Opponent Modeling based on Subgoal Inference

Xiaopeng Yu **Jiechuan Jiang** **Zongqing Lu**[†]

School of Computer Science, Peking University

Abstract

When an agent is in a multi-agent environment, it may face previously unseen opponents, and it is a challenge to cooperate with other agents to accomplish the task together or to maximize its own rewards. Most opponent modeling methods deal with the non-stationarity caused by unknown opponent policies via predicting the opponent's actions. However, focusing on the opponent's action is shortsighted, which also constrains the adaptability to unknown opponents in complex tasks. In this paper, we propose *opponent modeling based on subgoal inference*, which infers the opponent's subgoals through historical trajectories. As subgoals are likely to be shared by different opponent policies, predicting subgoals can yield better generalization to unknown opponents. Additionally, we design two subgoal selection modes for cooperative games and general-sum games respectively. Empirically, we show that our method achieves more effective adaptation than existing methods in a variety of tasks.

1 Introduction

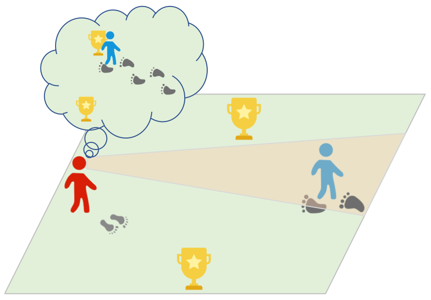


Figure 1: Infer the goal of others

Autonomous agents are systems capable of making decisions and acting independently in their environment, often operating without direct human intervention [3]. These agents can either cooperate with or compete against each other, depending on the context. In cooperative scenarios, many multi-agent reinforcement learning (MARL) methods [20, 41, 33, 38] aim to bridge the information gap between agents [44] by training agents in a centralized manner, called centralized training with decentralized execution, enabling agents to work together seamlessly to accomplish cooperative tasks. Alternatively, fully decentralized methods[17, 40] seek to break free from the constraints of centralized training, allowing agents to reach collaboration in a simpler and decentralized manner. In competitive scenarios, NFSP [15], PSRO [19], and DeepNash [29] employ self-play to train agents for equilibrium strategies, allowing agents to adapt and improve their policy. By considering how the agent affects the expected learning progress of other agents, LOLA [10] and COLA [49] apply opponent shaping to this setting. Overall, these methods focus on training agents in a way that accounts for their interactions, resulting in a set of policies that enable effective collaboration or competition within a group of agents.

While the above methods emphasizes the collective behavior of agents, it is also crucial to consider the role of individual agents, particularly self-interested agents, in these multi-agent environments. A self-interested agent [37, 12] operates with the primary goal of maximizing its own benefits,

[†]Correspondence to ✉ `zongqing.lu@pku.edu.cn`

even when interacting with other agents. When the objectives of a self-interested agent align with those of the team, this scenario falls under ad-hoc teamwork [21, 8, 39]; however, in more general cases, these interactions are framed as noncooperative games [34, 23, 45]. A key technique for self-interested agents in such settings is opponent modeling[24, 3], which enables them to analyze and predict the actions, goals, and beliefs of other agents. By modeling the intentions and policies of other agents, the training process of the agent might be stabilized [27]. Many studies rely on predicting the actions [14, 16, 13, 25, 26], goals [32, 31], and returns [42] of opponents during training. Then, the autonomous agent can adapt to different or unseen opponents by using the predictions or representations that are produced by the relevant modules.

Although a lot of the existing methods concentrate on modeling the opponent's actions, we argue that such an approach is short-sighted, pedantical, and highly complex. Generally, modeling an opponent's actions just predicts what it will do at the next step. Intuitively, it is more beneficial for the agent to make decisions if it knows the situation of the opponent several steps ahead. Predicting the actions over a few steps has high uncertainty. For example, to reach the goal point of $(2, 2)$, an opponent moves from $(0, 0)$ following the action sequence $<\uparrow, \uparrow, \rightarrow, \rightarrow>$ by four steps (Cartesian coordinates). But, there are also 5 other action sequences, *i.e.,* $<\uparrow, \rightarrow, \uparrow, \rightarrow>$, $<\uparrow, \rightarrow, \rightarrow, \uparrow>$, $<\rightarrow, \uparrow, \uparrow, \rightarrow>$, $<\rightarrow, \uparrow, \rightarrow, \uparrow>$, $<\rightarrow, \rightarrow, \uparrow, \uparrow>$, that can lead to the same goal. Obviously, the complexity and uncertainty of predicting the action sequence are much higher than the goal itself. Other methods that claim to predict the opponent's goal [31, 32], but without explicitly making a connection to the opponent's goal or just predicting the goal at the next step, are essentially as shortsighted as modeling actions.

Inspired by the fact that humans can predict the opponent's goal by observing the opponent's actions for several steps as illustrated in Figure 1, in this paper, we propose ***O**pponent **M**odeling based on sub**G**oals inference* (**OMG**), which uses variational inference to predict the opponent's future subgoals from historical trajectories. The trajectory of an opponent's policy consists of a set of subgoals, and the trajectories of different policies may contain the same subgoals. This combinatorial property of the subgoals facilitates the generalization of the agent to unseen opponents' policies. Moreover, we design two manners for selecting subgoals, which are applied to cooperative games and general sum games, respectively. Empirically, OMG outperforms existing opponent modeling methods in a variety of multi-agent environments, demonstrating the superiority of inferring subgoals over predicting actions.

2 Related Work

Opponent modeling. Opponent modeling plays a crucial role in enhancing the robustness and stability of reinforcement learning [27]. Given the presence of diverse opponent policies in multi-agent environments, the autonomous agent faces a significant challenge in learning resilient policies. When an agent perceives an opponent as part of the environment, the resulting environment becomes inherently unstable and intricate. To address this challenge, one straightforward method involves equipping the agent with the ability to incorporate information about its opponent, including aspects like the opponent's behavior, goals, and beliefs [3], *i.e.,* opponent modeling. It gives the agent a deeper insight and prediction ability about the opponent's policy. Thus, the autonomous agent views the environment as less unstable and can simply use single-agent reinforcement learning methods.

A common approach to modeling the policy of an opponent is predicting the opponent's actions. DRON [14] and DPIQN [16] extend DQN [22] by adding another network that estimates the opponents' actions from the observations. The DQN uses the hidden layer of this network to improve its policy. Variational auto-encoders can also be used to model the opponent's policy [25], which results in probabilistic representations instead of fixed vectors. PR2 [48] and TP-MCTS [47] combine the idea of recursive reasoning, nested form as "the agent believes [that the opponent believes (that the agent believes ...)]", based on modeling the action of the opponent. Some works focus on modeling beliefs. [53] combined the sequential and hierarchical variational auto-encoders to construct a belief inference model using meta-learning, for belief inference. [51] introduced landmarks into the behavior model and improved the model by the action sequence of the opponents, so as to recognize and compare the opponent's intention.

