Dynamic Belief for Decentralized Multi-Agent Cooperative Learning

Yunpeng Zhai[1] , **Peixi Peng**[1,2,*] , **Chen Su**[3] and **Yonghong Tian**[1,2,3,*]
[1]National Key Laboratory for Multimedia Information Processing, School of Computer Science, Peking University, Beijing, China
[2]Peng Cheng Laboratory, Shenzhen, China
[3]School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen, China
{ypzhai, pxpeng, suchen, yhtian}@pku.edu.cn

Abstract

Decentralized multi-agent cooperative learning is a practical task due to the partially observed setting both in training and execution. Every agent learns to cooperate without access to the observations and policies of others. However, the decentralized training of multi-agent is of great difficulty due to non-stationarity, especially when other agents' policies are also in learning during training. To overcome this, we propose to learn a dynamic policy belief for each agent to predict the current policies of other agents and accordingly condition the policy of its own. To quickly adapt to the development of others' policies, we introduce a historical context to learn the belief inference according to a few recent action histories of other agents and a latent variational inference to model their policies by a learned distribution. We evaluate our method on the StarCraft II micro management task (SMAC) and demonstrate its superior performance in the decentralized training settings and comparable results with the state-of-the-art CTDE methods.

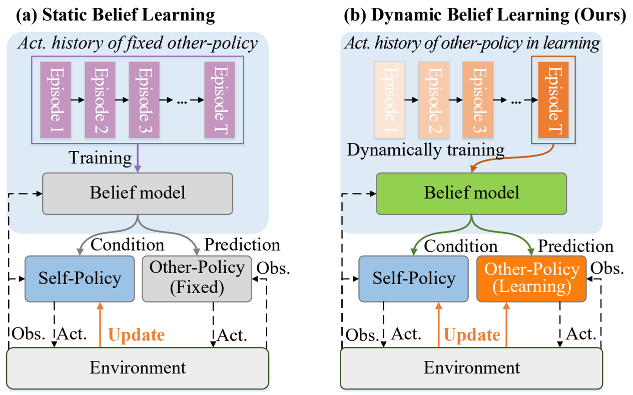

Figure 1: (a) Static belief learning used by previous works assuming fixed policies of other agents cannot apply to the decentralized learning systems where other-policy is updating simultaneously. (b) Our proposed dynamic belief learning tackles the decentralized scenarios by learning to model other agents over their changing policies dynamically from only a few recent historical steps.

1 Introduction

Many complex sequential decision-making problems [Chen *et al.*, 2017; Foerster *et al.*, 2018] require all agents to achieve a unified goal or the largest team utility in a decentralized way, where all agents coordinate their behaviors conditioned only on their own observation history. Due to the high-dimensional dynamic state space and unknown environment model, deep multi-agent reinforcement learning (MARL) shows great potential and has attracted a lot of interest in these years.

To learn decentralized policies, a typical baseline [Tan, 1993; de Witt *et al.*, 2020] is to develop an independent learner for each agent and regard the collective (or global) rewards as the individual rewards directly. This paradigm may be faced with the non-stationarity [Bowling and Veloso, 2002] problem: the dynamics of its environment effectively changes as allies (or teammates) change their behaviors through learning and agents may receive spurious reward signals that originate from their allies' behavior. To make the learning stable, most of the existing methods typically adopt the paradigm of centralized training and decentralized execution (CTDE) [Rashid *et al.*, 2018], which assume the learning occurs in a laboratory or a simulator where the extra global state and communication are available. Despite the great progress it has made, the CTDE paradigm still may be limited in some realistic multi-agent systems due to the difficulty of developing enough real simulators and the inaccessibility of the global states or agents' communication for training. Take the autonomous vehicles [Cao *et al.*, 2012] for example. Even though RL models have been deployed on vehicles in the factory before delivery, they still require further learning in the real road environment where every vehicle is independent without centralized scheduling. Therefore, it calls for a more practical decentralized approach with partially observed setting both in training and execution, where every agent learns to cooperate only by its own observation without global information as well as the observation and policies of others.

To develop a stable decentralized training method, existing independent learning handles the non-stationarity by elabo-

*Corresponding author.

rating training practices like clipping PPO policy ratio [Sun *et al.*, 2022] and repeating Q-learning updates[Abdallah and Kaisers, 2016], or improving experience replays[Palmer *et al.*, 2018][Lyu and Amato, 2018] . Different from them, our method handle the problem in an other-agent-modeling manner, in which latent behaviors of the other agents is considered for learning. Related to our work, some methods of agent modeling [Hernandez-Leal *et al.*, 2019] [Hong *et al.*, 2017] learn a static policy belief model to predict and respond to the actions of other agents. However, they assume fixed policies of other agents and fail to apply on the decentralized multi-agent learning in which the policies of other agents continue updating by learning itself, as illustrated in Fig. 1. In this situation, a desirable capability of agents is to dynamically change their policy beliefs over the development of others' policies. A naive way for the purpose is to fine-tune the policy belief model only utilizing the most recent action histories, considering that the histories in more previous episodes are out-of-date to the current policies of others. However, such approaches may suffer a lot from the lack of data and insufficient training and thus prevent the policy belief from modeling others effectively. In the subsequent part of this paper, **belief** refers to **policy belief** for clarity.

