A Variational Bound of Teammate modeling

In order to make the teammate models anticipate intentions of teammates, we propose to maximize the mutual information between a_j and $\boldsymbol{z}_{ij}$. We draw the idea from variational inference and derive a lower bound of this mutual information term.

Theorem 1. *Let $I\left(\boldsymbol{z}_{ij}; a_j \mid \tau_i, d_j\right)$ be the mutual information between the action a_j of agent j and the teammate model $\boldsymbol{z}_{ij}$ from agent i to agent j conditioned on agent i's local information. The lower bound is given by*

$$\mathbb{E}_{\mathcal{D}}\left[-D_{KL}\left(p\left(\boldsymbol{z}_{ij} \mid \tau_i, d_j\right) \| \, q_\xi\left(\boldsymbol{z}_{ij} \mid \tau_i, a_j, d_j\right)\right)\right]. \quad (1)$$

Here, d_j is the specific ID of teammate j in samples $(\boldsymbol{z}_{ij}, a_j, \tau_i, d_j)$, which are sampled from the replay buffer $\mathcal{D}$, and $q_\xi(\boldsymbol{z}_{ij} \mid \tau_i, a_j, d_j)$ is the variational posterior estimator with parameters ξ.

Proof. By a variational distribution $q_\xi\left(\boldsymbol{z}_{ij} \mid \tau_i, a_j, d_j\right)$ parameterized by ξ, we have

$$\begin{aligned}
&I\left(\boldsymbol{z}_{ij}; a_j \mid \tau_i, d_j\right) \\
=&\mathbb{E}_{\boldsymbol{z}_{ij}, a_j, \tau_i, d_j}\left[\log \frac{p(\boldsymbol{z}_{ij}, a_j \mid \tau_i, d_j)}{p(\boldsymbol{z}_{ij} \mid \tau_i, d_j) p(a_j \mid \tau_i, d_j)}\right] \\
=&\mathbb{E}_{\boldsymbol{z}_{ij}, a_j, \tau_i, d_j}\left[\log \frac{p\left(\boldsymbol{z}_{ij} \mid a_j, \tau_i, d_j\right)}{p\left(\boldsymbol{z}_{ij} \mid \tau_i, d_j\right)}\right] \\
=&\mathbb{E}_{a_j, \boldsymbol{z}_{ij}, \tau_i, d_j}\left[\log \frac{q_\xi\left(\boldsymbol{z}_{ij} \mid a_j, \tau_i, d_j\right)}{p\left(\boldsymbol{z}_{ij} \mid \tau_i, d_j\right)}\right] + \\
&D_{KL}\left(p\left(\boldsymbol{z}_{ij} \mid \tau_i, d_j\right) \| \, q_\xi\left(\boldsymbol{z}_{ij} \mid a_j, \tau_i, d_j\right)\right) \\
\geq&\mathbb{E}_{\boldsymbol{z}_{ij}, a_j, \tau_i, d_j}\left[\log \frac{q_\xi\left(\boldsymbol{z}_{ij} \mid a_j, \tau_i, d_j\right)}{p\left(\boldsymbol{z}_{ij} \mid \tau_i, d_j\right)}\right],
\end{aligned} \quad (2)$$

where the last inequality holds because of the non-negativity of the KL divergence. The lower bound is tight when $q_\xi(\boldsymbol{z}_{ij} \mid a_j, \tau_i, d_j)$ approximates $p(\boldsymbol{z}_{ij} \mid \tau_i, d_j)$ well. We can rewrite it as

$$\begin{aligned}
&I\left(\boldsymbol{z}_{ij}; a_j \mid \tau_i, d_j\right) \\
\geq&\mathbb{E}_{\boldsymbol{z}_{ij}, a_j, \tau_i, d_j}\left[\log q_\xi\left(\boldsymbol{z}_{ij} \mid a_j, \tau_i, d_j\right)\right] - \\
&\mathbb{E}_{\boldsymbol{z}_{ij}, a_j, \tau_i, d_j}\left[\log p\left(\boldsymbol{z}_{ij} \mid \tau_i, d_j\right)\right] \\
=&\mathbb{E}_{\boldsymbol{z}_{ij}, a_j, \tau_i, d_j}\left[\log q_\xi\left(\boldsymbol{z}_{ij} \mid a_j, \tau_i, d_j\right)\right] + H\left(\boldsymbol{z}_{ij} \mid \tau_i, d_j\right) \\
=&\mathbb{E}_{a_j, \tau_i, d_j}\left[\int p(\boldsymbol{z}_{ij} | a_j, \tau_i, d_j) \log q_\xi(\boldsymbol{z}_{ij} | a_j, \tau_i, d_j) \mathrm{d}\boldsymbol{z}_{ij}\right] + \\
&H\left(\boldsymbol{z}_{ij} \mid \tau_i, d_j\right).
\end{aligned} \quad (3)$$