Another key aspect of opponent modeling is to infer the opponent's goal. [5] formulated the goal recognition as a Markov decision process (MDP) and calculated the posterior probability of the

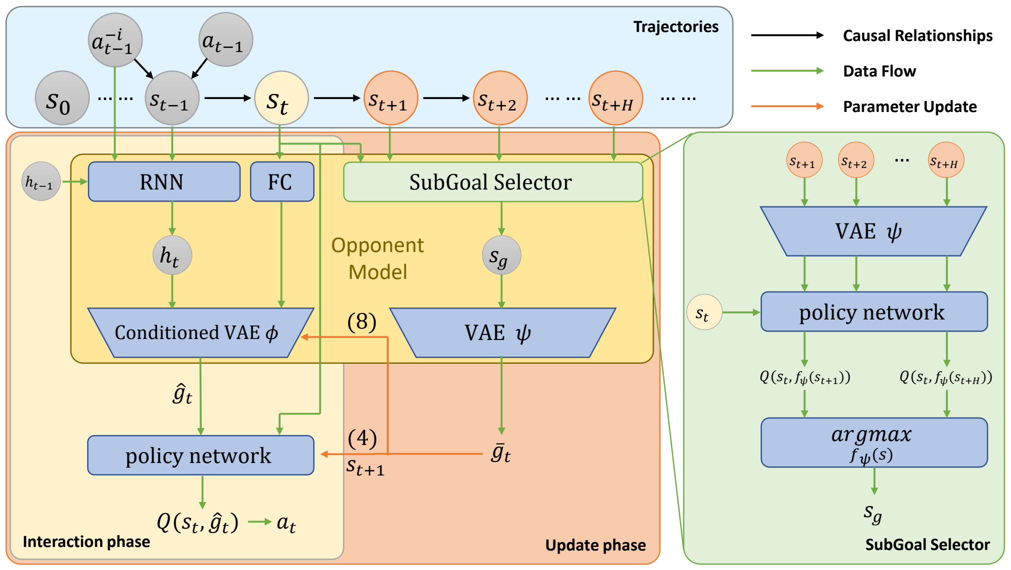


Figure 2: Diagram of OMG. During the interaction phase, OMG infers the subgoal from the historical trajectory. The CVAE ϕ acts as an inference model, deducing the opponent subgoal denoted as $\hat{g}$. The inferred subgoal serves as input for the policy model $Q(s, \hat{g}, a)$. In the update phase, OMG examines the entire trajectory in a hindsight manner to select subgoals $\bar{g}$ as priors for training the inference model. The subgoal selector employs a value-based heuristic to choose a state from the next few steps and then encodes it into a subgoal using the pre-trained VAE ψ.

goal by Bayes' rule based on a prior goal library. ToMnet [31] aims to give the agent a human-like Theory of Mind. It uses three networks to infer the agent's goal and action from previous and present information. SOM [32] implements the Theory of Mind with a goal library from a different perspective. SOM uses its own policy, the opponent's observation, and the opponent's action to work backward to learn the opponent's goal distribution by gradient ascent. These methods either require a prior goal library or infer implicit "goals" that are not supervised by ground truth goals.

In some scenarios, opponents may continuously learn during interaction. Meta-MAPG [18] combines Meta-PG [1] and LOLA [10], and focuses on the problem of the non-stationary environment caused by the continuous learning of opponents. MBOM [50] simultaneously targets a variety of adversaries, fixed policy, or continuous learning, by modeling the possible policies that an opponent may form, combined with Bayesian inference to generate an opponent's imagined policy. GSCU [11] chooses online between a real-time greedy strategy and a fixed conservative strategy through Bayesian belief in competitive environments. Unlike these methods, in this paper, we consider the most common setting where opponents have unseen, diverse, but fixed policies during test.

Goal-conditioned RL. Goal-conditioned reinforcement learning is an extension of the single-agent algorithm. Most works focus on learning a goal-conditioned policy, where the goals are usually predefined [30, 52]. Some works consider acquiring subgoals automatically to accelerate learning. [28] proposed a method that uses expert trajectories to generate subgoals, while [7] proposed to incorporate imaginary subgoals into policy learning to facilitate learning complex tasks, where subgoals are measured by value functions. Unlike existing goal-conditioned RL methods, we aim to infer the subgoal of the opponent and condition the agent policy on the inferred subgoal.

3 Preliminaries

In general, we consider an n-agent stochastic game $\mathcal{M} = (\mathcal{S}, \mathcal{A}^1, \dots, \mathcal{A}^n, \mathcal{P}, \mathcal{R}^1, \dots, \mathcal{R}^n, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}^i$ is the action space of agent $i \in [1, \dots, n]$, $\mathcal{A} = \prod_{i=1}^n \mathcal{A}^i$ is the joint action space of agents, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a transition function, $\mathcal{R}^i : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function of agent i , and $\gamma \in [0, 1)$ is the discount factor. The policy of agent i is π^i, and the joint policy of other agents is $\pi^{-i}(a^{-i}|s) = \prod_{j \neq i} \pi^j(a^j|s)$, where a^{-i} is the joint action except agent

i. All agents interact with the environment simultaneously without communication. The historical trajectory is available, *i.e.*, for agent i at timestep t, $\tau_t = \{s_0, a_0^i, a_0^{-i}, \ldots, s_{t-1}, a_{t-1}^i, a_{t-1}^{-i}\}$ is observable. The goal of the agent i is to maximize its expected cumulative discount rewards:

$$\mathop{\mathbb{E}}_{\substack{s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t^i,a_t^{-i}),\\ a\sim\pi^i(\cdot|s_t),a_t^{-i}\sim\pi^{-i}(\cdot|s_t)}}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}^i(s_t,a_t^i,a_t^{-i})\right]. \quad (1)$$

For convenience, the learning agent treats all other agents as a joint opponent with the joint action $a^{-i} \sim \pi^{-i}(\cdot|s)$ and reward r^{-i}. The action and reward of the learning agent are respectively denoted as $a \sim \pi(\cdot|s)$ and r for notation simplicity.

If an agent treats other agents as part of the environment and ignores the non-stationarity posed by the change of other agents' policies as independent Q-learning [43, 44]. Its Q-function $\mathcal{Q}$ is updated by:

$$\mathcal{Q}(s_t, a_t) = \mathbb{E}_{\mathcal{P}(s_{t+1}|s_t,a^{-i},a)}[r + \gamma \max_a \mathcal{Q}(s_{t+1}, a)]. \quad (2)$$

Opponent modeling typically predicts the actions of other agents to address the non-stationary problem. The opponent model uses historical trajectory as input to predict $\tilde{a}^{-i} \sim \tilde{\pi}(\cdot|\tau)$, where $\tilde{a}^{-i}$ is the estimate of a^{-i}. Then, its Q-function is updated as:

$$\mathcal{Q}(s_t, \tilde{a}_t^{-i}, a_t) = \mathbb{E}_{\mathcal{P}(s_{t+1}|s_t,a^{-i},a)}[r + \gamma \max_a \mathcal{Q}(s_{t+1}, \tilde{a}_{t+1}^{-i}, a)]. \quad (3)$$

Note that we cast our discussion here to Q-learning. All can be similarly applied to other RL methods, such as PPO [36].

4 Method

In this section, we present our method, opponent modeling based on subgoals inference (OMG). First, we discuss learning policies with the opponent's subgoals, compared to learning based on the opponent's actions. Then, we introduce our opponent model that infers the opponent's subgoals using a value-based heuristic.

4.1 Policy Learning with Opponent's Subgoals

In Equation (3), the traditional opponent modeling with the opponent's actions is introduced. Here, we introduce policy learning with the opponent's subgoals.

The opponent's subgoals offer a more structured representation compared to individual actions. Subgoals represent feature embeddings of future states that the opponent aims to achieve based on its policy. Although diverse action sequences can lead to the same state, focusing on subgoals provides a higher-level understanding of the opponent's long-term intentions. Instead of gaining new information, subgoal modeling reinterprets observed data to emphasize long-term objectives, reducing variability and improving learning efficiency [24]. By concentrating on the opponent's desired states rather than individual actions, the agent can achieve more stable and effective policy learning.

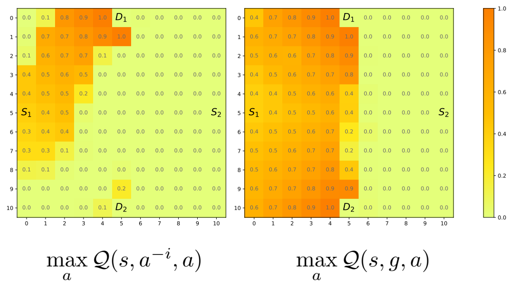
How are opponent's desired states inferred?
—> subgoal selector either argmin or argmax of Qfn

The opponent's subgoal distribution is determined by the opponent's action sequence, *i.e.*, the opponent's policy, but the subgoal space is still the representation of the state space. Here we decouple the subgoal from the opponent's policy and just consider decision-making problems conditioned on the opponent's subgoal. Formally, we transform the original stochastic game $\mathcal{M}$ into a state-augmented MDP, defined by $\mathcal{M}_\mathcal{G} = (\mathcal{S}, \mathcal{G}, \mathcal{A}^i, \mathcal{P}, \mathcal{R}^i, \gamma)$, where $\mathcal{G}$ is the subgoal space. $\mathcal{G}$ is a representation of future states the opponent may go, $|\mathcal{G}| \leq |\mathcal{S}|$.