In this paper, we propose to learn a dynamic belief for dynamically modeling other agents over their changing policies in learning and accordingly condition its policy. To deal with the developing policies of other agents, we meta-train the dynamic belief to learn to model other agents by only a few action histories in the most recent episodes, as illustrated in Fig. 1(b). Our learned belief model of each agent takes as input: (i) the current observation, and (ii) a set of recent observations and observed actions of other agents. Then it outputs an embedding of belief to be used by prediction of the future actions of others. Specifically, we introduce a historical context to dynamically process the recent histories relevant to the current state by essential use of soft attention. And then, we present a latent variational inference to produce the belief embedding by modeling policies of others as a distribution with VAE [Pu *et al.*, 2016]. Our method is implemented based on the Proximal Policy Optimization (PPO) [Schulman *et al.*, 2017] structure, which has shown impressive success in many reinforcement learning tasks. Trained by predicting actions of other agents, the belief embedding of an agent then conditions both the policy decision and the value computation in the training as well as the execution stages. Note that our method is not to seek the optimal policy of an agent based on the policies of other agents, but to learn effective policies of all agents simultaneously under decentralized learning framework. We take belief as an additional input when optimizing the policy function, which avoids the instability caused by recursive predictions.

In summary, our contribution is as follows: (i) We proposed a novel dynamic belief learning to alleviate the non-stationarity of decentralized multi-agent learning by modeling the changing policies of teammate agents and conditioning the policy of its own. (ii) It introduces a historical context and a latent variational inference which learn to dynamically model other agents by only a few action histories in the most recent episodes. (iii) Experiment results on two cooperative environments demonstrate that our approach improves the performance over other independent learning methods in decentralized settings and even achieves a comparable performance to CTDE.

2 Related Work

Multi-agent cooperative learning. Although deep RL has been well explored using centralized controllers, those methods designed for single agents cannot be applied for multi-agent RL tasks since the joint action space grows exponentially with agent numbers. Hence, centralized training and decentralized execution (CTDE) was proposed as a popular paradigm for MARL [Kraemer and Banerjee, 2016]. Typically, value-based approaches learn the global state-action values as the aggregation of individual state-action values. For instance, VDN [Sunehag *et al.*, 2017] and QMIX [Rashid *et al.*, 2018] produce the global Q-values by an arithmetic summation or a non-linear monotonic factorization. Other works further extend the class of value functions, *i.e.*, QTRAN [Son *et al.*, 2019]. Another typical approach, Actor-Critic MARL, learns independent actors by a centralized critic using policy gradients, such as MAPPO[Chao *et al.*, 2021][Sun *et al.*, 2022], COMA[Foerster *et al.*, 2018], MADDPG [Lowe *et al.*, 2017] and so on. Despite the significant progress that has been made, CTDE may still be limited in some realistic multi-agent tasks due to the requirement of centralized training. Therefore, the practical application calls for a decentralized approach with partially observed settings both in training and execution. Under this limit, independent learning (IL) like IPPO[de Witt *et al.*, 2020], IQL [Tan, 1993] and IAC [Foerster *et al.*, 2018] decompose an n-agent MARL problem into n decentralised single-agent problems and directly learn decentralized policies for each agent. However, they often suffer from the non-stationarity of the environment induced by agents simultaneously learning and exploring, making them unable to learn optimal policies in some environments. In this paper, we explore a different potential to alleviate non-stationary by learning and exploiting a dynamic belief of others.

Other agent modeling. Modeling and adapting to unknown other agents is crucial for multi-agent environments, where agents interact simultaneously. With access to opponents' trajectories, learning other agent adaptively is widely used in multi-agent competitive tasks [He *et al.*, 2016] or human-ai coordination [Carroll *et al.*, 2019]. ToM [Rabinowitz *et al.*, 2018], ToMoP [Yang *et al.*, 2019] consider a purely observational setting and predicted other agents' desires, beliefs, and intentions from the observed state and action. InAC[Ma *et al.*, 2021a] learns interactions in a centralized manner. Furthermore, several methods [Papoudakis and Albrecht, 2020] [Zintgraf *et al.*, 2021] model others' polices as a latent distribution by variational autoencoder (VAE) [Pu *et al.*, 2016]. Differently, our work assumes partially observed training settings with only access to the observation of its own. PR2[Wen *et al.*, 2019], GR2[Wen *et al.*, 2021] learn recursive reasoning of others to seek optimal responding while our method is to learn effective policies of all agents simultaneously. NRBS[Moreno *et al.*, 2021] learns recursive belief for policy training but ignores relative historical observations. OP[Ma *et al.*, 2021b] collects and matches the behavioral graph of opponents directly as policy inputs, whereas our work mines historical samples to learn belief by action prediction. Related to our work, previous methods [Hernandez-Leal *et al.*, 2019] [Hong *et al.*, 2017] model other agents by learning a static belief to predict their future actions assuming fixed policies of other agents, similar to imitation learning [Duan *et al.*, 2017]. However, such methods fail to apply to the decentralized learning setting in this work, where agents change their policies by learning simultaneously. MeLIBA [Zintgraf *et al.*, 2021] assumes a prior distribution over others' policies but still can't tackle our settings without prior distributions. For this situation, our method dynamically changes the belief over the development of others' policies by the recent historical observations and effectively helps the learning of policy.

Transformer in RL. Transformer [Vaswani *et al.*, 2017] has been employed in many recent RL methods [Janner *et al.*, 2021] [Parisotto *et al.*, 2020] [Banino *et al.*, 2021] [Meng *et al.*, 2021]. A typical line of methods used a transformer to replace GRU/LSTM for modeling temporal information over the sequence of states [Chen *et al.*, 2021] [Parisotto *et al.*, 2020] [Banino *et al.*, 2021], while another line of methods employed the architecture among multiple agents to exploit the inherent relation across agents for multi-agent scenarios [Meng *et al.*, 2021] [Hu *et al.*, 2020] [Inala *et al.*, 2020]. Different from them employing transformers to choose actions, our work introduces it to learn the belief about other agents by mining historical samples. Significantly, the attention in our work is applied to different objects from them, neither between different states in the same episode for temporal relations nor between different agents in the same state for inherent interactive impacts. It is between the current state and those in historical episodes to explore the relative samples as references for dynamic belief inference.

