Supplementary material for Leveraging Partial Symmetry for Multi-Agent Reinforcement Learning

Xin Yu[1], Rongye Shi[2*], Pu Feng[1], Yongkai Tian[1], Simin Li[1], Shuhao Liao[1], Wenjun Wu[2]

[1]School of Computer Science and Engineering, Beihang University, China
[2]Institute of Artificial Intelligence, Beihang University, China
{nlsdeyuxin, shirongye, fengpu, tianyk, lisiminsimon, lsh_iai, wwj}@buaa.edu.cn

1 Proof of the Proposition 1

We first recap that the Partially Symmetric Markov game $\mathcal{M}_g$, which has the following two defined properties:

The partial reward invariance is characterized with

$$|R(s,a) - R(gs,ga)| \leq \epsilon. \tag{1}$$

The partial transition invariance is characterized using the Maximum Mean Discrepancy (MMD) between the distributions of transitions $T(s'|s,a)$ and $T(gs'|gs,ga)$:

$$\text{MMD}_{\mathcal{F}}\left(T(s'|s,a), T(gs'|gs,ga)\right) \leq \delta. \tag{2}$$

Please note that $T(gs'|gs,ga)$ is also a distribution of s' with the sampling process $s' \sim T(gs'|gs,ga)$ being defined as 1) $gs' \sim T(gs'|gs,ga)$ and then 2) $s' = L_g^{-1}(gs')$.

The MMD is a measure of distance used to quantify the discrepancy between two distributions under a general class of bounded mapping functions $f \in \mathcal{F}$. Eq.(2) can be expressed as:

$$\begin{aligned}&\text{MMD}_{\mathcal{F}}\left(T(s'|s,a), T(gs'|gs,ga)\right)\\ &= \sup_{f\in\mathcal{F}} \left|\mathbb{E}_{s'\sim T(s'|s,a)}[f(s')] - \mathbb{E}_{s'\sim T(gs'|gs,ga)}[f(s')]\right|.\end{aligned}$$

Proposition 1 (Performance Error Bound)**.** *If a Partial Symmetry Markov game $\mathcal{M}_g$ satisfies the conditions in equations* (1) *and* (2) *for all $(s,a) \in \Psi$, then the performance error $Error_{\mathcal{M}_g} = |Q^\star(s,a) - Q^\star(gs,ga)|$ is bounded by $\frac{\epsilon}{1-\gamma} + \frac{\gamma\delta}{1-\gamma}$.*

Proof. We will apply the mathematical induction method for the proof. We define the m-step optimal discounted action value function recursively for all $(s,a) \in \Psi$ and for all non-negative integers m as follows:

$$Q_m(s,a) = R(s,a) + \gamma \sum_{s'\in S}[T(s'|s,a) \max_{a'\in A} Q_{m-1}(s',a')], \tag{3}$$

where we set $Q_{-1}(s',a') = 0$. Please note that because the reward R is bounded, $Q_m(s,a)$ is bounded for any natural number m, and so do the functions of $\max_{a'} Q_m(s',a')$ and $\max_{a'} Q_m(gs',ga')$. Therefore, they are all elements of $\mathcal{F}$.

Step 1 (base step): For the base case of $m = 0$, we have

$$|Q_0(s,a) - Q_0(gs,ga)| = |R(s,a) - R(gs,ga)| \leq \epsilon$$

Step 2 (base step): For the base case of $m = 1$, we have

$$\begin{aligned}&|Q_1(s,a) - Q_1(gs,ga)|\\ &= \Big|R(s,a) + \gamma \sum_{s'} T(s'|s,a) \max_{a'} Q_0(s',a')\\ &\quad - R(gs,ga) - \gamma \sum_{s'} T(gs'|gs,ga) \max_{a'} Q_0(gs',ga')\Big|\\ &\leq \Big|R(s,a) - R(gs,ga)\Big| + \gamma\Big|\sum_{s'} T(s'|s,a) \max_{a'} Q_0(s',a')\\ &\quad - \sum_{s'} T(gs'|gs,ga) \max_{a'} Q_0(gs',ga')\Big|\\ &\leq \epsilon + \gamma\Big|\sum_{s'} T(s'|s,a) \max_{a'} Q_0(gs',ga')\\ &\quad - \sum_{s'} T(gs'|gs,ga) \max_{a'} Q_0(gs',ga')\Big|\end{aligned}$$