As the teammate modeling encoder takes τ_i and d_j as input, the teammate model $\boldsymbol{z}_{ij}$ is independent from a_j given the local information. So we have

$$\begin{aligned}
&I\left(\boldsymbol{z}_{ij}; a_j \mid \tau_i, d_j\right) \\
\geq&\mathbb{E}_{a_j, \tau_i, d_j}\Big[\int p(\boldsymbol{z}_{ij} \mid a_j, \tau_i, d_j) \log q_\xi(\boldsymbol{z}_{ij} \mid a_j, \tau_i, d_j) - \\
&p(\boldsymbol{z}_{ij} \mid \tau_i, d_j) \log p(\boldsymbol{z}_{ij} \mid \tau_i, d_j) \mathrm{d}\boldsymbol{z}_{ij}\Big] \\
=&\mathbb{E}_{a_j, \tau_i, d_j}\left[-D_{KL}\left(p\left(\boldsymbol{z}_{ij} \mid \tau_i, d_j\right) \| \, q_\xi\left(\boldsymbol{z}_{ij} \mid \tau_i, a_j, d_j\right)\right)\right],
\end{aligned} \quad (4)$$

where a_j, τ_i, and d_j are computed from the replay buffer $\mathcal{D}$. Therefore, we use the replay buffer $\mathcal{D}$ to minimize the expectation of KL divergence as maximizing the lower bound of the mutual information. □

B Details and Additional Results of SMAC

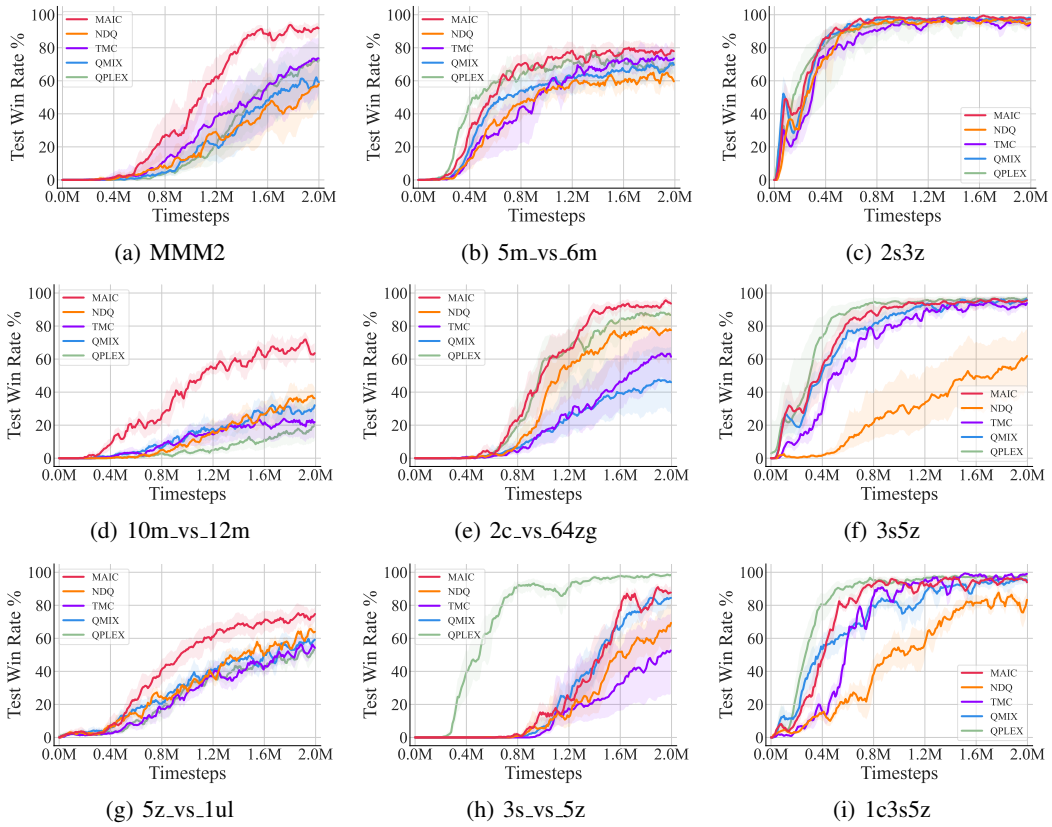
Our implementation is based on the PyMARL framework (Rashid et al. 2018) with a StarCraft version of 2.4.6.2.69232, which is always the default QMIX running version from official SMAC release[1]. We adopt the default environment setting from PyMARL, including the same observable content for each agent, the same enemy AI level, the same reward function, etc. Every algorithm follows the same training scheme during 2M timesteps. An episodic runner is used to collect trajectories, and all algorithms are trained every timestep after interacting with the environment. We apply the default ϵ-greedy exploration to every algorithm, with ϵ decaying from 1 to 0.05 in 50k timesteps, along with typical training tricks of deep Q-learning like a target network and double Q-learning. The target network is updated every 200 episodes, the same as settings in the original QMIX implementation. This setting may not provide optimal performance in some super hard scenarios, as the insufficient exploration and rare inadequate replay data will limit policy learning. However, the same setting to QMIX guarantees the fairness of comparisons.

