# HGAP: Boosting Permutation Invariant and Permutation Equivariant in Multi-Agent Reinforcement Learning via Graph Attention Network

**Bor Jiun Lin** [1] **Chun-Yi Lee** [1]

## Abstract

Graph representation has gained widespread application across various machine learning domains, attributed to its ability to discern correlations among input nodes. In the realm of Multi-agent Reinforcement Learning (MARL), agents are tasked with observing other entities within their environment to determine their behavior. Conventional MARL methodologies often suffer from training difficulties if Permutation Invariant (PI) and Permutation Equivariant (PE) properties are not considered during training. The adoption of graph representation offers a solution to these challenges by conceptualizing observed entities as a graph. In this context, we introduce the Hyper Graphical Attention Policy (HGAP) Network, which employs a graph attention mechanism to fulfill the PI and PE properties, while also understanding inter-entity interactions for decision-making. HGAP is assessed across various MARL benchmarks to confirm its effectiveness and efficiency. In addition, a series of ablation studies are provided to demonstrate its adaptability, transferability, and the capability to alleviate the complexities introduced by the POMDP constraint.

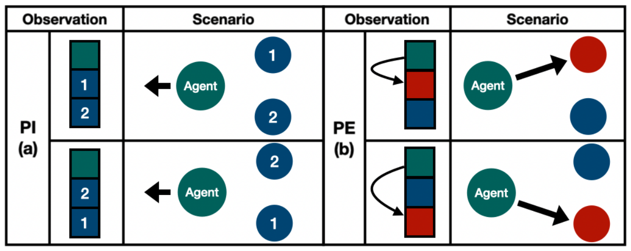


*Figure 1.* PI mandates that the agent treat the same types of entities uniformly, irrespective of their order, and make consistent action decisions. In contrast, PE demands that the agent understand the mapping relationship of the other entities to itself, rather than basing its actions on the entities' positions within its observation. (a) The agent moves toward the left even if the other entities' embedding orders change; (b) The agent acts on the red entity regardless of the positions of the entities in the agent's observation.

*Table 1.* A comparison of different methods associated with the PI, PE, and graph properties. An introduction to each method can be found in the Appendix. The symbol 'o' indicates that the method possesses the specified property, 'x' denotes the absence of it, and '△' suggests that the method only implicitly exhibits that property.

| **Methods** | **DA-MADDPG** | **Set** | **GNN** | **UPDeT** | **ASN** | **HPN** | **HGAP (Ours)** |
|---|---|---|---|---|---|---|---|
| **PI** | o | o | o | △ | o | o | o |
| **PE** | x | x | x | △ | △ | o | o |
| **Graph** | x | x | o | x | x | x | o |

## 1. Introduction

Centralized Training with Decentralized Execution (CTDE) (Oliehoek et al., 2008; Kraemer & Banerjee, 2016) paradigm in Multi-agent Reinforcement Learning (MARL) enables agents to acquire collective decision-making skills through centralized training, while maintaining individual action selection during execution. Several methodologies (Lowe et al., 2017; Foerster et al., 2018a;b; Iqbal & Sha, 2019; Jiang & Lu, 2018; Kim et al., 2019; Yang et al., 2018) have been proposed and demonstrated promising performance in various MARL benchmarks, particularly those employing value factorization approaches (Sunehag et al., 2018; Rashid et al., 2018; Son et al., 2019; 2020; Rashid et al., 2020; Wang et al., 2021a; Yang et al., 2020a;b). The success of value factorization methods has prompted modifications to agent networks (Sun et al., 2021; Hu et al., 2021), which have significantly enhanced training performance. However, these modifications tend to neglect the significance of Permutation Invariant (PI) and Permutation Equivariant (PE) properties (Hao et al., 2023), which may result in reduced training efficiency. PI necessitates consistent behavior despite variations in the embedding orders of entities, while PE involves understanding the mapping relationship between specific entities and actions. Fig. 1 illustrates that PI requires various arrangements of the same types of entities