If $\sum_{s'} T(s'|s,a) \max_{a'} Q_0(s',a')$ is greater than or equal to $\sum_{s'} T(gs'|gs,ga) \max_{a'} Q_0(gs',ga')$, then we have:

$$\begin{aligned}&\Big|\sum_{s'} T(s'|s,a) \max_{a'} Q_0(s',a') - \sum_{s'} T(gs'|gs,ga) \max_{a'} Q_0(gs',ga')\Big|\\ &\leq \Big|\sum_{s'} T(s'|s,a) \max_{a'}(Q_0(gs',ga') + \epsilon)\\ &\quad - \sum_{s'} T(gs'|gs,ga) \max_{a'} Q_0(gs',ga')\Big|\\ &\leq \epsilon + \Big|\sum_{s'} T(s'|s,a) \max_{a'} Q_0(gs',ga')\\ &\quad - \sum_{s'} T(gs'|gs,ga) \max_{a'} Q_0(gs',ga')\Big|\\ &= \epsilon + \Big|\mathbb{E}_{T(s'|s,a)}[f(s')] - \mathbb{E}_{T(gs'|gs,ga)}[f(s')]\Big|_{f(s')=\max_{a'} Q_0(gs',ga')}\\ &\leq \epsilon + \text{MMD}_{\mathcal{F}}\left(T(s'|s,a), T(gs'|gs,ga)\right)\\ &\leq \epsilon + \delta\end{aligned}$$

If $\sum_{s'} T(s'|s,a) \max_{a'} Q_0(s',a')$ is smaller than $\sum_{s'} T(gs'|gs,ga) \max_{a'} Q_0(gs',ga')$, then we have:

*Corresponding author: Rongye Shi.

$$
\begin{aligned}
&\left|\sum_{s'} T(s'|s,a)\max_{a'} Q_0(s',a') - \sum_{s'} T(gs'|gs,ga)\max_{a'} Q_0(gs',ga')\right| \\
&= \left|\sum_{s'} T(gs'|gs,ga)\max_{a'} Q_0(gs',ga') - \sum_{s'} T(s'|s,a)\max_{a'} Q_0(s',a')\right| \\
&\leq \left|\sum_{s'} T(gs'|gs,ga)\max_{a'}(Q_0(s',a')+\epsilon)\right. \\
&\quad \left. - \sum_{s'} T(s'|s,a)\max_{a'} Q_0(s',a')\right| \\
&\leq \epsilon + \left|\sum_{s'} T(gs'|gs,ga)\max_{a'} Q_0(s',a')\right. \\
&\quad \left. - \sum_{s'} T(s'|s,a)\max_{a'} Q_0(s',a')\right| \\
&= \epsilon + \left|\mathbb{E}_{T(gs'|gs,ga)}[f(s')] - \mathbb{E}_{T(s'|s,a)}[f(s')]\right|_{f(s')=\max_{a'} Q_0(s',a')} \\
&\leq \epsilon + \text{MMD}_{\mathcal{F}}\left(T(s'|s,a), T(gs'|gs,ga)\right) \\
&\leq \epsilon + \delta
\end{aligned}
$$

Taking the above discussions into account, we conclude that

$$|Q_1(s,a) - Q_1(gs,ga)| \leq \epsilon + \gamma(\epsilon+\delta) = (1+\gamma)\epsilon + \gamma\delta$$

Step 3 (induction step): As we have the base cases, let's assume that

$$|Q_j(s,a) - Q_j(gs,ga)| \leq \sum_{i=0}^{j}\gamma^i\epsilon + \sum_{i=1}^{j}\gamma^i\delta$$

holds for all non-negative integer $j < m$ and all state-action pairs $(s,a) \in \Psi$. Now we prove by induction on m that the proposition is true. For the case m, we have:

$$
\begin{aligned}
&|Q_m(s,a) - Q_m(gs,ga)| \\
&= \left|R(s,a) + \gamma\sum_{s'} T(s'|s,a)\max_{a'} Q_{m-1}(s',a')\right. \\
&\quad \left. - R(gs,ga) - \gamma\sum_{s'} T(gs'|gs,ga)\max_{a'} Q_{m-1}(gs',ga')\right| \\
&\leq |R(s,a) - R(gs,ga)| + \gamma\left|\sum_{s'} T(s'|s,a)\max_{a'} Q_{m-1}(s',a')\right. \\
&\quad \left. - \sum_{s'} T(gs'|gs,ga)\max_{a'} Q_{m-1}(gs',ga')\right| \\
&\leq \epsilon + \gamma\left|\sum_{s'} T(s'|s,a)\max_{a'} Q_{m-1}(s',a')\right. \\
&\quad \left. - \sum_{s'} T(gs'|gs,ga)\max_{a'} Q_{m-1}(gs',ga')\right|
\end{aligned}
$$

By repeating the same discussions in Step 2 regarding two situations of $\sum_{s'} T(s'|s,a)\max_{a'} Q_{m-1}(s',a')$ and $\sum_{s'} T(gs'|gs,ga)\max_{a'} Q_{m-1}(gs',ga')$, we can obtain that:

$$
\begin{aligned}
&\left|\sum_{s'} T(s'|s,a)\max_{a'} Q_{m-1}(s',a')\right. \\
&\quad \left. - \sum_{s'} T(gs'|gs,ga)\max_{a'} Q_{m-1}(gs',ga')\right| \\
&\leq \left(\sum_{i=0}^{m-1}\gamma^i\epsilon + \sum_{i=1}^{m-1}\gamma^i\delta\right) + \delta
\end{aligned}
$$

Therefore, we have

$$
\begin{aligned}
&|Q_m(s,a) - Q_m(gs,ga)| \\
&\leq \epsilon + \gamma\left(\sum_{i=0}^{m-1}\gamma^i\epsilon + \sum_{i=1}^{m-1}\gamma^i\delta\right) + \gamma\delta \\
&= \sum_{i=0}^{m}\gamma^i\epsilon + \sum_{i=1}^{m}\gamma^i\delta
\end{aligned}
$$

Based on mathematical induction, since the statement of the proposition in both the base step and induction step has been proved as true, $|Q_m(s,a) - Q_m(gs,ga)|$ is bounded for all m. When $0 < \gamma < 1$ and m approaches infinity, we have:

$$
\begin{aligned}
&\mathit{Error}_{\mathcal{M}_g} = |Q^\star(s,a) - Q^\star(gs,ga)| \\
&= \lim_{m\to\infty}\left|Q^m(s,a) - Q^m(gs,ga)\right| \\
&\leq \lim_{m\to\infty}\left(\sum_{i=0}^{m}\gamma^i\epsilon + \sum_{i=1}^{m}\gamma^i\delta\right) \\
&= \frac{\epsilon}{1-\gamma} + \frac{\gamma\delta}{1-\gamma}
\end{aligned}
$$

The proof of proposition 1 is completed. □

2 Data Augmentation in MAPPO

Although data augmentation is familiar in the realm of RL, we contend that its practical deployment poses challenges. We'll delve into its implementation, particularly in the equation (4). Within the standard MAPPO, agents leverage the technique of parameter sharing. As such, the MAPPO learns policy π_θ by optimizing the following objective:

$$J_\pi(\theta) = \sum_{i=1}^{n} E_{\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta\left(a^i \mid s\right)}{\pi_{\theta_{\text{old}}}\left(a^i \mid s\right)} A^\pi\left(s, a^i\right)\right], \quad (4)$$

where $\pi_{\theta_{\text{old}}}$ is the behavior policy used to collect trajectories, π_θ is the policy we want to optimize, and $A^\pi\left(s,a^i\right)$ denotes the advantage function. During training, the components of equation (4) are calculated as:

1. The numerator of the first term is derived from the current policy π using samples from the buffer. When dealing with augmented samples $(L_g[s], K_g[a])$, this term is represented as $\pi_\theta(K_g[a] \mid L_g[s])$.
2. The denominator is formulated using the state-action pair (s,a) and the previous policy $\pi_{\theta_{old}}$. We can only interact with the environment through the real state, The denominator is $\pi_{\theta_{old}}(a \mid s)$.
3. The term A in equation (4) is calculated using the rewards generated during the interaction. Therefore, this term is based on the original samples, i.e., $A^\pi_\psi\left(s,a^i\right)$.