We compare MAIC with baseline, including NDQ, TMC, VBC, QMIX, and QPLEX in 13 SMAC scenarios (Samvelyan et al. 2019) and other ablation and evaluation experiments. The descriptions of all 13 scenarios can be found in Table 1. For NDQ, QMIX, and QPLEX, we use the code provided by the authors from their original paper with default hyperparameters settings. For VBC and TMC, we implement them according to their official code as their implementations do not support an alterable number of agents well. As results of 4 of 13 scenarios have been shown above, we here exhibit curves from the other nine scenarios in Figure 1. We can observe that MAIC still outperforms other methods in most scenarios while has a less promising performance in some scenarios with fewer difficulties like 2s3z and 3s_vs_5z. QPLEX shows comparable performance in many easy and less difficult scenarios but cannot learn an effective policy in super hard scenarios.

C The Architecture, Infrastructure, and Hyperparameters Choices of MAIC

As the common training and environment settings are listed above and adopted for all algorithms, we here present specific settings, including network architectures and hyperparameters choices. The local agent network of MAIC shares the same architecture with QMIX, which has a GRU cell with a dimension of 64 to encode historical information, and two fully connected layers to compute local Q-values. The teammate modeling architecture utilizes a multi-layer perceptron (MLP) which has a hidden layer with 64 units as the encoder

[1] https://github.com/oxwhirl/smac



(a) MMM2 (b) 5m_vs_6m (c) 2s3z

(d) 10m_vs_12m (e) 2c_vs_64zg (f) 3s5z

(g) 5z_vs_1ul (h) 3s_vs_5z (i) 1c3s5z

Figure 1: Average test win rates for MAIC, NDQ, TMC, QMIX, and QPLEX in 9 additional SMAC scenarios.

Table 1: Properties of 13 conducted SMAC scenarios.

Map Name	Ally Units	Enemy Units	Type	Difficulty
2s3z	2 Stalkers, 3 Zealots	2 Stalkers, 3 Zealots	Symmetric, Heterogeneous	Easy
3s5z	3 Stalkers, 5 Zealots	3 Stalkers, 5 Zealots	Symmetric, Heterogeneous	Easy
1c3s5z	1 Colossus, 3 Stalkers, 5 Zealots	1 Colossus, 3 Stalkers, 5 Zealots	Symmetric, Heterogeneous	Easy
5m_vs_6m	5 Marines	6 Marines	Asymmetric, Homogeneous	Hard
3s_vs_5z	3 Stalkers	5 Zealots	Asymmetric, Homogeneous	Hard
2c_vs_64zg	2 Colossi	64 Zerglings	Asymmetric, Homogeneous	Hard
10m_vs_12m	10 Marines	12 Marines	Asymmetric, Homogeneous	Super hard
27m_vs_30m	27 Marines	30 Marines	Asymmetric, Homogeneous	Super hard
3c_vs_100zg	3 Colossi	100 Zerglings	Asymmetric, Homogeneous	Super hard
5z_vs_1ul	5 Zealots	1 Ultralisk	Asymmetric, Homogeneous	Super hard
1c3s5z_vs_1c3s6z	1 Colossi, 3 Stalkers, 5 Zealots	1 Colossi, 3 Stalkers, 6 Zealots	Asymmetric, Heterogeneous	Super hard
MMM2	1 Medivac, 2 Marauders, 7 Marines	1 Medivac, 2 Marauders, 8 Marines	Asymmetric, Heterogeneous	Super hard
MMM3	1 Medivac, 2 Marauders, 7 Marines	1 Medivac, 2 Marauders, 9 Marines	Asymmetric, Heterogeneous	Super hard

to calculate the parameters of teammate models represented by multivariate Gaussian distributions with a dimension of 8. We implement the encoder to simultaneously output all other teammates' models with the input of an agent's information to enhance parallelization. The message generator takes the agent's local information and representation drawed from teammate models into an MLP with one hidden layer, and computes $\boldsymbol{q}_i$ and $\boldsymbol{k}_{ij}$ with simple fully connected layers. These two intermediate representations have a dimension of 32. The final message has the same dimension to the action space to bias other agents' policies in an incentive way.

