdr}(\mathbf{s}, a^k) - V_{\phi}(\mathbf{s})]^2,\end{aligned} \tag{9}$$

where $R_\omega^{cdr}(\mathbf{s}, a^k) = -\sum_{t=l}^{T} \gamma^{t-l}(\log \sigma(1 - D_\omega(\mathbf{s}_t, a_t^k)))$ denotes the individual reward function for the agent i^k, and σ be activation function (generally Sigmoid*).*

Proof. To prove it, we need to establish the connection between the sum of individual value loss $\sum_{k=1}^{N} \mathbb{E}_{\mathbf{s}, a^k \in \mathcal{D}^{\pi^k}} [R_\omega^{cdr}(\mathbf{s}, a^k) - V_{\phi^*}(\mathbf{s})]^2$ and the individual advantage function $A_{\pi_\theta}^{k}(\mathbf{s}, a^k)$. Let $\mathcal{D}_t^\pi$ denote the data collected starting at time step t. Following (Zhang et al., 2022), we have:

$$\begin{aligned}&\sum_{k=1}^{N} \mathbb{E}_{\mathbf{s}, a^k \in \mathcal{D}^{\pi^k}} [R_\omega^{cdr}(\mathbf{s}, a^k) - V_{\phi}(\mathbf{s})]^2 \\ =& N \mathop{\mathbb{E}}_{\mathbf{s}, a^k \in \mathcal{D}^{\pi_k}} (V_\pi(\mathbf{s}) - V_\phi(\mathbf{s}))^2 + \sum_{t=0}^{\infty} \gamma^{2t} \frac{|\mathcal{D}_t^\pi|}{|\mathcal{D}^\pi|} \sum_{k=1}^{N} \mathop{\mathrm{Var}}_{\mathbf{s}, a^k \in \mathcal{D}_t^{\pi^k}} [A_{\pi_\theta}^{k}(\mathbf{s}, a^k)].\end{aligned} \tag{10}$$

Then we look for ϕ^* to minimize the loss of joint value. Ideally, ϕ^* would keep the difference between $V_\pi(\mathbf{s})$ and $V_{\phi^*}(\mathbf{s})$ at a small level, then we have:

$$\begin{aligned}
&\min_{\phi} \sum_{k=1}^{N} \mathbb{E}_{\mathbf{s},a^k\in\mathcal{D}^{\pi^k}}[R_\omega^{cdr}(\mathbf{s},a^k) - V_\phi(\mathbf{s})]^2 \\
=&\min_{\phi} \sum_{k=1}^{N} \mathbb{E}_{\mathbf{s},a^k\in\mathcal{D}^{\pi^k}}[R_\omega^{cdr}(\mathbf{s},a^k) - V_{\phi^*}(\mathbf{s})]^2 \\
=&N \mathop{\mathbb{E}}_{\mathbf{s},a^k\in\mathcal{D}^{\pi^k}} (V_\pi(\mathbf{s}) - V_\phi(\mathbf{s}))^2 + \sum_{t=0}^{\infty}\gamma^{2t}\frac{|\mathcal{D}_t^\pi|}{|\mathcal{D}^\pi|}\sum_{k=1}^{N}\mathop{\mathrm{Var}}_{\mathbf{s},a\in\mathcal{D}_t^{\pi^k}}[A_{\pi_\theta}^k(\mathbf{s},a^k)] \\
\approx&\sum_{t=0}^{\infty}\gamma^{2t}\frac{|\mathcal{D}_t^\pi|}{|\mathcal{D}^\pi|}\sum_{k=1}^{N}\mathop{\mathrm{Var}}_{\mathbf{s},a\in\mathcal{D}_t^{\pi^k}}[A_{\pi_\theta}^k(\mathbf{s},a^k)].
\end{aligned} \tag{11}$$

Then in order to increase the joint advantage variance, we should add the maximization operator on both sides of the equation:

$$\begin{aligned}
&\max_{w} \sum_{k=1}^{N} \mathbb{E}_{\mathbf{s},a^k\in\mathcal{D}^{\pi^k}}[R_\omega^{cdr}(\mathbf{s},a^k) - V_{\phi^*}(\mathbf{s})]^2 \\
\approx&\max_{w}\sum_{t=0}^{\infty}\gamma^{2t}\frac{|\mathcal{D}_t^\pi|}{|\mathcal{D}^\pi|}\sum_{k=1}^{N}\mathop{\mathrm{Var}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}[A_{\pi_\theta}^k(\mathbf{s},a^k)].
\end{aligned} \tag{12}$$

So the original proposition that increasing the joint advantage variance is equivalent to solving a bi-level joint value loss optimization problem is proved. ■