Thus, If transformation (L_g, K_g) is applied to the algorithm, the MAPPO objective changes, and equation (4) is replaced by (5).

$$J_\pi(\theta) = \sum_{i=1}^{n} E_{\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta\left(K_g[a^i] \mid L_g[s]\right)}{\pi_{\theta_{\text{old}}}\left(a^i \mid s\right)} A^\pi_\psi\left(s, a^i\right)\right]. \quad (5)$$

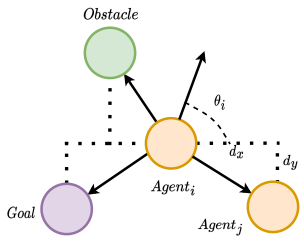


Figure 1: The observation space in formation change.

However, the right-hand side doesn't reliably estimate the left-hand side, given that $\pi_\theta(K_g[a] \mid L_g[s]) \neq \pi_\theta(a \mid s)$. Indeed, a specific transformation (L_g, K_g) in data augmentation might yield a large ratio of $\frac{\pi_\theta(K_g[a]|L_g[s])}{\pi_{\theta_{\text{old}}}(a|s)}$. In multi-agent settings, when multiple agents are considered, more sources of variance are introduced, making the training severely unstable. Our proposed symmetry consistency loss provides a solution to this challenge. It is worth noting that depending on the specific data augmentation method used, there may be minor differences in the details, but most papers that use data augmentation follow this approach(Laskin et al. 2020). When using different data augmentation implementations, even if the numerator and denominator are transformed simultaneously, the output of the formula is not the correct value.

3 Details of the Formation Change Task

In this section, we describe the 'formation change' task by defining its state space, action space, and reward function.

State space. As shown in figure 1, the observation of each robot is the concatenation of its direction and the relative coordinates of the others in the environment.

$$(\theta_i, dx_1, dy_1, dx_2, dy_2, dx_3, dy_3),$$

where θ_i is the velocity direction of agent i. And the (dx, dy) indicates the relative coordinates of the others in the environment. The subscript i represents the nearest obstacle, goal, and agents when equals 1,2,3 respectively.

Action space. We choose Epuck as the robot model in the Webots simulator. Agents can assign the wheel speed to control the robot.

Reward function. For each agent i, we define a reward function as Equation (6). The desired target position for agent i is represented as $T_i \in \mathbb{R}^2$, and the avoidance radius is denoted by $\Theta \geq 0$. At step t, the coordinates of agent i and obstacle j are specified as P_i^t and O_j, respectively.

$$R_i^t = D_i^t + C_i^t. \tag{6}$$

D_i^t stands for the distance reward:

$$D_i^t = 100 \left(\left\| P_i^{t-1} - T_i \right\| - \left\| P_i^t - T_i \right\| \right). \tag{7}$$

C_i^t stands for the collision penalty:

$$C_i^t = \begin{cases} 0.35 & \text{if } \|P_i^t - O_j\| < 0.5 \\ 0 & \text{otherwise} \end{cases}. \tag{8}$$

4 Symmetry Breaking

4.1 Predator-Prey and Cooperative Navigation

We focus on a discrete-action version of the MPE environment. Here, the range of actions is confined to movements up, down, left, right, or remaining stationary. If the action chosen is to move right, it equates to a displacement of 5 units to the right. To simulate the behavior of robots on uneven terrains, we incorporate Gaussian noise. After a robot executes an action within this environment, Gaussian noise is superimposed onto its state transition. In the main text, we made reference to 'noise intensity'. To further clarify, 'noise intensity' denotes the mean and variance of Gaussian noise. For instance, a noise intensity of one signifies Gaussian noise with both a mean and a variance of one. This introduced randomness simulates the uneven ground, effectively breaking spatial symmetry. When the Gaussian noise has a mean of zero and a non-zero variance, it emulates a rugged but flat surface. Such terrains can be visualized as ones disrupted by gravel, small rocks, or other minor obstructions, leading to their uneven nature. Gaussian noise characterized by both a non-zero mean and variance simulates slanted and rugged terrains, akin to a gravelly slope.

4.2 Wildlife Monitoring

In the context of the wildlife monitoring task, agents move in a grid world, taking discrete actions such as moving up, down, left, right, or remaining stationary. For instance, choosing the action to move right increments the agent's horizontal coordinate by one unit. To introduce partial symmetry, we infuse noise into the state transitions. Specifically, when the "noise level" cited in the main content is i, there's a probability of $i \times 10\%$ that, after executing an action, the agent will transition to a different state than initially intended, thereby disrupting the inherent symmetry.