3 Background

Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs). [Oliehoek and Amato, 2016] Consider a *fully cooperative multi-agent task* that can be described as a stochastic game G, defined by a tuple $G =< S, U, P, r, Z, O, n, \gamma >$. $s \in S$ describes the true state of the environment. At each time step, each agent $a \in A \equiv \{1, ..., n\}$ chooses an action $u^a \in U$, forming a joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$, which induces a transition in the environment according to the state transition function $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \rightarrow [0, 1]$. All agents share the same reward function $r(s, \mathbf{u}) : S \times \mathbf{U} \rightarrow \mathbb{R}$ and $\gamma \in [0, 1)$ is a discount factor. This paper considers a *decentralized multi-agent learning* setting in both training and execution. Every agent is trained individually without sharing policy model or replay buffer with each other. We consider a *partially observable* scenario in which each agent draws individual observations $z \in Z$ according to observation function $O(s, a) : S \times A \rightarrow Z$. Each agent has an action-observation history $\tau^a \in T \equiv (Z \times U)^*$, on which it conditions a stochastic policy $\pi^a(u^a|\tau^a) : T \times U \rightarrow [0, 1]$. MARL agents aim to maximize the discounted return $R_t = \sum_{l=1}^{\infty} \gamma^l r_{t+l}$. Throughout this paper, we omit the superscript a to represent the current agent for clarity. Quantities in bold represent joint quantities over agents, i.e., $\mathbf{u}$, and bold quantities with a tilde such as $\tilde{\mathbf{u}}$ denote joint quantities over agents other than the current one.

Action-observable Setting. Our work assumes that actions of the observed other agents are observable, which is practical and reasonable in realistic applications. Taking autonomous vehicles for example, the moving and steering actions of other vehicles can be easily observed.

Proximal Policy Optimization (PPO). The PPO algorithm [Schulman *et al.*, 2017] is one of the most popular single-agent RL methods due to its advanced performance, stable training, and easy implementation. PPO learns a policy function π_θ with parameter θ and a value function $V_\phi(s)$ with parameter ϕ, where $V_\phi(s)$ is used for variance reduction and only utilized during training. To optimize π_θ, PPO imposes the policy constraint through a clipped surrogate objective function:

$$\mathcal{L}^{CLIP}(\theta) = \mathbb{E}\left[\min(r_\theta A, \text{clip}(r_\theta, 1-\epsilon, 1+\epsilon)A)\right], \quad (1)$$

where $r_\theta = \frac{\pi_\theta(u|s)}{\pi_{\theta_{old}}(u|s)}$, θ_{old} is the previous parameter in rollout. and ϵ is the clipping bound. Noting that PPO is an actor-critic algorithm, the policy π is the actor and the advantage function A is the critic. The advantage function represents how good a state-action pair is compared with the average value of current state, i.e., $A(s, u) = Q(s, u) - V(s)$, where $Q(s, u)$ is the action-value function estimated by samples and $V(s)$ is the approximation of the state-value function. PPO uses the generalized advantage estimation (GAE) to compute the advantage, which uses the linear combination of n-step bootstrapping to obtain low bias and low variance.

Independent PPO (IPPO). In MARL settings *Independent PPO (IPPO)* [Chao *et al.*, 2021] decomposes a multi-agent problem into a collection of simultaneous single-agent problems that share the same environment. Each agent learns policy π_θ and state-value function V_ϕ of itself independently without sharing information such as observations or replay buffers. This approach considers other agents as a part of the environment and thus suffers from the non-stationarity of training,

$$P(s'|s, u, \tilde{\mathbf{u}}) \neq P(s'|s, u, \tilde{\mathbf{u}}'), \quad (2)$$

due to the changing policies of other learning agents, $\tilde{\mathbf{u}} \neq \tilde{\mathbf{u}}'$. Therefore, it could not guarantee convergence even in the limit of infinite exploration.

4 Methods

Since each agent receives the joint reward responding to the actions of all agents, it is hard to learn $\pi(u_t|o_t)$ from $r(s_t, u_t, \tilde{\mathbf{u}}_t)$ directly because the uncertainty of other agents' actions $\tilde{\mathbf{u}}_t$. To handle this challenge, we introduce belief δ_t as an embedding vector corresponding to $\tilde{\mathbf{u}}_t$, and aim to learn $\pi(u_t|o_t, \delta_t)$ alternatively. Note the state transition relies on all agents' actions in Dec-POMDP, and the observation of each agent belongs to the global state. Hence, the agent's observation history τ_t is related with other agents' actions, especially