Theorem 3. *Let $A_{\pi_\theta}^k(\mathbf{s},\mathbf{a}^{-k},a^k)$ be the advantage function of agent i_k with global dependency-enhanced discriminator, and let $\mathbf{A}_{\pi_\theta}(\mathbf{s},\mathbf{a})$ be the joint advantage function of all agents in independent framework, global dependency-enhanced discriminator framework has a more significant advantage variance compared to the independent framework:*

$$\sum_{k=1}^{N}\mathop{\mathrm{Var}}_{\substack{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}\\ \mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}}[A_{\pi_\theta}^k(\mathbf{s},\mathbf{a}^{-k},a^k)] \geq \sum_{k=1}^{N}\mathop{\mathrm{Var}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi}}[A_{\pi_\theta}^k(\mathbf{s},a^k)] \tag{13}$$

Proof. According to Lemma. (2), we take an arbitrary k, cause $\mathop{\mathbb{E}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[A_{\pi_\theta}^k(\mathbf{s},a^k)\right] = 0$, then following (Kuba et al., 2021) we have:

$$\begin{aligned}
&\mathop{\mathrm{Var}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi}}[A_{\pi_\theta}^k(\mathbf{s},a^k)] = \mathop{\mathbb{E}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[A_{\pi_\theta}^k(\mathbf{s},a^k)^2\right] \\
=&\mathop{\mathbb{E}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}}\left[\mathbf{A}_{\pi_\theta}^{1:N}(\mathbf{s},\mathbf{a}^{1:N})\right]^2\right] \\
\leq&\mathop{\mathbb{E}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}}\left[\mathbf{A}_{\pi_\theta}^{1:N}(\mathbf{s},\mathbf{a}^{1:N})^2\right]\right] \\
=&\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}}\left[\mathop{\mathbb{E}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[\mathbf{A}_{\pi_\theta}^{1:N}(\mathbf{s},\mathbf{a}^{1:N})^2\right]\right]
\end{aligned} \tag{14}$$

The above can be equivalently, but more tellingly, rewritten after permuting (cyclic shift) the labels of agents, in the following way

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}}\left[\mathop{\mathbb{E}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[\mathbf{A}_{\pi_\theta}^{1:N}(\mathbf{s},\mathbf{a}^{1:N})^2\right]\right]\\
=&\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}}\left[\mathop{\mathbb{E}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[\mathbf{A}_{\pi_\theta}^{1:k-1,k+1:N,k}(\mathbf{s},\mathbf{a}^{1:k-1,k+1:N,k})^2\right]\right]\\
=&\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}}\left[\mathop{\mathrm{Var}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[\mathbf{A}_{\pi_\theta}^{1:k-1,k+1:N,k}(\mathbf{s},\mathbf{a}^{1:k-1,k+1:N,k})\right]\right],
\end{aligned}\tag{15}$$

which, by the Lemma. 2, equals

$$\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}}\left[\mathop{\mathrm{Var}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[A_{\pi_\theta}^k(\mathbf{s},\mathbf{a}^{-k},a^k)\right]\right].\tag{16}$$

The equation above can be further simplified by

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}}\left[\mathop{\mathrm{Var}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[A_{\pi_\theta}^k(\mathbf{s},\mathbf{a}^{-k},a^k)\right]\right]\\
=&\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}}\left[\mathop{\mathbb{E}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}\left[A_{\pi_\theta}^k(\mathbf{s},\mathbf{a}^{-k},a^k)^2\right]\right]\\
=&\mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}\in\mathcal{D}_t^{\pi}}\left[A_{\pi_\theta}^k(\mathbf{s},\mathbf{a}^{-k},a^k)^2\right]=\mathop{\mathrm{Var}}_{\mathbf{s},\mathbf{a}\in\mathcal{D}_t^{\pi}}\left[A_{\pi_\theta}^k(\mathbf{s},\mathbf{a}^{-k},a^k)^2\right].
\end{aligned}$$

Then we have

$$\mathop{\mathrm{Var}}_{\substack{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}\\ \mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}}[A_{\pi_\theta}^k(\mathbf{s},\mathbf{a}^{-k},a^k)]\geq\mathop{\mathrm{Var}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi}}[A_{\pi_\theta}^k(\mathbf{s},a^k)],\tag{17}$$

and We sum both sides of the Eq. (17) from $k=1$ to N, we have $\sum_{k=1}^N\mathop{\mathrm{Var}}_{\substack{\mathbf{s},\mathbf{a}^{-k}\in\mathcal{D}_t^{\pi^{-k}}\\ \mathbf{s},a^k\in\mathcal{D}_t^{\pi^k}}}[A_{\pi_\theta}^k(\mathbf{s},\mathbf{a}^{-k},a^k)]\geq\sum_{k=1}^N\mathop{\mathrm{Var}}_{\mathbf{s},a^k\in\mathcal{D}_t^{\pi}}[A_{\pi_\theta}^k(\mathbf{s},a^k)]$. The proof is finished. ■