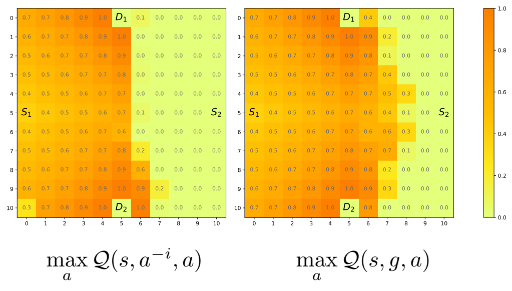
The state-augmented MDP's state space $\mathcal{S}$ extends to the MDP with state-subgoal pairs $(\mathcal{S}, \mathcal{G})$. Therefore, the agent's Q-function based on the opponent's subgoal is updated as:

$$\mathcal{Q}(s_t, g_t, a_t) = \mathbb{E}_{\mathcal{P}(s_{t+1}|s_t,a^{-i},a)}[r + \gamma \max_a \mathcal{Q}(s_{t+1}, g_t, a)]. \quad (4)$$

Here the pair (s_{t+1}, g_t) is used instead of (s_{t+1}, g_{t+1}), as we assume that the next state of (s_t, g_t) follows the same goal. In the framework of OMG, g_t and g_{t+1} will reach the same state at the end of the episode.



(a) training steps $= 1e5$



(b) training steps $= 1e7$

Figure 3: Learned Q-values using tabular Q-learning in an 11×11 gridworld. The agent and the opponent start from the S_1 and S_2, respectively. The two rewarding grids are D_1 and D_2, and the reward will only be given to the agent who arrives first. The opponent executes one of policies π_1^{-i} and π_2^{-i}, which target D_1 and D_2, respectively. The g and a^{-i} are obtained from an oracle.

Q-values with opponent's subgoals are just as effective as with opponent's actions. We carry out an experiment in an 11×11 gridworld with two agents, as detailed in Figure 3. After convergence, the Q-value increases as the agent gets closer to the rewarding grid, indicating a meaningful Q-value with the opponent's subgoal, as shown in Figure 3(b).

Effective opponent's subgoals enhance policy learning. The Q-value using the opponent's action learns slower than the Q-value with the opponent's subgoal in Figure 3(a), resulting from the tuple (s, a^{-i}, a) is more numerous than (s, g, a) in the Q-table. When there are fewer (s, g, a) than (s, a^{-i}, a), the method using (s, g, a) naturally holds the advantage of faster learning than the method of (s, a^{-i}, a). The quantity of (s, g, a) is contingent upon the goal selection, and we present an analysis of the quantitative relationship between pair (s, g) and (s, a^{-i}), see Appendix A. In short, the number of (s, g) is significantly smaller than that of (s, a^{-i}) in our method. Next, we explain how the opponent model secures these benefits.

4.2 Opponent Modeling Based on Subgoal Inference

Our opponent modeling consists of two components: *subgoal inference model* and *subgoal selector*. The subgoal inference model employs the historical trajectory to predict the opponent's subgoal, serving as input for the policy during the interaction phase. The subgoal selector scrutinizes the entire historical trajectory using a value-based heuristic to choose the appropriate subgoal for training the inference model during the update phase.

Subgoal inference model. The subgoal g represents a feature embedding of a future state. Specifically, for a trajectory $\{s_0, a_0, a_0^{-i}, \ldots, s_t, a_t, a_t^{-i}, \ldots, s_T\}$, the state corresponding to subgoal g_t at s_t is one of future states $\mathcal{N}_t = \{s_{t+1}, s_{t+2}, \ldots, s_T\}$, denoted as s_t^g and determined by the subgoal selector.

The objective of the subgoal inference model is to infer g_t from the historical trajectory $\tau_t = \{s_0, a_0, a_0^{-i}, \ldots, s_{t-1}, a_{t-1}, a_{t-1}^{-i}\}$. This aligns with the intuitive hypothesis that the opponent's intention can often be inferred after just a few initial actions.

Here, we introduce variational inference and employ a conditional variational auto-encoder (CVAE) as the subgoal inference model. In this model, we represent the subgoal posterior probability as $q_\phi(\hat{g}_t|\tau_t, s_t)$ and the likelihood estimate as $p_\theta(s_t|\hat{g}_t, \tau_t)$ with ϕ and θ respectively denoting the network parameters. The subgoal prior model, denoted as p_ψ, is a pre-trained variational autoencoder (VAE) using the states previously collected in the environment, and produces the subgoal prior $p_\psi(\bar{g}_t|s_t^g)$ given the subgoal state s_t^g chosen by the subgoal selector.

Further details about the network architecture are provided in Figure 2. The optimization objective of the subgoal inference model is:

$$< \hat{\theta}, \hat{\phi} >= \arg\max_{\theta,\phi} \mathbb{E}_{q_\phi(\hat{g}_t|\tau_t,s_t)} \Big[\log p_\theta(s_t|\hat{g}_t, \tau_t) \Big] - \mathrm{KL}\Big(q_\phi(\hat{g}_t|\tau_t, s_t) \| p_\psi(\bar{g}_t|s_t^g) \Big). \tag{5}$$

Algorithm 1 OMG

```
1:  ***Preparation:***
2:  Interact with $\nu$ opponents to collect $s$ and train the prior model $p_\psi$
3:  Initialize subgoal inference model parameters $\phi$ and $\theta$
4:  Initialize Q-network $\mathcal{Q}$ and the replay buffer $\mathcal{D}$
5:  **repeat**
6:      ***Interaction phase***
7:      Observe state $s$
8:      Infer $\hat{g}$ by subgoal inference model $q_\phi(\hat{g}|\tau, s)$
9:      Choose action $a$ by $\max_a \mathcal{Q}(s, \hat{g}, a)$ with $\epsilon$-greedy
10:     Store experience $(s, a, a^{-i}, r)$ in replay buffer $\mathcal{D}$
11:     ***Update phase***
12:     **if** time to update **then**
13:         Obtain prior subgoal $\bar{g}$ by (6) or (7)
14:         Calculate subgoal $g$ by (8)
15:         Update Q-network by (4)
16:         Update subgoal inference model $q_\phi$ and $p_\theta$ by (5)
17:     **end if**
18: **until** convergence
```

where the term $p_\psi(\bar{g}|s^g)$ in the KL divergence accounts for the prior distribution and is pre-trained. The purpose of including the KL divergence term is to prevent collapse of the inference model.

Subgoal selector. The primary objective of the subgoal selector is to choose the appropriate future state of the subgoal state s_t^g from $\mathcal{N}_t$ as input to the prior model. The choice of the subgoal state plays a significant role in shaping the agent's behavior and leaning towards either optimism or conservatism. This is especially critical when dealing with cooperative games and general-sum games, where the dynamics of interactions are complex and multifaceted. In these contexts, we provide two distinct manners for the subgoal selection:

$$\bar{g}_t = \arg\max_{s_i \in \mathcal{N}_t^H} \mathbb{E}_{g \sim p_\psi(\cdot|s_i)} V(s_t, g) \tag{6}$$

$$\bar{g}_t = \arg\min_{s_i \in \mathcal{N}_t^H} \mathbb{E}_{g \sim p_\psi(\cdot|s_i)} V(s_t, g), \tag{7}$$

where $V(s, g) = \mathbb{E}_a Q(s, g, a)$, $\mathcal{N}_t^H$ is the set of future states $\{s_{t+1}, \cdots, s_{t+H}\}$. As discussed in Section 4.1, the quantity of (s, g) pairs is crucial. Selecting candidate subgoal states is pivotal in this regard. Thus, we use states within the next H timesteps instead of all future states. The choice of H gives a tradeoff between the agent's generalization to diverse opponents induced by the fact that the subgoals of different trajectory fragments have combinatorial properties and the learning difficulty incurred by the increased opponent subgoals.