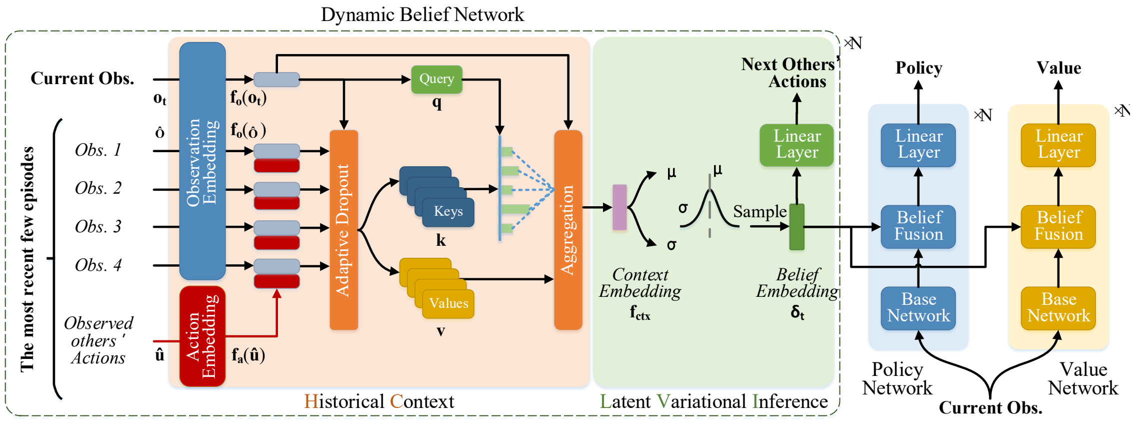

Figure 2: The framework of the proposed dynamic belief learning. **(Left)** The dynamic belief network consists of two components: historical context which dynamically learn policies of other agents by utilizing soft attention over the observation(Obs.) and others' actions of a few time steps in the most recent episodes, and latent variational inference which produces an embedding of belief by VAE and uses it for the prediction of the future actions of the teammates. **(Right)** The policy and the value networks are then conditioned on both the belief embedding and the current observation to learn optimal policies.

which are close with current agent. Hence, δ_t could be generated by τ_t, and further be used to predict $\tilde{\mathbf{u}}_t$. To effectively learn δ_t where other agents learn simultaneously, we first introduce the dynamic belief learning for dynamically modeling other agents over their changing policies, and then present how to extend the single agent RL method PPO to Belief PPO for decentralized multi-agent cooperative learning.

4.1 Dynamic Belief Learning

In principle, with the progress of learning, the previous interaction history can no longer reflect the policy information of other agents, except the most recent ones because of the continuity of learning. To this end, we propose dynamic belief learning, which meta-trains a model to learn to predict belief only by the action histories of other agents in the most recent episodes. In each step, our dynamic belief network receives as input (i) the current observation, and (ii) historical observations and the observed actions of other agents in the recent episodes. Then the dynamic belief network produces an embedding of belief to be used by predicting the future actions of the teammates and the policy of itself. As illustrated in Fig. 2, the proposed architecture is composed of two modules: the historical context and the latent variational inference, where the former is used to learn context embedding from historical steps in the recent episodes and the latter presents the belief δ_t^a by sampling from a learnt latent distribution.

Historical Context

Historical context utilizes soft attention over the most recent historical steps to dynamically learn belief. Firstly, an observation embedding network f_o and an action embedding network f_a are introduced. They receive an observation or a one-hot encoded action as input and produce an embedding of the observation or the action. In each step t, the current observation o_t and the historical observations $\{\hat{o}_{\hat{t}}, \hat{t} = 1, ..., T_h\}$ are encoded by embedding network f_o to produce a current observation embedding $f_o(o_t)$ and a sequence of historical observation embedding $\{f_o(\hat{o}_{\hat{t}}), \hat{t} = 1, ..., T_h\}$ respectively, where $\hat{t}$ is the time step in the history and T_h is the length of the used history. For dynamic belief inference, the input history can span hundreds to thousands of time steps, and training with such long sequences for every current step can be demanding in both time and memory usage. Considering that most historical steps are far irrelevant to the current states, we use an adaptive dropout operation to discard a subset of irrelevant time steps during both training and execution. The adaptive dropout depends on the cosine similarity $\mathcal{S}$ between the sequence of historical observation embedding and the current observation embedding,

$$\mathcal{K}_{o_t} = \underset{\{\hat{t}_i\}_{i=1}^{N_h} \subseteq [1, T_h]}{\arg\min} \sum_{i=1}^{N_h} \mathcal{S}(f_o(o_t), f_o(\hat{o}_{\hat{t}_i})), \tag{3}$$

where $\mathcal{K}_{o_t}$ represents the subset of top-N_h historical steps selected specifically for the current observation o_t. We denote p as the proportion of time steps that are preserved, and $N_h = pT_h$.

After downsampling the historical steps, the historical context processes embedding of both the current state and historical states, including observations as well as the observed others' actions, and outputs a context embedding. Hence, it is forced to capture only the relevant information, which will be used to produce the belief embedding and predict the actions of others. To this end, we make use of soft attention between the current state and historical states. The module starts by computing a query vector $\mathbf{q}(o_t)$ as a function of the current state embedding $f_o(o_t)$, which is then used to attend over the different states in the most recent histories. To capture sufficient information of the historical states, we compute the state embedding of historical steps as the aggregation of the observation embedding and the action embedding by element-wise adding $f_s(\hat{s}_{\hat{t}}) = f_o(\hat{o}_{\hat{t}}) + f_a(\hat{\tilde{u}}_{\hat{t}})$. The action embedding is produced by a linear layer on the one-hot encoded action $\hat{\tilde{u}}_{\hat{t}}$ of other agents. Then the key vector $\mathbf{k}(\hat{s}_{\hat{t}})$ and the value vector $\mathbf{v}(\hat{s}_{\hat{t}})$ are computed for every historical state as different functions of the historical state embedding. The context embedding is aggregated by adding the current state embedding and the weighted summary of the value vectors over different historical states,

$$\mathbf{f}_{ctx}(o_t) = f_o(o_t) + \sum_{i=1}^{N_h} w_i \mathbf{v}(\hat{s}_{\mathcal{K}_{o_t}[i]}) \tag{4}$$

where w_i is the attention weight computed by the scaled inner product between the query vector and the key vectors: $w_i =< \mathbf{q}(o_t), \mathbf{k}(\hat{s}_{\mathcal{K}_{o_t}[i]}) >$.