C Methodology Details

C.1 Details of Regularization for Lipschitz continuity condition

Lipschitz continuity is essential when employing the Wasserstein loss in various GAN settings (Arjovsky et al., 2017; Xiao et al., 2019). However, recent research (Kim et al., 2021) has demonstrated that the Lipschitz constant of the standard dot product self-attention layer can be unbounded, thereby violating the Lipschitz continuity condition in transformer-based discriminators. We employ two regularization techniques to ensure the Lipschitz continuity condition of our designed discriminator (Lee et al., 2022). Firstly, we adopt L2 attention regularization as proposed in (Kim et al., 2021). It replaces dot product similarity with Euclidean distance and establishes a linkage between the weights of the projection matrices used for querying and self-attention as $\mathrm{Attention}_h(\mathbf{X})=\mathrm{Softmax}(\frac{d(\mathbf{X}\mathbf{W}_1,\mathbf{X}\mathbf{W}_k)}{\sqrt{d_h}})\mathbf{X}\mathbf{W}_v$, where $\mathbf{W}_q=\mathbf{W}_k$, $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$ are the projection matrices for query, key, and value, respectively. $d(\cdot,\cdot)$ computes vectorized L2 distances between two sets of points. $\sqrt{d_h}$ is the feature dimension for each head. Secondly, we incorporate Spectral Normalization (SN) during discriminator training to further bolster Lipschitz continuity (Miyato et al., 2018). Given that Transformer blocks are sensitive to the Lipschitz constant, and a low Lipschitz constant for MLP blocks can cause the Transformer's output to collapse into a rank-1 matrix (Dong et al., 2021; Lee et al., 2022), we suggest augmenting the spectral norm of the projection matrices to address this concern. Specifically, inspired by (Lee et al., 2022), we multiply the normalized weight matrices of each layer by the spectral norm during initialization and update them according to the following rule for spectral normalization, wherein the standard spectral norm of the weight matrices is computed as $\bar{W}_{SN}(\mathbf{W}):=\sigma(\mathbf{W}_{init})\cdot\mathbf{W}/\sigma(\mathbf{W})$.

Algorithm 1: Multi-agent Imitation Learning via Global Dependency-enhanced Distribution Matching ($\mathbf{MILD}^2$)

Input: Initial parameters of policies θ_0, discriminator ω_0, and value estimator ϕ_0; expert trajectories $\mathcal{D} = \{(\mathbf{s}, a^k)\}_{k=1}^N$; batch size B; Markov game as a black box $(N, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^N, \mathcal{T}, \{\mathcal{R}_i\}_{i=1}^N, \gamma)$.
Output: Learned policies π_θ and reward functions D_ω.

for $u = 0, 1, 2, \ldots$ **do**
  Obtain trajectories of size B from π by the process: $\mathbf{s}_0 \sim \mathcal{S}, \mathbf{a}_t \sim \pi_{\theta_u}(\mathbf{a}_t|\mathbf{s}_t), \mathbf{s}_{t+1} \sim \mathcal{T}(\mathbf{s}_t|\mathbf{a}_t)$.
  Sample state-action pairs from $\mathcal{D}$ with batch size B.
  Denote state-action pairs from π and $\mathcal{D}$ as $\mathcal{D}_\pi$ and $\mathcal{D}_E$.
  Update ω to minimize the objective Eq. (6).
  for $k = 1, \ldots, N$ **do**
    Compute value estimate V^k and advantage estimate A^k for $(\mathbf{s}, \mathbf{a^{1:k}}) \in \mathcal{S}_\pi$.
    Update ϕ^k to decrease the objective Eq. (18).
    Update θ^k by policy gradient with small step sizes as Eq. (19).
  end for
end for
return policy model π_θ and reward model D_ω.

C.2 Details of sequential autoregressive modeling generator

Similar to the designed sequential autoregressive discriminator, we also need to consider the problem of modeling complex dependencies among multiple intelligent agents for the generator (policy model). Agents commonly collaborate in various multi-agent cooperative tasks to accomplish a shared objective. Consequently, the efficacy of an agent's actions can be influenced by the actions of teammates, as well as have an impact on their behavior. Neglecting the actions of other agents can result in suboptimal value assessment and hinder effective collaboration. Diverging from existing mechanisms that match individual policies in multi-agent imitation learning, our objective is to construct a generator that incorporates multi-agent interactions to facilitate joint policy matching among intelligent agents. To this end, we propose employing the MAT (Wen et al., 2022) model as the generator, which adeptly models sequences of actions performed by multiple agents in an autoregressive fashion, thereby generating a joint policy distribution for the intelligent agents.