[1]ELSA Lab, Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan. Correspondence to: Chun-Yi Lee <cylee@cs.nthu.edu.tw>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

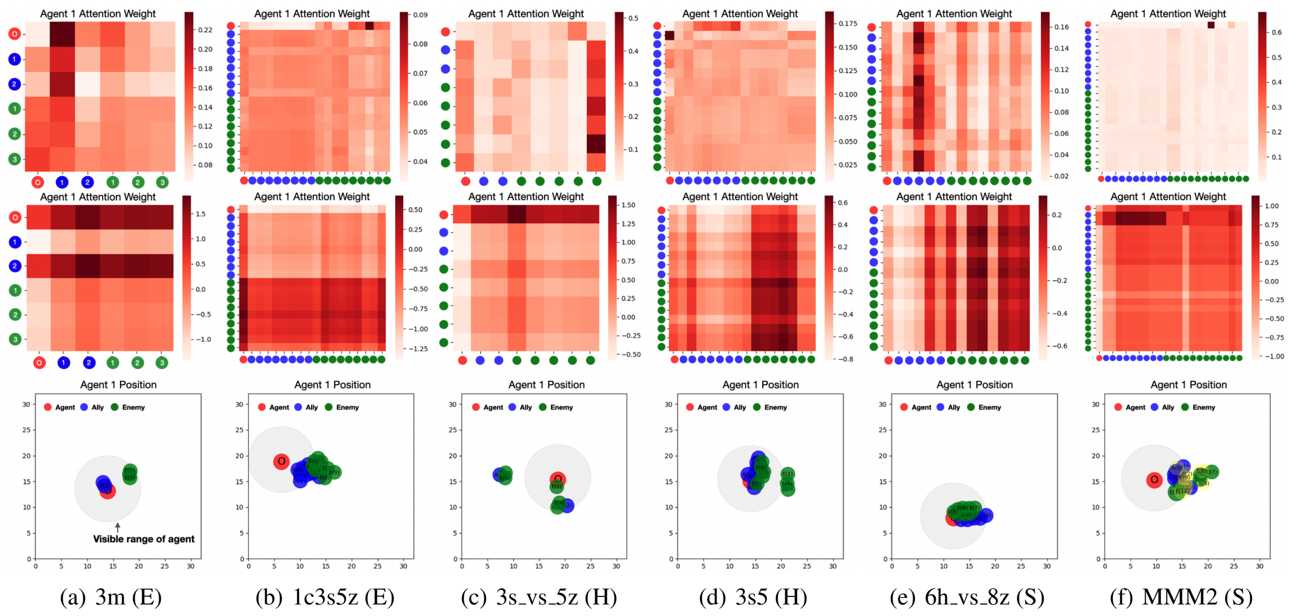


*Figure 2.* This figure illustrates the attention maps for scenarios depicted in the last row from SMAC (Vinyals et al., 2017). The first row indicates UPDeT, and the second row represents HGAP. The red, blue, and green dots represent the agent, allies, and enemies, respectively. The grey-shaded circle represents the visible range of the agent. (E), (H), and (S) denote the *Easy*, *Hard*, and *Super-Hard* scenarios, respectively. In each attention map, each horizontal row represents the attention of one agent/entity towards others, while each vertical column shows the attention received by a certain agent/entity from the others. Notice that UPDeT cannot correctly identify prioritized entities for decision-making, whereas HGAP not only focuses on crucial entities but also prioritizes the nearest ones based on their distance. In (a), (c), (d), and (e), the agents primarily undertake attacking actions, hence focusing more on enemies. In (b), the agent does not focus on other entities because it is retreating. This is due to the agent possessing a larger attacking range, leading it to prefer moving backward to avoid attacks and to strike from a distance. In (f), the agent is a Medivac (a healing unit), thus it focuses more on its allies.

possessing identical information, enabling an agent to make the same action decision regardless of the entities' order, while PE emphasizes that agents should learn the mapping relationship of entities to the agent rather than the entities' positions in its observation. Failure to consider these properties in the design of agent networks could lead to extended learning time and might adversely impact training efficiency and transferability (Hao et al., 2023). In addition, MARL setups typically involve a Partially Observable Markov Decision Process (POMDP), which can further exacerbate the above PI and PE issues, as the number of observable entities is limited. Overlooking these would demand significantly more training effort in developing effective CTDE policies.