Latent Variational Inference

Since the policies of other agents are not fixed, we assume other agents' policies as a Gauss distribution $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t)$ where $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$ represent the mean and variance respectively. Therefore, an encoder function is introduced following the soft attention over observation module to predict $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$ respectively, and then the belief is sampled from $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t)$. To make the sampling derivable to $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$, the re-parameter trick [Kingma and Welling, 2013] is used:

$$\delta_t = \boldsymbol{\mu}_t + \sigma_t \times \delta_I, \tag{5}$$

where δ_I is sampled from the standard normal distribution.

Based on δ_t, a decoder function composed by a linear layer is used to predict teammates' actions. Consider the observed teammate's actions $\tilde{\mathbf{u}}_t$ as the sampled datapoints and the learning objective could be formulated as maximizing the likelihood function $\max log(p(\tilde{\mathbf{u}}_t))$. However, $log(p(\tilde{\mathbf{u}}_t))$ could not be optimized directly. Similar to ELBO [Kingma and Welling, 2013], its lower bound is:

$$log(p(\tilde{\mathbf{u}}_t)) \geq \mathbf{E}_{\delta_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t)}[p(\tilde{\mathbf{u}}_t|\delta_t)] - \mathcal{D}_{KL}(p(\delta_t|\mathbf{f}_{ctx})||Q(\delta_t)), \tag{6}$$

where $\mathcal{D}_{KL}$ is the KL-divergence and $p(\delta_t|\mathbf{f}_{ctx}) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t)$. $Q(\delta_t)$ is the prior probability distribution and defined as the standard normal distribution in our method. Hence, maximizing $log(p(\tilde{\mathbf{u}}_t))$ could be translated to maximizing the lower bound. Specifically, maximizing $p(\tilde{\mathbf{u}}_t|\delta_t)$ is equal to minimizing the mean square error:

$$\|\mathbf{v}_t(\tilde{\mathbf{u}}_t - \tilde{\mathbf{u}}_t^p)\|_2, \tag{7}$$

where $\tilde{\mathbf{u}}_t^p$ is the predicted actions of other agents and $\mathbf{v}_t \in \{0,1\}^{n-1}$ is the visibility for other agents obtained from the environment for ignoring the prediction error of the invisible teammates.

4.2 Belief PPO

Here, we introduce how the proposed dynamic belief learning can be incorporated into the PPO algorithm and achieve effective decentralized multi-agent cooperative learning. We follow the algorithmic structure of PPO by learning a policy network π_θ and a state-value function network $V_\phi(s)$ for each agent. Since making decisions only by the individual observation of an agent will lead to non-stationarity of training, we make the policy and the value function condition on both the observation and the learned belief embedding about others. The policy and the value function share the same belief embedding within each agent for information reuse and robust training. Specifically, in each step t, the policy network starts by processing the current observation o_t with a base network and then combines the information of the current state and the inferred belief through a belief fusion module. In our experiments, the belief fusion is implemented by concatenating the belief embedding from the dynamic belief network and the state embedding from the base network and then applying to it a linear transformation. After that, the final policy with the size of action space is produced by another linear layer. The state-value function network is of the same structure as the policy but has a different output size of 1. During training, the policy network, the value function network, and the dynamic belief network are optimized simultaneously by the PPO loss and belief loss according to Eq. 1 and Eq. 6,

$$\mathcal{L} = \mathcal{L}^{CLIP}(\theta) + \alpha\|\mathbf{v}_t(\tilde{\mathbf{u}}_t - \tilde{\mathbf{u}}_t^p)\|_2 + \beta\mathcal{D}_{KL}(p(\delta_t|\mathbf{f}_{ctx})||Q(\delta_t)), \tag{8}$$

where α and β are the weight factors of MSE and KL-divergence, respectively.

5 Experiments

We evaluate our approach on a fully cooperative environment SMAC [Samvelyan *et al.*, 2019], a standardized decentralized StarCraft II micromanagement environment. In each scenario, algorithm-controlled independent ally units fight against enemy units controlled by the built-in game AI. And an episode of game is declared a victory only if all enemy units are eliminated. In the decentralized setting, each agent can only achieve partial observation within its sight range, including location, health, and actions taken by visible agents. Any global information used by CTDE methods is unused.

Details. During training, 8 paralleled episodes are rolled out independently to generate data. The most recent 800 steps before last optimization are used for reference histories in dynamic belief network. And p is set to 0.05 for the adaptive dropout. The loss weights $\alpha = 0.5$ and $\beta = 0.005$.

5.1 Comparison with Other Methods

Comparison with Decentralized Methods. We compare the overall results of our approach with three independent learning approaches: IPPO, IQL, and IAC in the same decentralized settings. Note that our implementation of IPPO is equivalent to SOTA independent method PG-ind[Fu *et al.*, 2022] in ICML2022. The results in three maps are illustrated in Fig. 3(a-c). Independent learning methods like IPPO, IQL, and IAC fail to learn policies that consistently defeat the enemy, especially in the map of 3s_vs_5z. Moreover, the performance over training steps is volatile due to the non-stationarity of the environment as other agents learn to change their policies during training. However, our approach achieves far higher performance than those methods in the same decentralized settings without sharing information. The superior results demonstrate that our approach alleviates the disadvantages of non-stationarity by the dynamic belief, which effectively anticipates the actions of others.