Hence, the generator also adopts "encoder-decoder" architecture, consisting of an encoder that learns representations of the joint observations, and a decoder that outputs actions for each agent in an autoregressive manner. The encoder of the generator exhibits a similar architecture to that of the discriminator, with the distinction that it lacks any regularization mechanisms and includes an additional MLP layer for estimating the individual state values. During the training phase, our objective is to approximate the value function using the encoder and minimize the empirical Bellman error, which can be achieved through the equation:

$$\mathcal{L}_V(\phi) = \frac{1}{T_n} \sum_{k=1}^{N} \sum_{t=0}^{T-1} \left[r_t^k(\mathbf{s}_t, \mathbf{a}_t^{-k}, a_t^k) + \gamma V_{\bar{\phi}}(\hat{o}_{t+1}^k) - V_\phi(\hat{o}_t^k) \right]^2, \tag{18}$$

where $\bar{\phi}$ is the target network's parameter, which is non-differentiable and updated every few epochs (Wen et al., 2022). In contrast, the decoder of the generator captures the dependencies among the actions of multiple intelligent agents by employing a masked self-attention mechanism. It further incorporates a cross-attention layer to merge the encoder's hidden state representation with the intelligent agents' action representation. The first input to the decoder is a symbol denoting the initiation of the decoding process, enabling the generation of the individual action for the first agent. Subsequently, the decoder sequentially produces the complete joint action of the multiple intelligent agents in an autoregressive manner. Following (Wen et al., 2022), to train the decoder, we minimize the following clipping PPO objective of

$$\mathcal{L}_\pi(\theta) = \frac{1}{T_n} \sum_{k=1}^{N} \sum_{t=0}^{T-1} \min(r_t^k(\theta) \hat{A}_t^m, \text{clip}(r_t^k(\theta), 1 \pm \epsilon) \hat{A}_t^k), \tag{19}$$

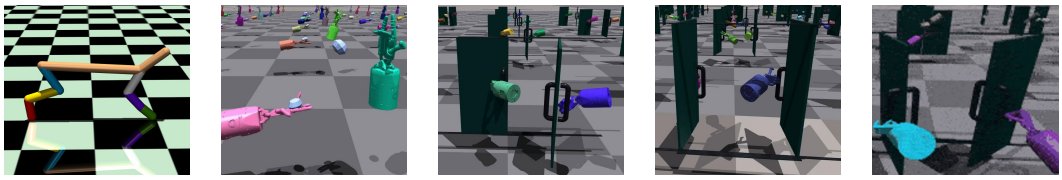

Figure 6: Demonstrations of the Bi-DexHands and the HalfCheetah environments.

Figure 7: Demonstrations of the Google Football environments.

where $r_t^k(\theta) = \frac{\pi_\theta^k(a_t^k|\mathbf{s}_t, \hat{\mathbf{a}}_t^{1:k-1})}{\pi_{\theta_{old}}^k(a_t^k|\mathbf{s}_t, \hat{\mathbf{a}}_t^{1:k-1})}$, and $\hat{A}_t^k(\mathbf{s}_t, \mathbf{a}_t^{1:k-1}) = r_t^k(\mathbf{s}_t, \mathbf{a}_t^{-k}, a_t^k) + \gamma^N V_\phi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - V_\phi(\mathbf{s}_t, \mathbf{a}_t)$ is an estimate of the individual advantage function.

The proposed framework's training pipeline follows classical adversarial generative imitation learning (Song et al., 2018; Wang et al., 2023), i.e., alternate training for the generator and discriminator according to Eq. (19), Eq. (18), and Eq. (6). We include the $\mathbf{MILD}^2$algorithm as Alg. (1).

D Experiment Details

D.1 Experimental Setup

Benchmark datasets. We evaluated $\mathbf{MILD}^2$ using four benchmarks: StarCraftII Multi-Agent Challenge (SMAC) benchmark (Samvelyan et al., 2019), Google Research Football benchmark (Football) (Kurach et al., 2020), Bimanual Dexterous Hands Manipulation benchmark (Bi-DexHands) (Chen et al., 2022), and Multi-agent MuJoCo benchmark (Ma-Mujoco) (de Witt et al., 2020). We constructed several multi-agent offline datasets on these benchmarks by collecting 10,000 (for tasks in Bi-DexHands) and 100,000 (for tasks in others) transitions of expert policy from HAPPO (Kuba et al., 2022).

- **SMAC**. We constructed an offline dataset using data from the game "StarCraft II" on four maps with discrete action space, each with varying difficulty settings. All maps employ an identical reward function, and the dataset for each map comprises 100,000 transitions.
- **Football**. This benchmark encompasses a series of discrete control tasks in a football game that require cooperation. Our approach was evaluated using data consisting of an average of 100,000 transitions.
- **Bi-DexHands**. This benchmark offers a set of challenging bimanual manipulation tasks: continuous control tasks involving the control of two 24-DoF robotic hands to mimic human behavior. We constructed an offline dataset for three tasks and evaluated our method using 10,000 transitions.
- **Ma-Mujoco**. This benchmark comprises a collection of continuous control tasks in machine learning. In each Ma-MuJoCo environment, each agent controls a specific part of a shared robot (e.g., a leg of a Hopper), and all agents aim to maximize a shared reward function. Our method was evaluated using 100,000 transitions.