To address the challenges, various CTDE-based MARL methods have been explored. Table 1 shows that some methods primarily focus on PI, while others consider both PI and PE properties. Earlier works such as DA-MADDPG (Ye et al., 2023), Set (Lee et al., 2019), and GNN (Liu et al., 2019) have begun to account for PI through data rearrangement-based augmentation. Nevertheless, they require considerable training time and computational resources and do not take PE into consideration (Hao et al., 2023). Recent approaches such as UPDeT (Hu et al., 2021), ASN (Wang et al., 2020b), and HPN (Hao et al., 2023) have integrated both PI and PE properties into the design of their agent networks through the use of either hypernetworks (Wang et al., 2020b; Hao et al., 2023) or attention layers (Hu et al., 2021). HPN and ASN achieve these properties by encoding the features of different entities separately, thus avoiding issues related to the ordering of them. However, both generate the embeddings of entities without considering the intercorrelations among different entities. UPDeT employs attention layers to learn correlations, and while attention layers inherently possess PI and PE properties implicitly, they may yield inconsistent attention results (Hao et al., 2023). Another concern is the increase in total training parameters (Wang et al., 2020b; Hao et al., 2023) (due to additional hypernetworks) or computational costs (Hu et al., 2021) (due to longer attention sequences) as the number of entities grows, which can lead to increased memory requirements for training agent networks (Hao et al., 2023). Furthermore, these methods do not account for entities that are out of sight due to POMDP, which can further limit the agent network's representational capability and the agent's comprehension of other entities. Addressing these challenges is therefore the primary objective of this research.

To meet the above demands, Graph Attention Network (GAT) (Veličković et al., 2018; Brody et al., 2022) has emerged as a promising candidate to be integrated into the agent network design to overcome the challenges. GATs have been applied to multiple domains (Bao et al., 2019) and have demonstrated the capability to leverage dynamic attention for learning correlations among different graph nodes, which allows them to produce representative embeddings that enhance performance in various downstream tasks (Park et al., 2022). This suggests its potential applicability to MARL problems, where capturing correlations among observed and/or even unobserved entities is crucial in POMDP. Utilizing GAT's graph representation capability also enables an agent network to disregard the ordering relationship of entities, which helps it achieve PI and PE properties. Another key advantage of GAT is its feasibility for addressing POMDP, due to its ability to infer unseen entities through graph-connected adjacent entities. This feature allows the agent to extract environmental information from graph-derived clues for more informed decision-making.

Fig. 2 presents an analysis between UPDeT and our proposed GAT-based method. The attention heatmaps generated by UPDeT lack the capability to learn entity correlations, especially in more challenging scenarios. In addition, UPDeT fails to provide sufficient information for entities out of sight. In contrast, our GAT-based method not only focuses attention on crucial entities, such as opponents, but also allocates attention based on their proximity. Moreover, our method exhibits the ability to implicitly account for entities out of sight by leveraging the strengths of GAT. These advantages position GAT as an excellent solution, filling a niche that prior approaches have not adequately addressed.