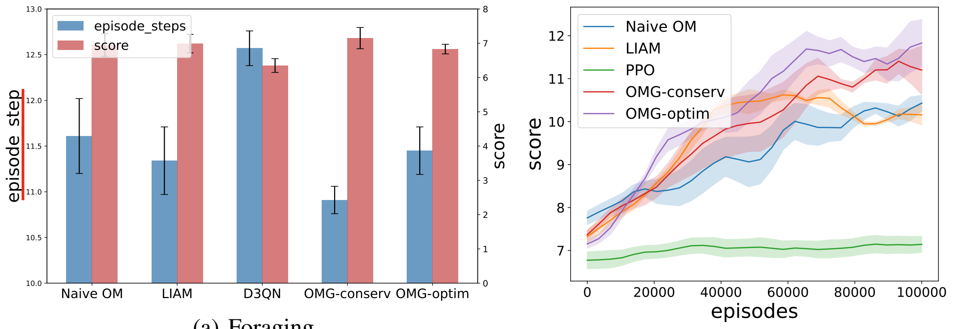
As indicated in Equation (6), we pinpoint the subgoal within an H-horizon that maximizes the V-value. The agent incorporates this to optimize the Q-function, thus adopting an optimistic strategy akin to the maximax strategy [6], which applies to cooperative games. Conversely, if we choose the subgoal as in Equation (7), it corresponds to the subgoal yielding the lowest value. The agent then employs this for learning Q-function, leading to a conservative strategy similar to the minimax strategy, which is commonly used in general-sum games.

The subgoal selector and the subgoal inference model as a whole constitute our opponent modeling module. During the interaction phase, the subgoal inference model is used to get the inferred subgoal $\hat{g}$, which is combined with the state as the input to the Q-network. During the update phase, the prior subgoal $\bar{g}$ generated by the subgoal selector is provided to the inference model for training. The subgoal inference model is unstable at the beginning, which may disturb the updating of the Q-network. Therefore, we use the following combination of the prior subgoal $\bar{g}$ and the inferred subgoal $\hat{g}$ as the input of Q-network,

$$g_t = \hat{g}_t \mathbb{I}(\eta > \epsilon) + \bar{g}_t \mathbb{I}(\eta \leq \epsilon), \quad \eta \sim U[0, 1], \tag{8}$$

where ϵ is a hyperparameter that decreases to zero over training. We will further empirically study this in Section 5.4.

For completeness, the full procedure of OMG is given in Algorithm 1.



(a) Foraging

(b) Predator-Prey

Figure 4: (a) Performance in Foraging. The red bar shows the total score obtained by the agent. The blue bar illustrates the number of steps in each episode. The results show that OMG can converge to the same score as the baselines but end the episode in fewer steps because it predicts the opponent's goal. (b) Performance in Predator-Prey. The results show the score obtained by the agent as a predator with two other uncontrolled predators, and OMG outperforms the baselines.

5 Experiments

First, we evaluate OMG's training performance in two environments (discrete and continuous state spaces) and then test its generalization to opponents with unseen policies in a more complex environment. In all the experiments, the baselines have the same neural network architectures as OMG. All the methods are trained for five runs with different random seeds, and results are presented using mean and standard deviation. More details about experimental settings and hyperparameters are available in Appendix B. To ensure reproducibility, we include the code in the supplementary material and will make it open-source upon acceptance.

We experiment in the following three multi-agent environments. Foraging [2, 4] is an 8×8 gridworld where the agent aims to collect foods. Predator-Prey [20] is a three-against-one scenario with continuous space where the agent collaborates with predators to capture prey. SMAC [35] is a high-dimensional environment for collaborative multi-agent reinforcement learning based on StarCraft II, where the agent cooperates with a set of opponents with unknown policies to accomplish tasks.

5.1 Baselines

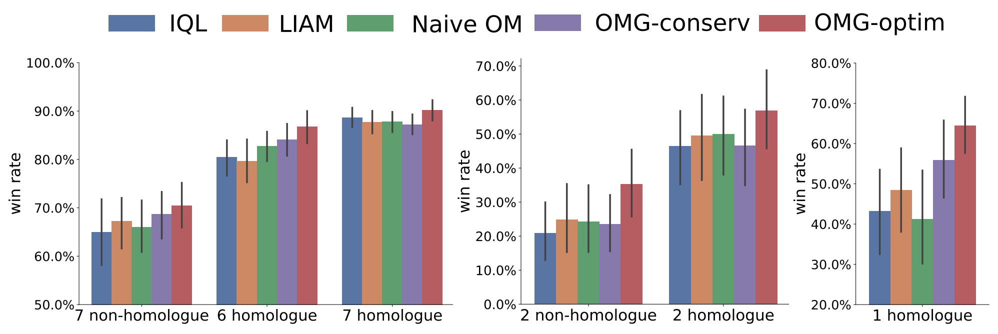
In the experiments, we implement two variants of OMG, OMG-optimistic and OMG-conservative, based on the subgoal selection manners in Equation (6) and Equation (7), respectively. OMG compared with the following methods:

- Naïve OM [14] uses observation to directly model the opponent's policy, which assists the agent in decision-making by predicting the opponent's actions.
- LIAM [26] uses the observations and actions of the opponent with an encoder-decoder architecture, and the model learns to extract representations about the opponent, conditioned only on the local observations of the controlled agent.
- D3QN & PPO & IQL [46, 36, 43] are classical RL algorithms without opponent modeling.

We use D3QN, PPO, and IQL as the backbone algorithms in Foraging, Predator-Prey, and SMAC, respectively, to reproduce the performance of baselines. The versions of OMG that are based on D3QN and IQL incorporate "dueling" and "double" tricks over Algorithm 1. For OMG based on PPO, please refer to Appendix F for details.

5.2 Performance of Training

We evaluate the performance of OMG on Foraging and Predator-Prey, and the results are shown in Figure 4(a) and Figure 4(b), respectively. In the foraging environment, our method attains similar scores to the baseline methods, and both the agent and the opponent achieve comparable scores.



(a) 8m (b) 3s_vs_5z (c) 2c_vs_64zg

Figure 5: Test performance of cooperation with unseen opponents in *8m* (a), *3s_vs_5z* (b) and *2c_vs_64zg* (c) maps of SMAC. The X-axis represents the opponent's policies, and "homologue" refers to the policy learned by the same algorithm, while "non-homologue" represents different ones; e.g. 7 homologue refers to 7 opponents from 8 agents trained by the same algorithm (QMIX, VDN or IQL) on the 8m, and 7 non-homologue involves 7 opponents from different runs of those algorithms. The results show that OMG-optimistic outperforms all baselines. The results are averaged over collaborating with 30 opponents of different policies, with 95% confidence intervals.

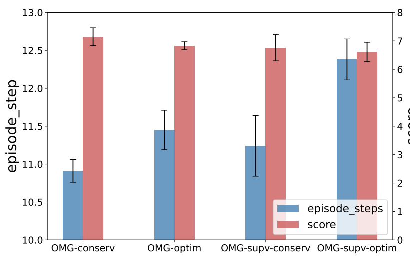
OMG has a shorter episode length compared to other methods because OMG can predict the subgoal that the opponent is heading to and thus avoid wasting steps in the same direction. In addition, the results show that OMG-conservative is more suitable than OMG-optimistic in this scenario since this is a general-sum game. The baselines based on action modeling, LIAM and Naïve OM, demonstrate comparable performance, whereas D3QN without opponent modeling, exhibits subpar results. In the predator-prey environment, the agent acts as the predator and collaborates with the other two uncontrolled predators to catch the prey. The results in Figure 4(b) show that OMG obviously outperforms action modeling methods, which demonstrates that OMG can also work efficiently in continuous state space. PPO without opponent modeling can hardly improve performance in training due to the non-stationarity caused by opponents. OMG-optimistic slightly performs better than OMG-conservative because OMG-optimistic is suitable for the cooperative game.