We adopt the open-source implementations for these Benchmarks. Fig. (6) and Fig. (7) illustrate tasks from these benchmarks. The *HalfCheetah 6×1* task is shown in Fig. (6a) while Fig. (6b-e) illustrate the *CatchOver2Underarm*, *DoorOpenInward*, *DoorCloseOutward* and *DoorCloseInward* environments from the Bi-DexHands benchmark. Fig. (7a-c) illustrate the *3 vs 1 with Keeper*, *Easy*

Table 2: Specs of tested tasks (maps) in the SMAC benchmark.

Name	Agents	Enemies	Type
3m	3 Marines	3 Marines	homogeneous & symmetric
3s5z	3 Stalkers and 5 Zealots	3 Stalkers and 5 Zealots	heterogeneous & asymmetric
6h vs 8z	6 Hydralisks	8 Zealots	micro-trick: focus fire
MMM2	1 Medivac, 2 Marauders & 7 Marines	1 Medivac, 3 Marauders & 8 Marines	heterogeneous & asymmetric

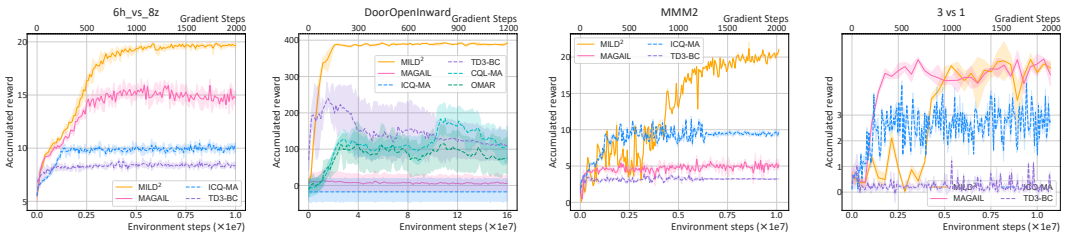

Figure 8: The learning curve comparisons on several complex and challenging tasks from SMAC, Football, and Bi-DexHands benchmarks. $\mathbf{MILD}^2$ and MAGAIL use environmental steps as the horizontal axis, and other non-GAIL methods use gradient steps.

Counter-attack, and *Pass and Shoot* environments from the Football benchmark. Tested maps in the StarCraft II micromanagement benchmark are summarized in Tab. (2).

Baselines. We compare our method against five classical multi-agent offline RL methods, including MAGAIL (Song et al., 2018), the multi-agent version of CQL (CQL-MA) (Kumar et al., 2020), ICQ-MA (Yang et al., 2021), TD3-BC (Fujimoto & Gu, 2021), and OMAR (Pan et al., 2022). Following most baseline methods, each algorithm runs with five seeds, where the performance is evaluated 20 times every 50 episodes.

Implementation Details. All baseline methods were implemented consistently with their official repositories, maintaining their hyper-parameters at their original best-performing settings. In our approach, we employed a 1-block Transformer-based discriminator and a 1-block Transformer-based generator for all tasks. Following MAT (Wen et al., 2022), the feature dimension for all Transformer blocks and MLP layers was set to 64. The learning rates for the actors and critics were set to 5e-4, 5e-4, 5e-5, and 5e-5 for the SMAC, Football, Bi-DexHands, and Ma-Mujoco benchmarks, respectively. The training of our models was conducted on a single NVIDIA Tesla V100 GPU. The batch size and update epoch for updating the generator and discriminator once were set to 128 and 5.

D.2 Ablation Studies

Analysis on different distribution matching settings. We comprehensively analyzed three variants of the $\mathbf{MILD}^2$ model to evaluate the effects of incorporating global dependencies on the discriminator and generator components. As shown in Tab. (3), a substantial deterioration in the model's performance was observed when the global dependencies were eliminated from the discriminator (i.e., independent modeling of individual reward distributions). Likewise, a significant decline in performance was observed when the global dependencies were removed from the generator (i.e., independently modeling individual policy distributions). These empirical findings emphasize the indispensability of introducing global dependencies and establishing joint reward and policy distributions within the model.

Analysis on different model architectures. We also compare the implementation of different model architectures to verify the necessity of different architecture components. We conducted this ablation study on both homogeneous and heterogeneous scenarios, as depicted in Tab. (4). The complete Transformer architecture achieved the best performance, highlighting the advantages of the Transformer and the necessity of the encoder-decoder framework.