We apply mixing networks to MAIC according to existing MARL methods. The mixing network with simple additivity from VDN (Sunehag et al. 2018) is used in two small cooperative tasks, Level-Based Foraging and Hallway. The QMIX mixing network (Rashid et al. 2018) is adopted in the complex SMAC benchmark. We select the factor $\lambda_m = 0.001$ for the mutual information loss term of teammate modeling and $\lambda_c = 0.01$ for the sparse regularization in all scenarios, which are tuned from multiple different values. An RMSProp optimizer is used with parameters including the learning rate of 5×10^5, $\alpha = 0.99$, and RMSProp $\epsilon = 1 \times 10^5$. The whole framework is trained end-to-end with collected episodic data on NVIDIA GeForce RTX 2080 Ti GPUs with a time cost no more than 12 hours in Level-Based Foraging and Hallway, and no more than 24 hours in SMAC scenarios.

D Value Decomposition Methods in MARL

Recently, value function factorization learning emerges as a promising way in collaborative multi-agent systems (e.g., VDN (Sunehag et al. 2018), QMIX (Rashid et al. 2018), and QPLEX (Wang et al. 2020)). These three methods all follow the Individual-Global-Max (IGM) (Son et al. 2019) principle, which asserts the consistency between joint and local greedy action selections by the joint value function $Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a})$ and individual value functions $[Q_i(\tau_i, a_i)]_{i=1}^n$, respectively:

$$\begin{aligned} &\forall \boldsymbol{\tau}, \quad \arg\max_{\boldsymbol{a} \in \boldsymbol{\mathcal{A}}} Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}) \\ &= \left(\arg\max_{a_1 \in \mathcal{A}} Q_1\left(\tau_1, a_1\right), \ldots, \arg\max_{a_n \in \mathcal{A}} Q_n\left(\tau_n, a_n\right) \right). \end{aligned} \tag{5}$$

VDN utilizes the additivity to factorize the global value function $Q_{tot}^{\mathrm{VDN}}(\boldsymbol{\tau}, \boldsymbol{a})$:

$$Q_{tot}^{\mathrm{VDN}}(\boldsymbol{\tau}, \boldsymbol{a}) = \sum_{i=1}^{n} Q_i\left(\tau_i, a_i\right), \tag{6}$$

while QMIX constrains the global value function $Q_{tot}^{\mathrm{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a})$ with monotonicity property:

$$\forall i \in \mathcal{N}, \frac{\partial Q_{tot}^{\mathrm{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a})}{\partial Q_i\left(\tau_i, a_i\right)} > 0. \tag{7}$$

These two structures are sufficient conditions for the IGM principle but not necessary. To achieve a complete IGM function class, QPLEX (Wang et al. 2020) uses a duplex dueling network architecture by decomposing the global value function $Q_{tot}^{\mathrm{QPLEX}}(\boldsymbol{\tau}, \boldsymbol{a})$ as:

$$\begin{aligned} &Q_{tot}^{\mathrm{QPLEX}}(\boldsymbol{\tau}, \boldsymbol{a}) = V_{tot}(\boldsymbol{\tau}) + A_{tot}(\boldsymbol{\tau}, \boldsymbol{a}) \\ &= \sum_{i=1}^{n} Q_i\left(\boldsymbol{\tau}, a_i\right) + \sum_{i=1}^{n} \left(\lambda_i(\boldsymbol{\tau}, \boldsymbol{a}) - 1\right) A_i\left(\boldsymbol{\tau}, a_i\right), \end{aligned} \tag{8}$$

where $\lambda_i(\boldsymbol{\tau}, \boldsymbol{a})$ is the weight depending on the joint history and joint action. The difference among the three methods is in the mixing networks, with increasing representational complexity. Our proposed framework MAIC follows the value factorization learning paradigm but focuses on enhancing the representative ability of agents' individual local networks. Different global mixing networks in VDN, QMIX, and QPLEX can be freely integrated with MAIC.

References

Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, 4295–4304.

Samvelyan, M.; Rashid, T.; De Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G.; Hung, C.-M.; Torr, P. H.; Foerster, J.; and Whiteson, S. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*.

Son, K.; Kim, D.; Kang, W. J.; Hostallero, D. E.; and Yi, Y. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *ICML*, 5887–5896.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V. F.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; and Graepel, T. 2018. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *AAMAS*, 2085–2087.

Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; and Zhang, C. 2020. QPLEX: Duplex dueling multi-agent Q-learning. *arXiv preprint arXiv:2008.01062*.