5.3 Generalization to Unknown Opponents

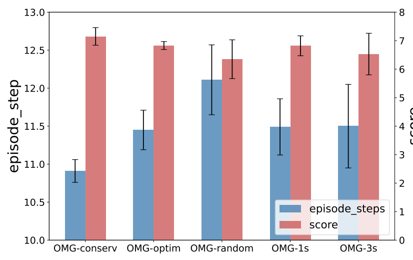
We evaluate the generalization of OMG in a more complex multi-agent environment, SMAC, which enables the opponents to exhibit more diverse policies. The experimental results of *8m*, *3s_vs_5z* and *2c_vs_64zg* are shown in Figure 5. Without opponent modeling, IQL struggles to adapt to various unknown opponents, resulting in poor performance, especially when the opponents are *non-homologue*. This underscores the effectiveness of opponent modeling in autonomous agent tasks. LIAM and Naïve OM, the action modeling methods, contribute to the team's improved win rate to some extent. The mediocre performance of OMG-conservative is attributed to its overly cautious subgoal selection. OMG-conservative is on par with IQL, which is consistent with the "conservative". OMG-optimistic surpasses the baseline methods, indicating that OMG-optimistic can generalize well to unknown collaborators through positive subgoal selection.

5.4 Ablation Study

The ablation study is conducted for the network structure of the inference model, subgoal selection, and hyperparameter horizon H. OMG uses CVAE as the inference model. Here, we instead employ supervised learning to train an inference model using the subgoal selector's output $\bar{g}_t$, obtained from either Equation (6) or Equation (7). This model is referred to as OMG-supervised. The results in the foraging environment are presented in Figure 6(a). The results indicate that OMG-optimistic and OMG-conservative outperform their counterparts, which is attributed to the enhanced adaptability of variational inference to the uncertainty in the opponent's policy. Dealing with multiple opponents employing distinct policies poses a challenge for supervised learning, as establishing a mapping relationship between historical trajectories and subgoals becomes intricate.

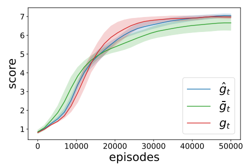


(a) Inference model architectures

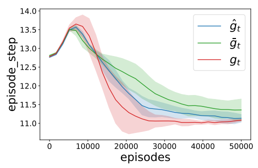


(b) Subgoal selection strategies

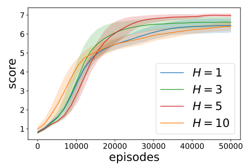
Figure 6: Ablation study in Foraging. In (a), methods on the X-axis labeled with "supv" indicate that the inference model uses an MLP instead of a CVAE. In (b) OMG-random, OMG-1s, and OMG-3s represent subgoals selected from the opponent's future states: randomly, at the next step, and at the third step, respectively.



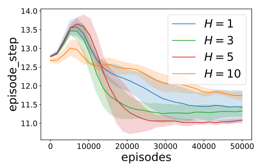
(a) Score



(b) Episode length



(c) Score



(d) Episode length

Figure 7: Ablation study of OMG in Foraging. (a) and (b) compares OMGs with different subgoal inputs for policy learning. (c) and (d) show ablation study for the hyperparameter horizon H.

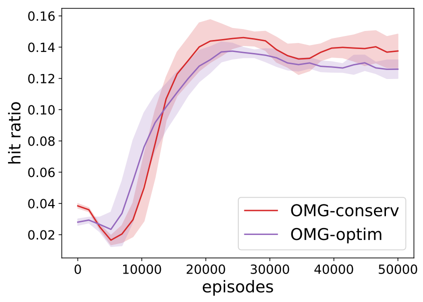
In the OMG, the subgoal is selected by choosing the state within the future H steps that either maximizes or minimizes the value function $V(s, g)$. To explore the impact of different subgoal selection strategies, we introduced three alternatives: random selection within the H steps (OMG-random), selecting the first step as the subgoal (OMG-1s), and selecting the third step as the subgoal (OMG-3s). The results, presented in Figure 6(b), suggest that the choice of subgoal selection strategy significantly affects performance, with OMG's strategy leading to more effective training compared to the alternatives. We also observe that the subgoal often remains constant over consecutive time steps for OMG-supervised. Further details can be found in Appendix D.

We further investigate our design choice on the subgoal selection for the policy. During the policy update, Equation (8) (*i.e.*, g) is utilized. As p_ψ is pre-trained and fixed during the update phase, $\bar{g}$ remains stable. On the other hand, $\hat{g}$, which represents the inferred subgoal when executing the policy, also stabilizes as the training steps increase. Thus, we choose a gradual transition of g from $\bar{g}$ to $\hat{g}$, which should help avoid instability during the training of the subgoal inference model. Here we perform the experiments in the foraging environment with different subgoal inputs for the policy, *i.e.*, $g, \hat{g}, \bar{g}$. As shown in Figure 7(a) and Figure 7(b), OMG with g indeed shows faster and better convergence.

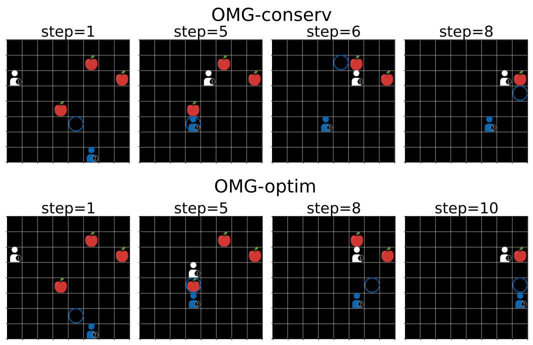
The parameter H denotes the horizon of the subgoal selector. The ablation experiment results are shown in Figure 7(c) and Figure 7(d). It is observed that an appropriate horizon value is neither excessively high nor excessively low. When $H = 1$, it is essentially equivalent to combining with QSS [9] and opponent modeling, which can be interpreted as another way of action modeling. However, if H is set too large, such as $H = 10$, the agent may skip important states in the trajectory, leading to a degradation in performance. Therefore, selecting an appropriate value for H is crucial in achieving satisfactory results.

5.5 Inferred Subgoal Analysis

We analyze the predictive performance of the opponent model. In Figure 8(a), we plot the ratio of that an opponent's future trajectory passes through the opponent's subgoal state, termed subgoal hit ratio. The subgoal state is reconstructed by the inferred subgoal $\hat{g}$ using the decoder of the subgoal prior model. The subgoal hit rate gradually improves during training, which indicates that



(a) Hit ratio of subgoal state



(b) Illustration of inferred subgoal state

Figure 8: Subgoal analysis of OMG in Foraging. The subgoal hit rates for OMG-conservative and OMG-optimistic are shown in Figure 8(a). In Figure 8(b), the agent controls player 1 (white), and the opponent controls player 2 (blue). The blue circle represents the state obtained through the reconstruction of the subgoal inferred by the agent. The figure illustrates the difference between OMG-conservative and OMG-optimistic under the same initial state and opponent policy.

the subgoal-based opponent modeling is able to predict the future state of the opponent. OMG tends to predict the opponent's future state several steps ahead as the subgoal, rather than focusing solely on the next step. This kind of prediction requires validation over multiple steps, and the agent policy conditioned on the predicted subgoal may also influence the behavior of the opponent. These make it challenging to verify the predicted subgoal. Consequently, the overall hit ratio remains at a moderate level at the end of training. There is a small gap between the subgoal hit rates of OMG-conservative and OMG-optimistic, which leads to a longer episode length for OMG-optimistic than OMG-conservative, as illustrated in Figure 8(b). The root cause lies in the differences in the subgoal selection between OMG-conservative and OMG-optimistic. More details can be found in Appendix E.