Table 3: Performance comparison across different distribution matching settings to evaluate the effect of introducing global dependencies on the discriminator and generator, where $\mathbf{MILD}^2$ is the original implementation; Ind. Disc involves an independent discriminator (reward function) without global dependencies; Ind. Gen utilizes independent generator (individual policies) without global dependencies; Ind. D & G combines both independent discriminator and independent generator. The winning rates are shown in brackets.

Tasks	$\mathbf{MILD}^2$	$\mathbf{MILD}^2$ (Ind. Disc)	$\mathbf{MILD}^2$ (Ind. Gen)	$\mathbf{MILD}^2$ (Ind. D & G)
3m	**20.00±0.00**	19.91±0.06	18.88±0.06	19.97±0.05
	(1.00±0.00)	(0.99±0.01)	(0.99±0.01)	(0.99±0.01)
3s5z	**20.00±0.02**	19.80±0.08	19.89±0.08	19.92±0.02
	(1.00±0.00)	(0.95±0.02)	(0.97±0.02)	(0.94±0.02)
6h vs 8z	**19.78±0.07**	19.54±0.14	17.10±0.24	16.82±0.29
	(0.96±0.01)	(0.88±0.02)	(0.49±0.05)	(0.47±0.02)
MMM2	**20.52±0.09**	5.39±0.07	20.21±0.15	5.01±0.03
	(0.86±0.03)	(0.00±0.00)	(0.85±0.12)	(0.00±0.00)

Table 4: Performance comparison for different discriminator and generator architectures to explore the effect of each component, where $\mathbf{MILD}^2$ is the original implementation; $\mathbf{MILD}^2$-dec is implemented with the encoder only, without the autoregressive process; $\mathbf{MILD}^2$-enc is implemented with the decoder only, keeping the auto-regressive process.

Tasks	$\mathbf{MILD}^2$	$\mathbf{MILD}^2$-Dec	$\mathbf{MILD}^2$-Enc
3s5z	**20.00±0.02**	19.81±0.08	19.93±0.03
	(1.00±0.00)	(0.95±0.02)	(0.98±0.00)
6h vs 8z	**19.78±0.07**	16.22±0.21	19.57±0.16
	(0.96±0.01)	(0.39±0.03)	(0.92±0.02)
MMM2	**20.52±0.09**	4.87±0.03	18.29±0.35
	(0.86±0.03)	(0.00±0.00)	(0.64±0.03)
DoorOpenInward	**395.98±0.39**	382.31±0.05	386.23±0.78

D.3 Robustness Studies

D.3.1 Noisy data regime

In this section, we endeavor to substantiate our hypothesis, positing that the sequential autoregressive policy and reward models exhibit enhanced robustness. This heightened robustness stems from the model's action evaluation and decision-making processes contingent upon global correlations rather than solely relying on localized information. Even in scenarios where datasets encompass a degree of noisy transitions, the model continues to demonstrate commendable performance. This capability to harness global correlations proves particularly advantageous in settings characterized by non-Markovian dynamics, such as cooperative tasks, wherein the decisions made by other agents wield influence over future outcomes. To empirically investigate the validity of our hypothesis, we created a "mixed" dataset through the amalgamation of medium datasets (exploring with medium-score policies), deliberately introduced as sources of noise, with an expert dataset. This amalgamation comprises datasets featuring varying noise ratios. Subsequently, we subjected the proposed sequential autoregressive framework $\mathbf{MILD}^2$, along with the baseline models MAGAIL and ICQ-MA, to a battery of tests under diverse noise ratios, as illustrated in Fig. (9). The results consistently demonstrated that $\mathbf{MILD}^2$ outperformed MAGAIL and ICQ-MA across all experimental configurations. Notably, the performance of both MAGAIL and ICQ-MA exhibited conspicuous susceptibility to variations in the noise ratio, showing a precipitous decline as the noise ratio increased from 0% to 50%. In contrast, $\mathbf{MILD}^2$ performed well even when the noise ratio reached 50%. For instance, in the 6h vs 8z scenario, $\mathbf{MILD}^2$ maintained a win rate of approximately 90%, even in the presence of a 50% noise ratio. Conversely, in the 3s vs 5z scenario, MAGAIL and ICQ-MA displayed vulnerability to noise, even at lower noise ratios such as 5%.