4.3 Formation Change

The Formation Change task stands as a notable challenge in robotics. In our approach, we employed the Webots simulator and harnessed its distinct `uneven terrain` feature. Rather than just a visual aspect, this terrain is generated mathematically through Perlin noise, a reputable method for crafting uneven terrains. The noise intensity mentioned in the main content signifies the height parameter of the `uneven terrain` within Webots. As the noise intensity increases, the terrain becomes more rugged, breaking the spatial symmetry and thereby posing additional challenges for the robots.

5 Physical Deployment

In real-world scenarios, we executed experiments with e-puck2 robots, a mini mobile robot conceived by EPFL (Mondada et al. 2009). To amplify their computational capabilities, we incorporated an expansion board endowed with a Raspberry Pi (Millard et al. 2017). Our assessments transpired in a 2m x 2m indoor domain, encircled by walls. We employed a motion capture system for our experiments (Pfister et al. 2014). For the sake of clarity, we omitted the obstacles previously discussed.

Figure 2: The Epuck robots for formation change task.

By integrating our PSE framework with the MAPPO algorithm, the agents are able to complete tasks with fewer risky states. A state is labeled as "high-risk" when the distance between agents is less than 5 centimeters. We measured the frequency of such high-risk states. Our results indicated that the PSE-MAPPO approach reduced these occurrences to 2.1% of the total states, while with the standalone MAPPO, it was 5.6%. This underscores the efficacy of our method in enhancing algorithmic performance in real-world scenarios. For a more detailed and visual insight into our findings, we recommend consulting our videos.

6 Hyperparameters

In all our scenarios, the hyperparameters used the default parameters from paper (Yu et al. 2021). Table 1 describes the hyperparameters for MAPPO. For PSE, each interaction sample along with its augmented ones will be added all together to the buffer at each time step. We used the same buffer size for both PSE and non-PSE across all experiments. Table 2 describes the hyperparameters for the QMix and MADDPG. We utilize parameter sharing when there are homogenous agents because this has been the accepted practice in the majority of past MARL works (Christianos et al. 2021; Rashid et al. 2018).

To facilitate the training process for the QMIX algorithm in the "formation change" task, we made a modification to the continuous action space. We transformed the continuous action space by breaking it down into distinct atomic actions. For instance, for an action space that spans continuously from -1 to 1, we implemented a discretization strategy at regular 0.2 intervals. The computational simulations and experiments for our study were executed on Ubuntu 18.04 equipped with an i9-12900KF CPU. As for the software framework, the experiments leveraged Python version 3.9.12 coupled with PyTorch version 1.13.0 to ensure efficient computation and reproducibility.

7 Symmetry Loss for Value-based Algorithm

For clarity of presentation, the MADDPG is used as an example to introduce the symmetry consistency loss for a value-based algorithm. The MADDPG adapts the centralized training with decentralized execution (CTDE) framework. The centralized action-value function $Q_i^{\boldsymbol{\mu}}$ is updated by:

$$S_{MADDPG}(\phi) = E_{s,a}\left[\left(Q\left(s, a_1, \ldots, a_N\right) - y\right)^2\right] \quad (9)$$

where $y = r + \gamma Q(s', a_1', \ldots, a_N')$ and a_j' is obtained by the current policy; $Q_\phi(s, a)$ represent the Q-value under the original state input, and $Q_\phi(L_g[s], K_g^s[a])$ to represent the Q-value under symmetry state input where a indicates the joint action of all agents. Then, the square of L_2 distance between the Q-network outputs under the original state, and the symmetric transformed state can be directly calculated by:

$$S_{MADDPG}\left(\phi\right) = E_{s,a}\left[\left(Q_\phi(s,a) - Q_\phi(L_g[s], K_g^s[a])\right)^2\right]. \quad (10)$$