In light of the preceding discussions, we introduce a graph attention-based agent network, termed '**H**yper **G**raphical **A**ttention **P**olicy Network (HGAP)', as a solution to the challenges discussed earlier. HGAP serves as a versatile framework applicable to a wide range of MARL paradigms, including actor-critic, value factorization, and policy-based methodologies. HGAP employs GATv2 (Brody et al., 2022) to extract correlations among entities, which also ensures its adherence to the PI and PE properties. In addition, by leveraging GATv2's property as a bi-directional complete graph, HGAP possesses the ability to infer environmental clues, even when entities are out of sight. This allows HGAP to mitigate the challenges posed by POMDP. Moreover, HGAP maintains a manageable number of parameters, even as the number of entities increases. To substantiate the effectiveness of HGAP, we conduct a series of experiments across diverse MARL benchmarks (Vinyals et al., 2017; Ellis et al., 2023; Mordatch & Abbeel, 2017). We also provide ablation studies to validate HGAP's applicability to different MARL methodologies, its transfer learning capabilities, as well as its efficacy in retrieving information about unseen entities. The key contribution of this work is the evidence suggesting that HGAP satisfies PI and PE, overcomes the constraints of POMDP, and is applicable to various MARL paradigms.

## 2. Preliminaries

In this section, we begin by defining the MARL problem with Dec-POMDP, and introduce the notation to be used. Then, we present the formulation of GATv2. The backgrounds for the PI and PE properties, as well as several relevant MARL approaches, are detailed in the Appendix.

### 2.1. Definition of MARL (Dec-POMDP)

We model our MARL problem within the framework of Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek & Amato, 2016). A Dec-POMDP is formally defined as a tuple $<\mathbf{N}, \mathbb{O}, \mathbb{S}, \mathbb{A}, \mathbb{U}, \mathbf{T}, \mathbf{R}, \gamma>$, where $\mathbf{N}$ represents the set of agents, $o^i \in \mathbb{O}$ denotes the observation of agent $i$, $\mathbb{S}$ denotes the global state space, and $\mathbb{A}$ represents the action space. The global state $s \in \mathbb{S}$ includes the joint agent observation (i.e., $o^1 \times o^2 \times o^3 \times ... \times o^N$) and additional environmental information. $\mathbf{T}$ represents the state transition probability function that maps from the current global state and action space to the probability of reaching subsequent states: $\mathbb{S} \times \mathbb{A} \times \mathbb{S} \to [0, 1]$. Each agent $i$ can apply an action $u_t^i \in \mathbf{A_i}$ at each timestep $t$ from its own action space $\mathbf{A_i} \subseteq \mathbb{A}$ to the environment and subsequently receive a reward $r_t^i$ from reward function $\mathbf{R} : \mathbb{S} \times \mathbb{A}$. The joint action space is defined as $\mathbb{U} = \Pi_{i \in N} \mathbf{A_i}$, and the joint action is denoted as $u = \{u_i | i \in \mathbf{N}\} \in \mathbb{U}$. The observation of agent $i$ at $t$ is defined as $o_t^i = \{o_t^{i,own}, o_t^{i,1}, o_t^{i,2}, ..., o_t^{i,K}\}$, where $o_t^{i,own}$ is the observed environmental information and private information for agent $i$, and $o_t^{i,j}$ refers to the information observed from entity $j$ among $K$ entities. A policy $\pi_i : \mathbb{O} \times \mathbf{A_i} \to [0, 1]$ specifies the probability distribution over the actions for agent $i$. The objective for each agent $i$ is to learn a $\pi_i$ that maximizes the expected cumulative reward $G = \mathbb{E}_{\pi_i}[\sum_{t=0}^{\infty} \gamma^t \cdot r_t^i]$, where $\gamma$ is a discount factor.

### 2.2. Graph Attention Transformer (GAT)

GAT is designed to process graph-structured data, typically defined as $G = (V, E)$, where nodes $V$ represent entities and edges $E$ denote their interrelations. GAT employs attention mechanisms to dynamically adjust the significance of nodes within a graph and assigns variable weights to them. This capability enhances tasks such as node classification and link prediction by prioritizing relevant nodes. GAT enables more efficient handling of irregular data structures and better representation of correlations among data elements.