D.3.2 Small data regime

This section aims to investigate the advantages of capturing global dependencies for in-sample generalization using $\mathbf{MILD}^2$. Precisely, we assess its robustness compared to baseline models when dealing with limited dataset size or reduced dataset diversity in specific states, common challenges in imitation learning with real-world data. To conduct this investigation, we curated custom datasets

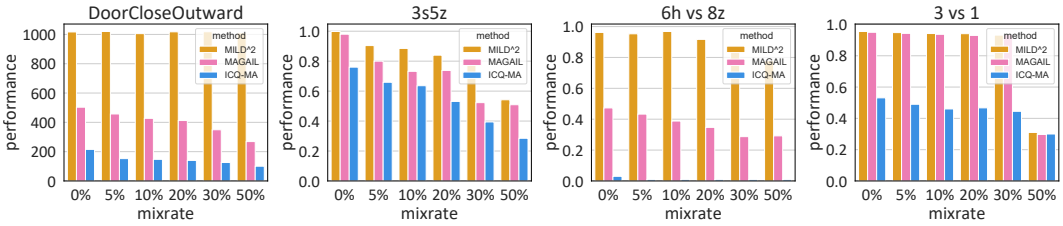

Figure 9: Performance comparison of various methods in noisy data regimes. In the leftmost environment (from the Bi-DexHands benchmark), cumulative rewards are employed as the performance evaluation metric, whereas in the remaining environments (from SMAC and Football benchmark), win rates serve as the performance evaluation metric.

Table 5: The average accumulated trajectory rewards (win rates) and standard deviation of $\mathbf{MILD}^2$, MAGAIL, and ICQ-MA in small data regimes.

Benchamarks		Bi-Dexhands	SMAC				Football	
Tasks		DoorCloseOutward	3s5z		6h vs 8z		counterattack	
Method	ratio	acc. rewards	acc. rewards	win. rate	acc. rewards	win. rate	acc. rewards	win. rate
$\mathbf{MILD}^2$	1%	1015.92±0.13	9.09±0.19	(0.00±0.00)	4.13±0.08	(0.00±0.00)	4.70±0.19	(0.90±0.04)
	5%	1015.84±0.11	18.87±0.21	(0.79±0.03)	19.00±0.11	(0.74±0.03)	4.75±0.19	(0.91±0.08)
	10%	1014.93±0.12	19.76±0.10	(0.94±0.01)	19.55±0.16	(0.93±0.03)	4.83±0.10	(0.93±0.02)
	20%	1015.74±0.12	19.85±0.09	(0.96±0.02)	19.77±0.16	(0.96±0.03)	4.89±0.12	(0.94±0.03)
	40%	1016.06±0.13	19.94±0.04	(0.98±0.01)	19.74±0.13	(0.96±0.02)	4.85±0.09	(0.93±0.02)
MAGAIL	1%	370.72±0.37	14.33±0.34	(0.29±0.04)	9.45±0.21	(0.00±0.00)	4.44±0.06	(0.85±0.01)
	5%	445.81±0.28	19.47±0.17	(0.89±0.03)	12.24±0.37	(0.11±0.03)	4.50±0.13	(0.85±0.03)
	10%	463.75±0.20	19.68±0.13	(0.89±0.03)	13.42±0.27	(0.17±0.04)	4.58±0.18	(0.87±0.04)
	20%	480.00±0.23	19.82±0.11	(0.96±0.03)	14.82±0.42	(0.32±0.04)	4.58±0.25	(0.88±0.06)
	40%	491.36±0.16	19.84±0.08	(0.96±0.02)	15.69±0.55	(0.35±0.08)	4.64±0.06	(0.89±0.01)
ICQ-MA	1%	149.74±0.21	13.25±0.45	(0.02±0.00)	8.09±0.22	(0.00±0.00)	0.98±0.19	(0.14±0.08)
	5%	189.97±0.12	15.03±0.62	(0.17±0.09)	9.42±0.21	(0.00±0.00)	1.57±0.49	(0.26±0.09)
	10%	202.61±0.09	16.24±0.69	(0.35±0.11)	9.37±0.30	(0.00±0.00)	1.78±0.59	(0.29±0.12)
	20%	216.46±0.14	17.50±0.52	(0.49±0.09)	9.38±0.38	(0.00±0.00)	2.02±0.55	(0.36±0.18)
	40%	177.30±0.12	18.29±0.60	(0.63±0.12)	9.88±0.44	(0.01±0.02)	1.15±0.28	(0.20±0.05)

by excluding specific transitions in datasets like DoorCloseOutward, 3s5z, 6h vs 8z, and counteratattack. The exclusion criteria are based on the proximity to the target location, simulating scenarios where data near the task goal is constrained due to the stochastic nature of data generation policies (Xu et al., 2023). We introduced a retention ratio parameter governing dataset composition to simulate demonstration data at different scales. We compared the performance of $\mathbf{MILD}^2$ to two other models, MAGAIL and ICQ-MA, by measuring average accumulated trajectory rewards (win rate) during evaluation and training standard deviation, as shown in Tab. (5). MAGAIL and ICQ-MA experience a significant drop in performance as the retention ratio decreases. In more challenging tasks, the standard deviation increases exponentially, indicating a substantial growth in generalization error with limited data. In contrast, $\mathbf{MILD}^2$ consistently demonstrates stable and commendable performance across various retention ratios. Additionally, $\mathbf{MILD}^2$ has a notably more minor standard deviation than MAGAIL and ICQ-MA. This compelling evidence highlights the advantage of capturing global dependencies, enabling better use of dataset samples to mitigate incorrect value estimations and improve overall performance.