6 Conclusion and Limitation

In this paper, we introduce OMG, a novel method for opponent modeling based on subgoal inference. OMG is a simple and efficient opponent modeling method and can be combined with various RL algorithms. Unlike most opponent modeling methods, which primarily focus on predicting the opponent's actions, OMG focuses on modeling the opponent's subgoals. Specifically, it leverages the value function of the policy to guide the selection of subgoals, which yields two variants of OMG for cooperative and general-sum games, respectively. Empirical results demonstrate the remarkable performance achieved by OMG, as compared to baselines based on action modeling, and that OMG exhibits better generalization when cooperating with opponents with unknown policies. We analyze the subgoals obtained by the inference model, and the results show they closely correlate with the opponent's trajectory. The limitation of OMG is it cannot handle open multi-agent systems where agents may enter and leave during the interaction. This is left for future work.

Acknowledgments

This work was supported by NSFC under Grant 62450001 and 62476008. The authors would like to thank the anonymous reviewers for their valuable comments and advice.

References

[1] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.

[2] Stefano V Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. *arXiv preprint arXiv:1506.01170*, 2015.

[3] Stefano V Albrecht and Peter Stone. Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems. *Artificial Intelligence*, 258:66–95, 2018.

[4] Stefano V Albrecht and Peter Stone. Reasoning about hypothetical agent behaviours and their parameters. *arXiv preprint arXiv:1906.11064*, 2019.

[5] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[6] Yakov Ben-Haim. *Info-gap decision theory: decisions under severe uncertainty*. Elsevier, 2006.

[7] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*, pages 1430–1440. PMLR, 2021.

[8] Shuo Chen, Ewa Andrejczuk, Athirai Aravazhi Irissappane, and Jie Zhang. Atsis: Achieving the ad hoc teamwork by sub-task inference and selection. *IJCAI*, 2019.

[9] Ashley Edwards, Himanshu Sahni, Rosanne Liu, Jane Hung, Ankit Jain, Rui Wang, Adrien Ecoffet, Thomas Miconi, Charles Isbell, and Jason Yosinski. Estimating q (s, s') with deep deterministic dynamics gradients. In *International Conference on Machine Learning*, pages 2825–2835. PMLR, 2020.

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

[11] Haobo Fu, Ye Tian, Hongxiang Yu, Weiming Liu, Shuang Wu, Jiechao Xiong, Ying Wen, Kai Li, Junliang Xing, Qiang Fu, et al. Greedy when sure and conservative when uncertain about the opponents. In *International Conference on Machine Learning*, pages 6829–6848. PMLR, 2022.

[12] Herbert Gintis. Modeling cooperation among self-interested agents: a critique. *The journal of socio-economics*, 33(6):695–714, 2004.

[13] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *International conference on machine learning*, pages 1802–1811. PMLR, 2018.

[14] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pages 1804–1813. PMLR, 2016.

[15] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[16] Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee. A Deep Policy Inference Q-Network for Multi-Agent Systems. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018.

[17] Jiechuan Jiang and Zongqing Lu. I2q: A fully decentralized q-learning algorithm. *Advances in Neural Information Processing Systems*, 35:20469–20481, 2022.

[18] Dong-Ki Kim, Miao Liu, Matthew Riemer, Chuangchuang Sun, Marwa Abdulhai, Golnaz Habibi, Sebastian Lopez-Cot, Gerald Tesauro, and Jonathan P. How. A Policy Gradient Algorithm for Learning To Learn in Multiagent Reinforcement Learning. In *International Conference on Machine learning proceedings (ICML)*, 2021.

[19] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

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

[21] Francisco S Melo and Alberto Sardinha. Ad hoc teamwork by learning teammates' task. *Autonomous Agents and Multi-Agent Systems*, 30:175–219, 2016.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518(7540):529–533, 2015.

[23] John F Nash et al. *Non-cooperative games*. Princeton University Princeton, 1950.

[24] Samer Nashed and Shlomo Zilberstein. A survey of opponent modeling in adversarial domains. *Journal of Artificial Intelligence Research*, 73:277–327, 2022.

[25] Georgios Papoudakis and Stefano V Albrecht. Variational Autoencoders for Opponent Modeling in Multi-Agent Systems. *arXiv preprint arXiv:2001.10829*, 2020.

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

[27] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019.

[28] Sujoy Paul, Jeroen Vanbaar, and Amit Roy-Chowdhury. Learning from trajectories via subgoal discovery. *Advances in Neural Information Processing Systems*, 32, 2019.

[29] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

[30] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[31] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine Theory of Mind. In *International Conference on Machine Learning (ICML)*, 2018.

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

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

[34] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

[35] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *CoRR*, abs/1902.04043, 2019.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[37] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[38] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR, 2019.

[39] Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1504–1509, 2010.

[40] Kefan Su and Zongqing Lu. A fully decentralized surrogate for multi-agent policy optimization. *Transactions on Machine Learning Research*, 2024.

[41] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[42] Andrea Tacchetti, H Francis Song, Pedro AM Mediano, Vinicius Zambaldi, Neil C Rabinowitz, Thore Graepel, Matthew Botvinick, and Peter W Battaglia. Relational forward models for multi-agent learning. *arXiv preprint arXiv:1809.11044*, 2018.

[43] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

[44] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *International conference on machine learning*. PMLR, 1993.

[45] Alexander Vezhnevets, Yuhuai Wu, Maria Eckstein, Rémi Leblond, and Joel Z Leibo. Options as responses: Grounding behavioural hierarchies in multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 9733–9742. PMLR, 2020.

[46] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[47] Jannis Weil, Johannes Czech, Tobias Meuser, and Kristian Kersting. Know your enemy: Investigating monte-carlo tree search with opponent models in pommerman. *arXiv preprint arXiv:2305.13206*, 2023.

[48] Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic Recursive Reasoning for Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2019.

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

[50] Xiaopeng Yu, Jiechuan Jiang, Wanpeng Zhang, Haobin Jiang, and Zongqing Lu. Model-based opponent modeling. *Advances in Neural Information Processing Systems*, 35:28208–28221, 2022.

[51] Zhang Zhang, Yifeng Zeng, Wenhui Jiang, Yinghui Pan, and Jing Tang. Intention recognition for multiple agents. *Information Sciences*, 628:360–376, 2023.

[52] Menghui Zhu, Minghuan Liu, Jian Shen, Zhicheng Zhang, Sheng Chen, Weinan Zhang, Deheng Ye, Yong Yu, Qiang Fu, and Wei Yang. Mapgo: Model-assisted policy optimization for goal-oriented tasks. *arXiv preprint arXiv:2105.06350*, 2021.

[53] Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann. Deep interactive bayesian reinforcement learning via meta-learning. *arXiv preprint arXiv:2101.03864*, 2021.