Comparison with CTDE Methods. The state-of-the-art CTDE methods that train agents in a centralized way are also compared, including MAPPO, QMix, COMA, RODE[Wang

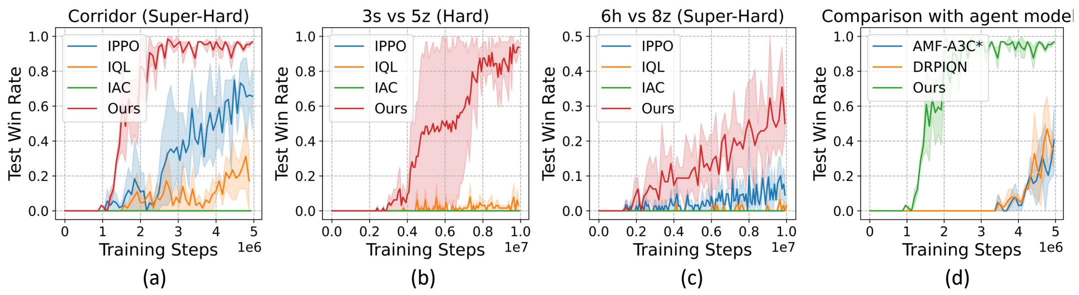

Figure 3: **(a-c) Results on the decentralized settings.** Win rates of our method and other independent learning approaches on a range of SC mini-games including one hard and two super-hard maps. **(d) Comparison with agent modeling methods on corridor.** * denotes that the method is implemented based on the better PPO algorithm. Best viewed in color.

Maps	Ours	MAPPO	QMix	COMA	RODE	QPLEX
2c vs. 64zg	**100.0**	**100.0**	**100.0**	10.0	**100.0**	90.6
3s vs. 5z	**100.0**	**100.0**	**100.0**	0.0	78.9	98.4
3s5z	**96.9**	**96.9**	88.3	6.25	93.75	96.8
6h vs. 8z	48.4	**88.3**	84.0	0.0	78.1	1.5
corridor	**100.0**	**100.0**	**100.0**	0.0	90.6	0.0

Table 1: Comparison of **Win Rate** with state-of-the-art CTDE methods. Our method achieves comparable performance with most of them. Since our method is under decentralized learning, the comparison is only for reference. The best results are highlighted with bold fonts.

Method	Success (%)
Baseline	73.44
Static Belief	71.87
Belief with LVI	75.00
Dynamic Belief	85.94

Table 2: Ablation studies on a non-fully cooperative environment: Traffic Junction (medium level) in the decentralized setting. *LVI* denotes the latent variational inference.

et al., 2020b] and QPLEX[Wang *et al.*, 2020a]. Among them, MAPPO with feature-pruned global states and the fine-tuned QMix [Hu *et al.*, 2021] with elaborate optimizations perform the best in SMAC. As shown in Table 1, these methods perform well due to extra global state information and centralized training to address non-stationarity. Nevertheless, our approach achieves comparable and even superior performance with them in most scenarios, which indicates its capability to effectively utilize the limited partial information. However, our method performs not well in 6h vs. 8z for two reasons: **1)** Global states are significant for such complex scenario while our method dose not use them. Specifically, the controlled units in 6h_vs_8z are much weaker (with lower armor and slower speed) than enemies compared with other maps, which is difficult and demands more cooperation among agents. With the help of the extra global state and agents' communication (i.e., the share of model and replay buffer), MAPPO learns better cooperation and outperforms our method. Under the decentralized setting, MAPPO is equivalent to IPPO and is outperformed by us with a large margin as shown in Fig. 3, which demonstrates that the superiority of MAPPO to our method is heavily attributed to the global states and agent's communication. **2)** Since the scenario is hard to explore[Hu *et al.*, 2021], our decentralized setting is limited by lower data efficiency, about only 1/6 of CTDE, because only the trajectories of an agent's own are used for training. As shown in Fig. 4, our method achieves comparable performance eventually as the training steps increase. And for other easy-to-explore scenarios such as corridor, our method achieves faster improvement, demonstrating its superiority in learning ability without the limitation of exploration. Moreover, the approach is more practical without centralized restriction.

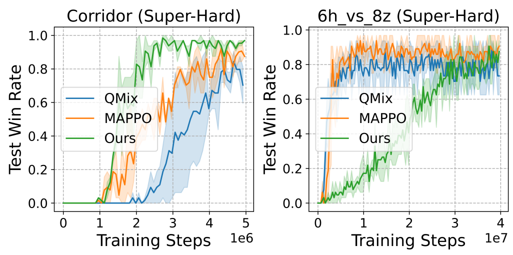

Figure 4: Training curves of CTDE methods and ours on SMAC. The comparison is only for reference due to our different decentralized settings. Our method needs more training steps in hard-to-explore scenarios like 6h_vs_8z due to the lower data efficiency of the decentralized setting.

Comparison with Agent Modeling. We compare our method with related agent modeling methods: DRPIQN [Hong *et al.*, 2017] and AMF-A3C[Hernandez-Leal *et al.*, 2019], which also learn networks to predict and respond to teammates' actions. Specifically, we re-implemented AMF-A3C based on a better baseline structure of PPO. As shown in Fig. 3(d), both the methods perform not well because they learn a static belief designed for fixed teammates and thus fail to tackle the decentralized learning where other agents also change their policies during training. Our dynamic belief outperforms them by large margins, thanks to quickly adapting to the development of teammate policies.