In this work, we utilize GATv2 (Brody et al., 2022) and describe it in the following context. Assuming a graph with $m$ nodes, we define $W \in \mathbb{R}^{d \times d'}$ as a node transformation matrix, where $d$ represents the dimension of the original entity features and $d'$ denotes the size of the output embedding. The embedding for node $i$, denoted as $h_i$, is determined by $h_i = W \cdot \nu_i$, where $\nu_i \in \mathbb{R}^d$ symbolizes the original

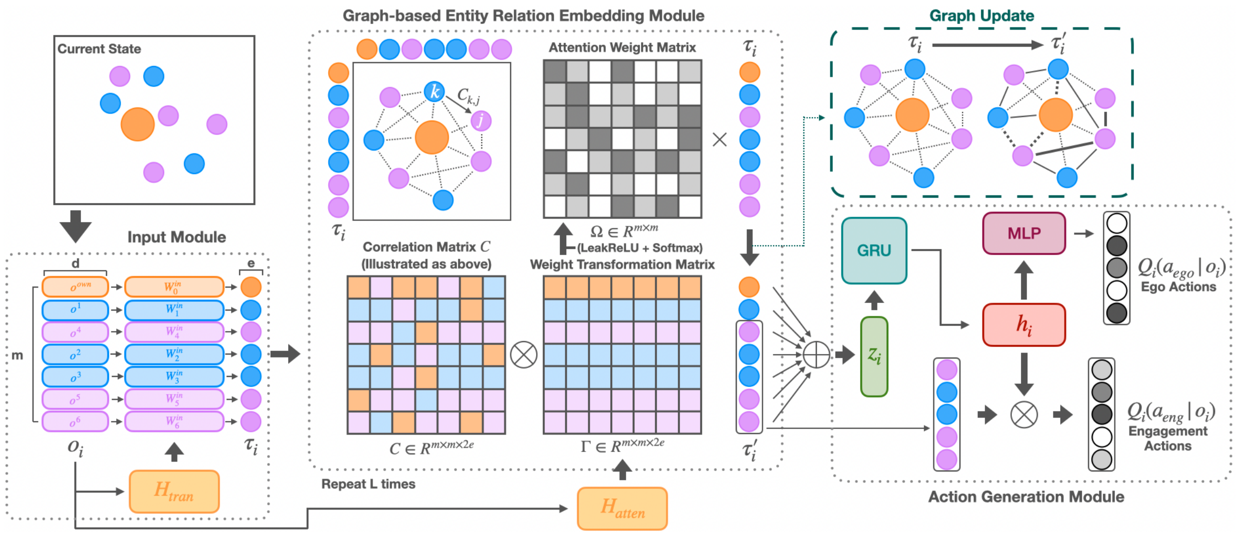


*Figure 3.* An overview of the proposed Hyper Graphical Attention Policy (HGAP) framework.

entity feature of node $i, i \in \{1, \cdots, m\}$. GATv2 employs a scoring function, $f_s : \mathbb{R}^{2d'} \to \mathbb{R}$, to assign a weight to every edge $(i, j)$, which reflects the significance of the features of neighbor $j$ to node $i$. The scoring function is defined as:

$$f_s(h_i, h_j) = LeakyReLU(\alpha^\top \cdot [h_i||h_j]), \qquad (1)$$

where $\alpha \in R^{2d'}$ is a learned vector, and $||$ denotes vector concatenation. The attention weight $\omega_{i,j}$ can be derived as:

$$\omega_{i,j} = Softmax(f_s(h_i, h_j)) = \frac{exp(f_s(h_i, h_j))}{\sum_{k=1}^{m} f_s(h_i, h_k)}, \qquad (2)$$

With the scoring process, GATv2 calculates a new representation for each node $i$ by taking a weighted average of the transformed features of its neighboring nodes followed by a nonlinearity transformation $\Psi$. This calculation uses the normalized attention coefficients obtained from the scoring function. Specifically, for a node $i$, GATv2 updates it as $h'_i[k] = \Psi(\sum_{j=1}^{m} \omega_{k,j} \cdot h_i[j])$, where $[k]$ is the $k^{th}$ element.