A Analysis of (s, g)

In opponent modeling, we usually build (s, g) and (s, a^{-i}) by observing the opponent's action trajectories. We construct a tree to describe the trajectories of the opponent's action sequences, as shown in Figure 9. The non-leaf nodes and edges represent the state and opponent's action respectively. Without loss of generality, we simplify the problem by using a complete tree with the leaf node as goal. The length of the action sequences is k and the opponent action space is denoted as A. We compare the number of (s, a) and (s, g) that can be observed via trajectories, and their sets are denoted as $\mathcal{S}_a$ and $\mathcal{S}_g$ respectively. The sizes of $\mathcal{S}_a$ and $\mathcal{S}_g$ as:

$$\begin{aligned} card(\mathcal{S}_a) &= \sum_{l=0}^{k-1} \sum_{s \in \mathcal{S}^{(l)}} n_A = \frac{n_A^k - 1}{n_A - 1} n_A \\ card(\mathcal{S}_g) &= \sum_{l=0}^{k-1} \sum_{s \in \mathcal{S}^{(l)}} \sum_{g \in G} \mathbb{I}(s \rightarrow g) \\ &\leq |G| + n_A \cdot \frac{|G|}{n_A} + \cdots + n_A^{k-1} \cdot \frac{|G|}{n_A^{k-1}} \\ &= k|G|, \end{aligned}$$

where $\mathcal{S}^{(l)}$ represents the set of all states of depth l in the tree. $s \rightarrow g$ means g is reachable from s. n_A is the size of A. Let $card(\mathcal{S}_g) \leq card(\mathcal{S}_a)$, we get a bound over $|G|$, as Equation (9). When the goal number of our method is within the bound, the number of expanded states can be significantly reduced, which means the RL algorithm learns faster than those methods based on action modeling.

$$card(\mathcal{S}_g) \leq card(\mathcal{S}_a) \Rightarrow |G| \leq \frac{n_A}{k} \frac{n_A^k - 1}{(n_A - 1)} = \frac{n_A}{k} |\mathcal{S}|. \tag{9}$$

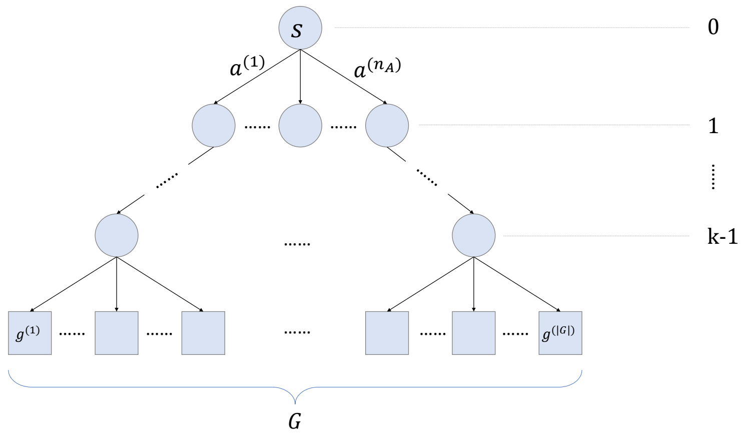


Figure 9: Illustration of opponent's decision tree. Circles, edges, and squares represent state nodes, action, and goal nodes respectively.

When $|G|$ is below n_A/k times the number of observed states, the goal-based opponent modeling method proves more advantageous compared to the methods based on action modeling. Consequently, this criterion can be met by maintaining a relatively modest value for k. Due to our method favoring the adoption of extreme values as goal states, a limited quantity of such states exist. So, it is loosely bound of $|G|$ for OMG.

B Experiments settings

Multi-Agent Environments. Foraging environment [2, 4] is an 8×8 gridworld with full observation, containing two players: the agent and the opponent. At the beginning of each round, the players and three foods are randomly generated in the environment. The goal of the agent is to collect all foods as quickly as possible. The agent can move in four directions or pick up the food. The agent must judge the opponent's target food as soon as possible to avoid futile actions for the same food.

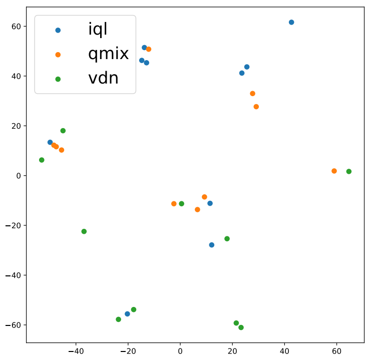
Predator-Prey [20] is a three-against-one multi-agent environment with a continuous space. Three predators coordinate to touch the prey, and all participants have full observation. The agent acts as one of the predators, and the opponents are the other two predators and the prey, which leads to the non-stationarity of the environment from the agent's view despite not belonging to one camp. The agent aims to maximize its reward and therefore needs to collaborate with the other two predators to complete the encirclement and cut the prey's escape route.

SMAC [35] is a high-dimensional partial observation complex environment for research in the field of collaborative MARL based on StarCraft II. The agent joins a set of opponents with unknown policies to accomplish the task. The only way to accomplish the task is to collaborate with the other agents. The agent's goal is to complete the task with a group of opponents controlled by unknown policies.

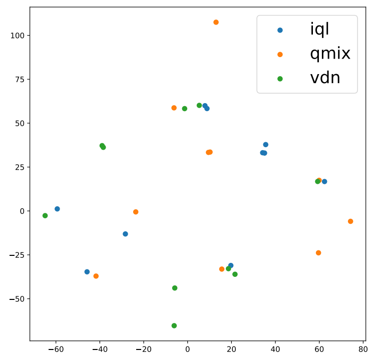
Opponent. The autonomous agent is trained in a multi-agent environment, where it interacts with the opponents controlled by a set of pre-trained policies. At the onset of each episode, the opponent's policy is selected randomly from the set. In the case of SMAC, the autonomous agent's index is also randomly determined. For Foraging, Predator-Prey, and SMAC environments, D3QN, PPO, and QMIX are used to train the opponents, respectively. All the opponents in the training set comprise 10 distinct policies.

In SMAC, the test set consists of 30 opponents with different policies, trained by the IQL, VDN [41], and QMIX [33]. In *8m*, the opponents are reorganized into three groups: *7 homologues*, *6 homologues*, and *7 non-homologues*. In *3s_vs_5z*, the opponents falls into two groups: *2 homologues* and *2 non-homologues*. Here, *homologue* refers to the policy from the same algorithm with the same parameters, and *non-homologue* represents the policy from two different algorithms. The remaining agent is controlled by OMG or baseline algorithms.

When assessing the performance of the autonomous agent in the SMAC with a test set, these opponents in the set are trained separately using IQL, VDN, and QMIX, with 10 instances for each training method. To illustrate the dissimilarity of the test opponent's policies, we utilize a set of identical states to acquire the action vectors of the policy in the test set. We visualize the action vectors, as demonstrated in Figure 10. The figure shows the diversity of test set policies employed by the test opponents. The test results are averaged over 100 episodes of fine-tuning, with 5 random seeds.



(a) 8m



(b) 3s_vs_5z

Figure 10: The distribution of the opponent's policy for the test of generalization.

Pre-train the subgoal's prior model. The subgoal's prior model $p_\psi(\bar{g}|s^g)$ is a VAE that learns from a set of states that are collected while training opponents. The optimization objective of VAE is :

$$< \hat{\omega}, \hat{\psi} >= \arg\max_{\omega,\psi} \mathbb{E}_{g\sim q_\psi(g|s)} \Big[\log p_\omega(s|g) \Big] - \mathrm{KL}\Big(q_\psi(g|s)||\mathcal{N}(0,1) \Big). \tag{10}$$

The decoder $p_\omega(s|g)$ is also used to reconstruct the subgoal state, as discussed in Section 5.5.

Hyperparameters. All hyperparameters are listed in Table 1.

Table 1: Hyperparameters

	Q-based RL	Foraging(D3QN)	SMAC(IQL)	**Policy-based RL**	Predator-prey(PPO)
RL Algorithm	hidden units	MLP[64, 32]	RNN[64, 64]	hidden units	MLP[64, 32]
	activation function	ReLU	ReLU	activation function	ReLU
	optimizer	Adam	RMSProp	optimizer	Adam
	learning rate	0.005	0.0005	learning rate	0.0005
	target update interval	100	200	num. of updates	10
	epsilon start	0.5	0	value discount factor	0.99
	epsilon end	0.95	0.95	GAE parameter	0.99
	epsilon anneal time	4500	50000	clip parameter	0.115
	batch size	32	32	max grad norm	0.5
	buffer size	5000	5000		
Opponent model	hidden units	MLP[64, 32]	MLP[64, 32]		MLP[64, 32]
	learning rate	0.001	0.001		0.001
	subgoal horizon H	5	10		5
	KL weight	0.001	0.001		0.001
	$\Delta\eta$	0.001	0.001		0.001
	ϵ start	0.5	0.5		0.5
	ϵ anneal time	50000	50000		50000

The computational resources for the experiments are as follows: the CPU is Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz, and the GPU is A100-PCIE-40GB.