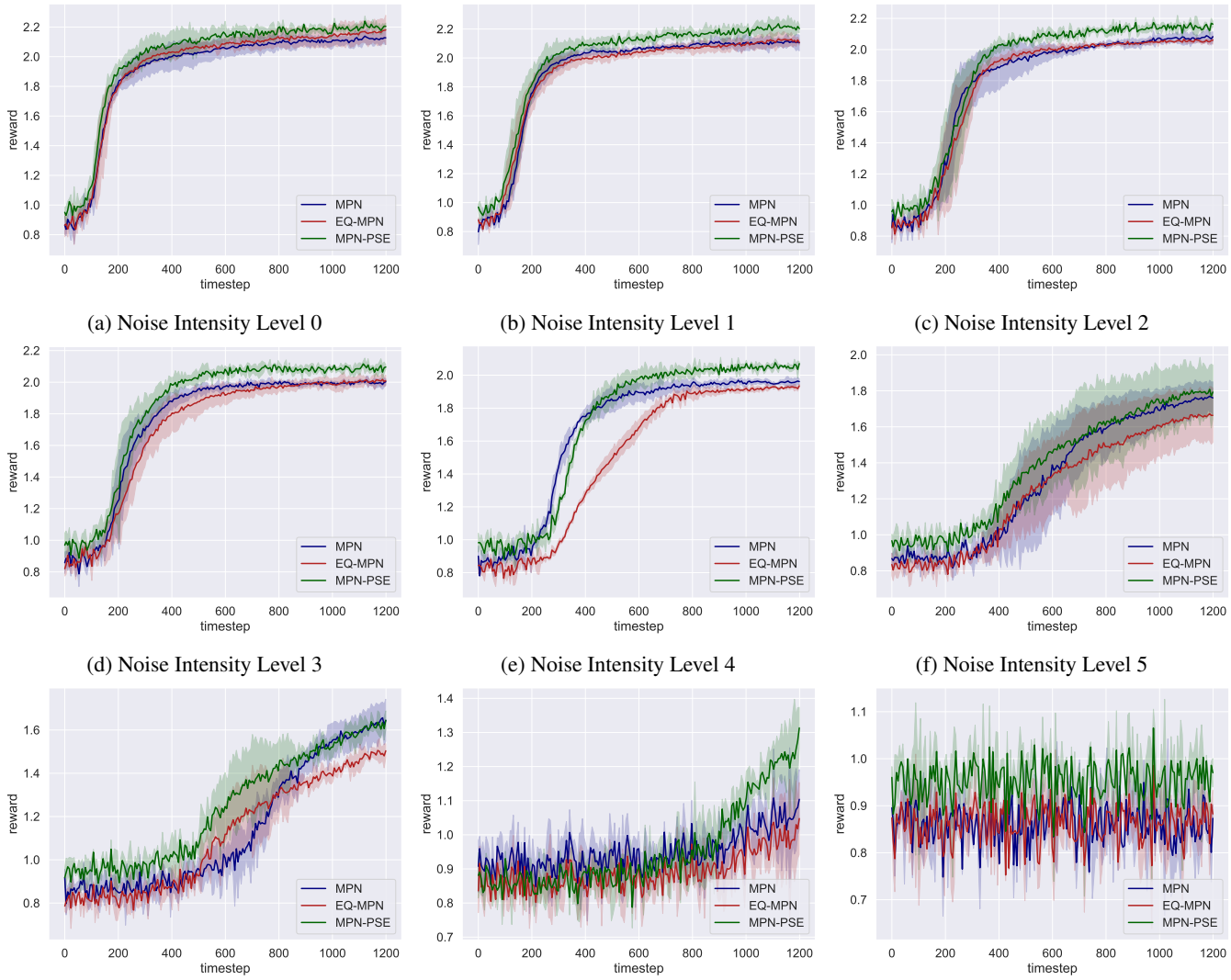
This item can help constrain the agent's value function to satisfy the defined symmetry constraints.

Table 1: hyperparameters used in MAPPO.

Hyperparameters	value
recurrent data chunk length	10
gradient clip norm	10.0
gae lamda	0.95
gamma	0.99
value loss	huber loss
huber delta	10.0
batch size	buffer length × num agents
mini batch size	batch size / mini-batch
optimizer	Adam
optimizer epsilon	1e-5
weight decay	0
network initialization	Orthogonal
use reward normalization	True
use feature normalization	True

Table 2: Hyperparameters used in QMix and MADDPG.