D.4 Analysis of cooperation scale

As shown in Tab. (1), $\mathbf{MILD}^2$ performs better in tasks with large cooperation scales due to its ability to model global interdependencies among agents and increased variance of advantage actions. For example, in the case of SMAC, agents' scalability is demonstrated across four tasks: 3m, 3s5z, 6h vs 8z, and MMM2, with scalability factors of 3, 8, 8, and 12, respectively. $\mathbf{MILD}^2$ significantly outperforms baseline methods like MAGAIL, with an 81.27% improvement, especially in the expansive cooperative setting of MMM2. However, in smaller-scale environments like 3m, the improvement is only 0.15%.

Table 6: Hyperparameters used for the discriminator (reward model) in four benchmarks.

Benchmarks	Hyper-Parameter	Default Configuration
Common parameters	optimizer	Adam
	scheduler	StepLR
	hidden_size	64
	batch size	128
	learning rate	5e-6
	disc epoch	5
	disc warmup epoch	100
	disc warmup steps	10
	layers num	1
	heads num	1
Bi-DexHands	max grad norm	0.5
	distance metric	Wasserstein distance
	attention type	Dot-product attention
SMAC	max grad norm	10.0
	distance metric	KL divergence
	attention type	L2 attention
Football	max grad norm	0.5
	distance metric	KL divergence
	attention type	L2 attention
Ma-MuJoCo	max grad norm	0.5
	distance metric	Wasserstein distance
	attention type	Dot-product attention

Table 7: Common hyperparameters used for the generator (policy model) in the SMAC benchmark.

Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
critic lr	5e-4	actor lr	5e-4	use gae	TRUE
gain	0.01	optim eps	1e-5	batch size	3200
training threads	16	num mini-batch	1	rollout threads	32
entropy coef	0.01	max grad norm	10	episode length	100
optimizer	Adam	hidden layer dim	64	use huber loss	TRUE

These findings highlight that larger cooperative environments exhibit complex agent dependency structures. $\mathbf{MILD}^2$'s ability to capture and represent these global dependencies is crucial for accurately modeling cooperative relationships and allocating credit among agents. It helps reduce distribution matching errors caused by the non-stationarity of environmental dynamics.

E HYPERPARAMETERS

During experiments, the implementations of baseline methods are consistent with their official repositories, and all hyper-parameters left unchanged at the origin best-performing status. The hyperparameters adopted for the discriminator are listed in Tab. (6), and those adopted for the generator are listed in Tab. (7-11).

Table 8: Different hyperparameters used for the generator (policy model) in the SMAC benchmark.

Tasks	ppo epochs	ppo clip	num blocks	num heads	stacked frames	steps	γ
3m	15	0.2	1	1	1	5e5	0.99
3s5z	10	0.05	1	1	1	3e6	0.99
6h vs 8z	15	0.05	1	1	1	1e7	0.99
MMM2	5	0.05	1	1	1	1e7	0.99

Table 9: Hyperparameters used for the generator (policy model) in the Bi-Dexhands benchmark.

Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
cirtic lr	5e-5	actor lr	5e-5	hidden dim	64
gamma	0.96	steps	5e7	stacked frames	1
gain	0.01	optim eps	1e-5	ppo epochs	5
ppo clip	0.2	num mini-batch	1	rollout threads	80
batch size	6000	episode length	75	optimizer	Adam
entropy coef	0.001	max grad norm	0.5	training threads	16

Table 10: Hyperparameters used for the generator (policy model) in the Football benchmark.

Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
cirtic lr	5e-4	actor lr	5e-4	gamma	0.99
ppo clip	5e-2	num head/block	1	ppo epochs	10
gain	0.01	optim eps	1e-5	batch size	4000
training threads	16	num mini-batch	1	rollout threads	20
entropy coef	0.001	max grad norm	0.5	episode length	200
optimizer	Adam	hidden layer dim	64	stacked frames	1

Table 11: Hyperparameters used for the generator (policy model) in the Ma-Mujoco benchmark.

Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
cirtic lr	5e-5	actor lr	5e-5	ppo epochs	10
ppo clip	5e-2	num block	1	num head	1
gamma	0.99	steps	1e7	stacked frames	1
gain	0.01	optim eps	1e-5	batch size	4000
training threads	16	num mini-batch	40	rollout threads	40
entropy coef	0.001	max grad norm	0.5	episode length	100
optimizer	Adam	hidden layer dim	64	use huber loss	TRUE