5.2 Ablation Studies

Component effects. To validate the effect of each component, ablations are studied in detail on the environment of the

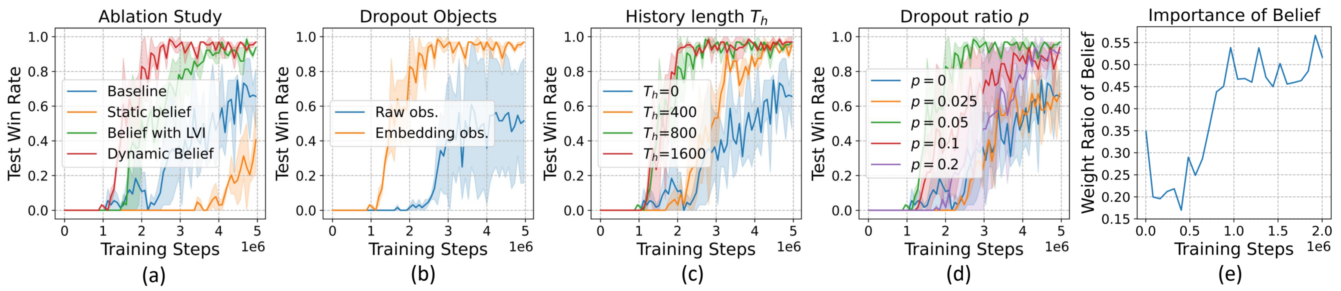

Figure 5: (a) Ablation studies on SMAC corridor map demonstrating the effect of components of the dynamic belief learning. *LVI* denotes the latent variational inference. (b) Comparison of using raw observation and embedding observation for the dropout approach. (c-d) Parameter analysis for the history length and ratio p of the Top-N. (e) The ratio of sum of weights for belief to all weights in the belief fusion network.

SMAC corridor map, as shown in Fig. 5(a). With IPPO as *baseline*, we first evaluate the *static belief* by directly learning an embedding that predicts the actions of others and then conditions the policy. As shown in Fig. 5(a), the performance drops distinctly compared to the baseline due to it being incapable of predicting the changing policies of other agents. Then we evaluate the latent variational inference (LVI) by applying it to learn a distribution of latent variables for the belief, denoted as *Belief with LVI*. The experiment improves the performance of static belief and surpasses the baseline, demonstrating the effect of LVI to process the uncertainty of others' policies and alleviate adverse effects of mispredictions. Furthermore, we validate the effectiveness of the historical context. With the context module, the overall *dynamic belief* achieves a rapidly growing win rate and outperforms the *Belief with LVI* by large margins. The improvements can be explained by the soft attention, which helps dynamically predict the updating policies of others. It effectively learns to infer their policies only from a few historical steps in the most recent episodes. We also validate the *embedding observation* for dropout by replacing it using *raw observation* as shown in Fig. 5(b), which fails to learn effective policy. Our method calculates the distance of historical observations on their embedding because the raw observation space is not discriminative, which directly contains the values of health, position and ignores importance of every element. And the embedding network automatically learns to project the raw observation into a latent space, where semantically similar observations have smaller distance. Since the embedding network is only a 2-layer MLP, the additional computational cost is low and affordable.

Generalization. To validate the generalization of the method, we evaluate it on a non-fully cooperative environment traffic junction [Sukhbaatar *et al.*, 2016], where each agent needs to maximize independent rewards by coordinating with others. In the traffic junction, cars enter a junction from all entry points with a probability of p_{arr}. Every car can take two actions at each time-step, *gas* and *brake* respectively. For traffic junction, an episode is considered a failure if a collision happens. The **success rate** refers to the percentage of episodes without failure. We adopt the medium level of difficulty as described in [Sukhbaatar *et al.*, 2016]. As shown in Table 2, consistently improved results can also be observed in the traffic junction environment. It demonstrates the general effectiveness of the proposed methods for different environments in decentralized settings.

Parameter analysis. Ablations of history length T_h in historical context and preservation ratio p in the adaptive dropout are evaluated in Fig. 5(c-d). The results show that a larger T_h is likely to converge faster and perform better, while smaller T_h can still achieve comparable results eventually with less computation. We use $T_h = 800$ for both performance and complexity. Ablations of p (for Top-N) show that the method cannot learn effective policies when p is too small which abandons useful histories, while it may converge slowly when p is too large. The method performs best when $p = 0.05$.

Importance of belief over training. To validate the importance of belief over training steps, we evaluate the sum of weights in the belief fusion network, which takes as inputs both the belief embeddings and current observation. The sum ratio of weights for belief to all weights (for both belief and current observation) is shown in Fig. 5(e). The ratio increases after an initial decrease, indicating that the belief has less influence on policy at the beginning and becomes more important to it over training. Specifically, when the teammates take random actions in the initial learning, the dropout approach is hard to preserve valuable historical observations, since even similar observations may have different teammates' actions. Hence, their actions cannot be accurately predicted for producing belief embeddings. Hence, the policy may pay more attention to the current observation than the belief to learn general actions without relation to others. As training goes on, since the teammates' actions are relatively deterministic and predictable, the belief embeddings will learn valuable information and play a more important role in the policy.

6 Conclusion

In this paper, we introduced dynamic belief learning for decentralized multi-agent cooperation, which relieves the nonstationarity by dynamically modeling the changing policies of other agents. Extensive experimental results demonstrate that our method outperforms independent learning approaches and even achieves comparable performance with CTDE methods.

Acknowledgements

This work is partially supported by grants from the Key-Area Research and Development Program of Guangdong Province under contact No. 2020B0101380001 and grants from the National Natural Science Foundation of China under contract No. 62027804, No. 61825101 and No. 62088102. The computing resources of Pengcheng Cloudbrain are used in this research.

References

[Abdallah and Kaisers, 2016] Sherief Abdallah and Michael Kaisers. Addressing environment non-stationarity by repeating q-learning updates. *JMLR*, 17(1):1582–1612, 2016.

[Banino *et al.*, 2021] Andrea Banino, Adrià Puidomenech Badia, Jacob Walker, Tim Scholtes, Jovana Mitrovic, and Charles Blundell. Coberl: Contrastive bert for reinforcement learning. *arXiv preprint arXiv:2107.05431*, 2021.