C Performance of CTDE agent in autonomous agent task

The motivation of this paper is to address the autonomous agent through opponent modeling. The question is, can the CTDE agents be adapted to do tasks like autonomous agents? The two domain are fundamentally different, and the training method of CTDE doesn't work well in such situations because it hasn't been exposed to a variety of opponents during its training. We conducted tests in *8m* where QMIX acts as the agent with opponents of test set.

Table 2: Test performance on 8m

opponent type	7 non-homologue	6 homologue	7 homologue
QMIX	21.6%	65.6%	61.0%
OMG-optim	70.5%	86.8%	90.2%

This training paradigm employed by QMIX leads to a lack of generalization for different opponents. Using the same training methodology for QMIX as OMG leads to a degradation in IQL,which already serves as one of the existing baselines. The test results indicate that opponents trained using different methods and seeds are not homogeneous, which poses challenges for cooperation.

D s_g selection frequency

In Equation (6) and Equation (7), subgoals are selected from $\mathcal{N}_t^H = \{s_{t+k} | 1 \leq k \leq H\}$, where $\mathcal{N}_t^H$ is a sliding window. If the value is monotonic along the trajectory, a possible case is that different subgoals $\bar{g}$ are chosen at each step. We used 100 trajectories and counted the selection frequency within the trajectory, as shown in Table 3.

Table 3: The proportion of each s_{t+k} in $\mathcal{N}_t^H$.

k	1	2	3	4	5
proportion	35.4%	20.0%	22.8%	19.3%	2.5%

The results suggest that in multi-agent settings, characterized by cooperative and competitive interactions among agents, the value function displays multiple peaks along the trajectory. Additionally, the selected subgoals exhibit a certain level of continuity over several steps.

E Details for Inferred Subgoal Analysis

Subgoal hit ratio. Define the opponent's trajectory sequence from $t = 0$ to $t = T$ as $\mathcal{T} = (s_0, s_1, \dots, s_T)$, and the agent's prediction from time $t = 0$ to $t = T - 1$ as $\hat{\mathcal{T}} = (\hat{s}_0, \hat{s}_1, \dots, \hat{s}_{T-1})$. Let $\mathcal{T}_i = (s_i, s_{i+1}, \dots, s_T)$ represent the opponent's trajectory from step i onward.

The subgoal hit ratio is calculated as follows:

$$\text{subgoal hit ratio} = \frac{|\{\hat{s}_t \mid \hat{s}_t \in \mathcal{T}_t,\, t = 0, 1, \dots, T-1\}|}{|\mathcal{T}|}$$

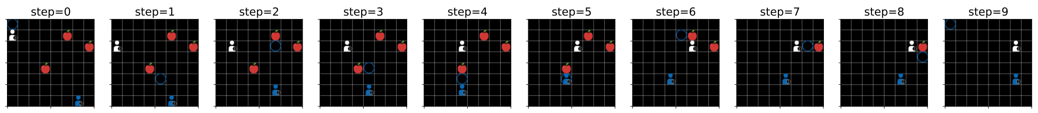
where $\{\hat{s}_t \mid \hat{s}_t \in \mathcal{T}_t,\, t = 0, 1, \dots, T-1\}$ represents the set of predicted states that are also present in the opponent's future trajectory starting from the current time step t.

For example, the opponent's trajectory is $\mathcal{T} = (s_1, s_2, s_3, s_4, s_5)$. For each time step, the relevant opponent trajectory is: $\mathcal{T}_0 = (s_1, s_2, s_3, s_4, s_5)$, $\mathcal{T}_1 = (s_2, s_3, s_4, s_5)$, $\mathcal{T}_2 = (s_3, s_4, s_5)$, $\mathcal{T}_3 = (s_4, s_5)$. The agent's predictions are $\hat{\mathcal{T}} = (s_3, s_1, s_5, s_5)$. The matched predicted states are $\{s_3, s_5\}$.

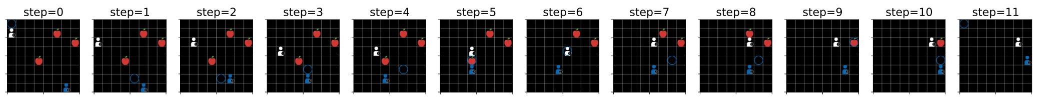
Thus, the hit ratio is calculated as:

$$\text{subgoal hit ratio} = \frac{|\{s_3, s_5\}|}{|\{s_1, s_2, s_3, s_4, s_5\}|} = \frac{2}{5} = 0.4$$

The complete trajectory of the example in Figure 8(b) is shown below:



(a) OMG-conserv



(b) OMG-optim

Figure 11: Illustration of inferred subgoal state

F OMG based on PPO

Algorithm 2 OMG based on PPO

```
1: ***Preparation:***
2: Interact with $\nu$ opponents to collect $s$ and train the prior model $f_\psi$
3: Initialize subgoal inference model parameters $\tau$ and $\theta$
4: Initialize policy parameters $\delta$ and value function parameters $\varphi$
5: **for** k=0,1,2,... **do**
6:     ***Interaction phase***
7:     Observe state $s$ and last opponent's action $a^{-i}$
8:     Infer the subgoal $\hat{g}$ by subgoal inference model $q_\phi(g|\tau)$
9:     Choose action $a$ by $\pi_{\delta_k}(\cdot|s,\hat{g})$
10:    Store experience $(s, a, a^{-i}, r)$ in buffer $\mathcal{D}_k$
11:    ***Update phase***
12:    Calculate prior subgoal $\bar{g}$ by Equation (6) or Equation (7)
13:    Calculate subgoal $g$ by Equation (8)
14:    Update policy parameters by
           $$\delta_{k+1} = \arg\max_{\delta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau\in\mathcal{D}_k} \sum_{t=0}^{T} \min\big(\frac{\pi_\delta(a_t|s_t)}{\pi_{\delta_k}(a_t|s_t)} A^{\pi_{\delta_k}}(s_t,a_t), g(\epsilon, A^{\pi_{\delta_k}}(s_t,a_t))\big) \quad (11)$$
15:    Update value parameters by
           $$\varphi_{k+1} = \arg\min_{\varphi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau\in\mathcal{D}_k} \sum_{t=0}^{T} (V_\varphi(s_t) - \hat{R}_t)^2 \quad (12)$$
16:    Update inference model $q_\phi$ and $p_\theta$ by Equation (5)
17: **end for**
```

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize the paper's contributions by outlining the OMG method, its application to cooperative and general-sum games, and its superior performance.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: See Section 6.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

Answer: [NA]

Justification: This paper does not have strict theoretical results. Appendix A offers example analysis, shedding light on the insights of the method presented in this paper, though it has not yet advanced to the theoretical level.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: The algorithm is described in detail as Algorithm 1 and Algorithm 2. The code is provided in the supplemental material, and all hyperparameters and environment Settings are listed in the Appendix B.

   Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The code is provided in the supplemental material, and all hyperparameters and environment Settings are listed in the Appendix B.

   Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: Details of the environment and experimental setup are detailed in the Appendix B.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: All the methods are trained for five runs with different random seeds, and results are presented using mean and standard deviation. This is mentioned in first paragraph of Section 5.

   Guidelines:

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

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: See Appendix B.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: After careful comparison with the NeurIPS Code of Ethics, we were confident that no guidelines were violated.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: [NA]

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All baselines and environments are cited with the original paper.

    Guidelines:

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

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

13. **New Assets**

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

    Answer: [NA]

    Justification: [NA]

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: [NA]

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.