Hyperparameters	Value
Gradient clip norm	10.0
Random episodes	5
Epsilon	$1 \rightarrow 0.05$
Epsilon anneal time	50000 timesteps
Train interval	1 episode
Gamma	0.99
Buffer size	5000 episodes
Batch size	32 episodes
Optimizer eps	1e-5
Optimizer	Adam
Weight decay	0
Network initialization	Orthogonal
Use reward normalization	True
Use feature normalization	True



(a) Noise Intensity Level 0 (b) Noise Intensity Level 1 (c) Noise Intensity Level 2

(d) Noise Intensity Level 3 (e) Noise Intensity Level 4 (f) Noise Intensity Level 5

(g) Noise Intensity Level 6 (h) Noise Intensity Level 8 (i) Noise Intensity Level 9

Figure 3: Performance of EQ-MPN, MPN, and MPN-PSE in wildlife monitoring across different noise levels

8 Additional Experimental Results

As illustrated in Figure 3, we assessed the performance of three distinct network architectures: EQ-MPN, MPN, and MPN-PSE, across various noise levels, representing different degrees of symmetry breaking(van der Pol et al. 2021). As the noise intensity escalated, the EQ-MPN, inherently designed for perfect symmetry, exhibited a pronounced decline in its performance. In contrast, MPN showcased commendable resilience. Most notably, the MPN-PSE, anchored in the PSE framework, consistently outperformed its counterparts at every noise level, underscoring the efficacy of our approach in contexts characterized by partial symmetry.

9 Scope and Limitations of the Method

The "Definition 2" in Sec 4.2 defines the Partially Symmetric Markov game $\mathcal{M}_g$ without specifying the scope of observation space, because we would like to provide the concept without the loss of generality: As long as the observation condition $s \neq L_g(s)$ or $a \neq K_g(a)$ holds in the game, the data points can be augmented using symmetry and our PSE method is applicable to improve the MARL. Symmetry is correlated to the observation space, and the scope should be discussed. To this end, we can divide the features of each agent's observation into non-invariant feature set $G = [g_1, g_2, \ldots, g_m]$ (e.g., global view's features) and invariant feature set $I = [i_1, i_2, \ldots, i_n]$ (e.g., individual view's features). Each g_i is sensitive to the transformations (e.g., rotations, or other geometric ones) and $g_i \neq L(g_i)$. In contrast, each i_j remains unchanged after the transformations, and $i_j = L(i_j)$. The consideration of symmetry is meaningless if we only have I in the observation space. Therefore, we should define the observation space to include non-invariant features to make the symmetry meaningful.

References

Christianos, F.; Papoudakis, G.; Rahman, M. A.; and Albrecht, S. V. 2021. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International Conference on Machine Learning*, 1989–1998. PMLR.

Laskin, M.; Lee, K.; Stooke, A.; Pinto, L.; Abbeel, P.; and Srinivas, A. 2020. Reinforcement Learning with Augmented Data. *Advances in Neural Information Processing Systems*, 33.

Millard, A. G.; Joyce, R.; Hilder, J. A.; Fleşeriu, C.; Newbrook, L.; Li, W.; McDaid, L. J.; and Halliday, D. M. 2017. The Pi-puck extension board: a Raspberry Pi interface for the e-puck robot platform. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 741–748. IEEE.

Mondada, F.; Bonani, M.; Raemy, X.; Pugh, J.; Cianci, C.; Klaptocz, A.; Magnenat, S.; Zufferey, J.-C.; Floreano, D.; and Martinoli, A. 2009. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, 59–65. IPCB: Instituto Politécnico de Castelo Branco.

Pfister, A.; West, A. M.; Bronner, S.; and Noah, J. A. 2014. Comparative abilities of Microsoft Kinect and Vicon 3D motion capture for gait analysis. *Journal of medical engineering & technology*, 38(5): 274–280.

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 *International Conference on Machine Learning*, 4295–4304. PMLR. ISBN 2640-3498.

van der Pol, E.; van Hoof, H.; Oliehoek, F. A.; and Welling, M. 2021. Multi-Agent MDP Homomorphic Networks. *arXiv preprint arXiv:2110.04495*.

Yu, C.; Velu, A.; Vinitsky, E.; Wang, Y.; Bayen, A.; and Wu, Y. 2021. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. *arXiv preprint arXiv:2103.01955*.