[Bowling and Veloso, 2002] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[Cao *et al.*, 2012] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, 9(1):427–438, 2012.

[Carroll *et al.*, 2019] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *NeurIPS*, 32:5174–5185, 2019.

[Chao *et al.*, 2021] YU Chao, A VELU, E VINITSKY, et al. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv:2103.01955*, 2021.

[Chen *et al.*, 2017] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *ICRA*, pages 285–292, 2017.

[Chen *et al.*, 2021] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *NeurIPS*, 34, 2021.

[de Witt *et al.*, 2020] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv:2011.09533*, 2020.

[Duan *et al.*, 2017] Yan Duan, Marcin Andrychowicz, Bradly C Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *arXiv:1703.07326*, 2017.

[Foerster *et al.*, 2018] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI*, pages 2974–2982, 2018.

[Fu *et al.*, 2022] Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. Revisiting some common practices in cooperative multi-agent reinforcement learning. In *ICML*, pages 6863–6877. PMLR, 2022.

[He *et al.*, 2016] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *ICML*, pages 1804–1813. PMLR, 2016.

[Hernandez-Leal *et al.*, 2019] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. Agent modeling as auxiliary task for deep reinforcement learning. In *AIIDE*, volume 15, pages 31–37, 2019.

[Hong *et al.*, 2017] 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. *arXiv:1712.07893*, 2017.

[Hu *et al.*, 2020] Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. Updet: Universal multi-agent rl via policy decoupling with transformers. In *ICLR*, 2020.

[Hu *et al.*, 2021] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih-wei Liao. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. *arXiv e-prints*, pages arXiv–2102, 2021.

[Inala *et al.*, 2020] Jeevana Priya Inala, Yichen Yang, James Paulos, Yewen Pu, Osbert Bastani, Vijay Kumar, Martin Rinard, and Armando Solar-Lezama. Neurosymbolic transformers for multi-agent communication. *NeurIPS*, 33:13597–13608, 2020.

[Janner *et al.*, 2021] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *NeurIPS*, 34, 2021.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

[Kraemer and Banerjee, 2016] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

[Lowe *et al.*, 2017] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv:1706.02275*, 2017.

[Lyu and Amato, 2018] Xueguang Lyu and Christopher Amato. Likelihood quantile networks for coordinating multi-agent reinforcement learning. *arXiv preprint arXiv:1812.06319*, 2018.

[Ma *et al.*, 2021a] Xiaoteng Ma, Yiqin Yang, Chenghao Li, Yiwen Lu, Qianchuan Zhao, and Jun Yang. Modeling the interaction between agents in cooperative multi-agent reinforcement learning. In *AAMAS*, pages 853–861, 2021.

[Ma *et al.*, 2021b] Yuxi Ma, Meng Shen, Yuhang Zhao, Zhao Li, Xiaoyao Tong, Quanxin Zhang, and Zhi Wang. Opponent portrait for multiagent reinforcement learning in competitive environment. *International Journal of Intelligent Systems*, 36(12):7461–7474, 2021.

[Meng *et al.*, 2021] Linghui Meng, Muning Wen, Yaodong Yang, Chenyang Le, Xiyun Li, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, and Bo Xu. Offline pre-trained multi-agent decision transformer: One big sequence model conquers all starcraftii tasks. *arXiv preprint arXiv:2112.02845*, 2021.

[Moreno *et al.*, 2021] Pol Moreno, Edward Hughes, Kevin R McKee, Bernardo Avila Pires, and Théophane Weber. Neural recursive belief states in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.02274*, 2021.

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

[Palmer *et al.*, 2018] Gregory Palmer, Rahul Savani, and Karl Tuyls. Negative update intervals in deep multi-agent reinforcement learning. *arXiv preprint arXiv:1809.05096*, 2018.

[Papoudakis and Albrecht, 2020] Georgios Papoudakis and Stefano V Albrecht. Variational autoencoders for opponent modeling in multi-agent systems. *arXiv:2001.10829*, 2020.

[Parisotto *et al.*, 2020] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *ICML*, pages 7487–7498. PMLR, 2020.

[Pu *et al.*, 2016] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. *NeurIPS*, 29:2352–2360, 2016.

[Rabinowitz *et al.*, 2018] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *ICML*, pages 4218–4227. PMLR, 2018.

[Rashid *et al.*, 2018] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, pages 4292–4301, 2018.

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

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

[Son *et al.*, 2019] 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 *ICML*, pages 5887–5896. PMLR, 2019.

[Sukhbaatar *et al.*, 2016] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *NeurIPS*, 29:2244–2252, 2016.

[Sun *et al.*, 2022] Mingfei Sun, Sam Devlin, Katja Hofmann, and Shimon Whiteson. Monotonic improvement guarantees under non-stationarity for decentralized ppo. *arXiv preprint arXiv:2202.00082*, 2022.

[Sunehag *et al.*, 2017] 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:1706.05296*, 2017.

[Tan, 1993] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *ICML*, pages 330–337, 1993.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[Wang *et al.*, 2020a] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv:2008.01062*, 2020.

[Wang *et al.*, 2020b] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. *arXiv:2010.01523*, 2020.

[Wen *et al.*, 2019] Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. *arXiv preprint arXiv:1901.09207*, 2019.

[Wen *et al.*, 2021] Ying Wen, Yaodong Yang, and Jun Wang. Modelling bounded rationality in multi-agent interactions by generalized recursive reasoning. In *IJCAI*, pages 414–421, 2021.

[Yang *et al.*, 2019] Tianpei Yang, Jianye Hao, Zhaopeng Meng, Chongjie Zhang, Yan Zheng, and Ze Zheng. Towards efficient detection and optimal response against sophisticated opponents. In *IJCAI*, 2019.

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