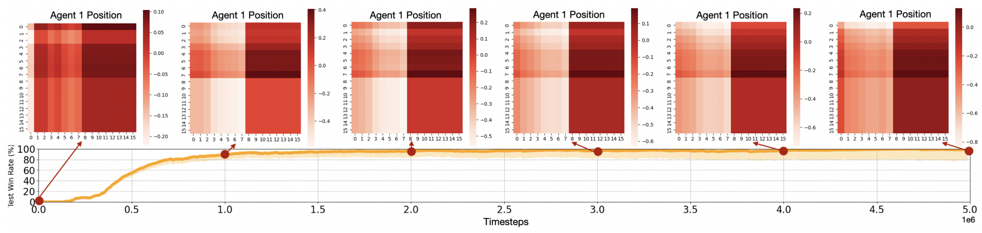
## 3. Methodology

In this section, we begin with an overview of the HGAP framework and its components. Subsequently, we introduce the graph-based entity relation embedding module, a key element of HGAP. Following this, we delve into the action generation module, which generates ego and engagement actions. Lastly, we discuss the loss function used by HGAP.

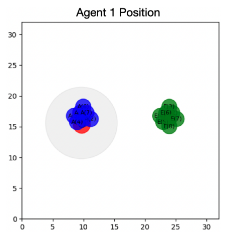
### 3.1. Hyper Graphical Attention Policy Network (HGAP)

Fig 3 presents an overview of the HGAP framework for MARL. Assuming an environment that contains $m$ entities, including the agent itself, the objective of HGAP is to enable an agent to efficiently learn an optimal policy while still possessing the properties of PI, PE, and utilizing GATv2 to represent the relationships between an agent and different entities. HGAP comprises four major components: (1) an Input Module $\tau_i = I(o_i)$ that transforms a raw observation $o_i$ into a high-dimensional embedding $\tau_i$, (2) a Graph-based Entity Relation Embedding Module that represents the relationship of the entities within a graph as a latent embedding $z_i$, fostering a more comprehensive representation of the relationships between agents and entities for improved decision-making, and (3) an Action Generation Module that recurrently embeds the past latent embedding as $h_i = GRU(z_i)$ to generate the state-action values for ego actions, $Q_i(a_{ego}|o_i)$, and engagement actions, $Q_i(a_{eng}|o_i)$. Ego actions primarily affect the agent itself, without directly influencing other entities. Engagement actions, on the other hand, are actions an agent takes that directly affect or target other entities. The design of the HGAP framework ensures consistent behavior (PI) as well as facilitating the learning of mapping relationships between entities and actions (PE).

Given an input $o_i$ containing features of $m$ entities, HGAP applies $m$ independent weight matrices generated by a hypernetwork, denoted as $H_{tran}$, for transforming entity features. These matrices are represented as $W_{in} = [W_0^{in}, \cdots, W_{m-1}^{in}]^\top$. The output is described as $\tau_i[j] = \sum_{j=0}^{m-1} W_j^{in} o_i[j]$, where $[j]$ indicates the j-th element. The transformed features in our framework are designated as entity embeddings, collectively denoted by $\tau_i$. These embeddings are input into the Graph-based Entity Relation Embedding Module, detailed in Section 3.2, to generate the latent embedding $z_i$. In line with prior methods such as QMIX (Vinyals et al., 2017), MADDPG (Lowe et al., 2017), and MAPPO (Yu et al., 2022), HGAP emploies a Gated Recurrent Unit (GRU) (Chung et al., 2014) to produce the hidden embedding $h_i$. This embedding incorporates previ-



(a)



(b)

*Figure 4.* (a) Attention heatmap variation during training in SMAC 3s5z map at the initial state. (b) Initial position for all entities in 3s5z.

ous history to provide temporal information and serves as a crucial component for the Action Generation Module to generate the state-action values, as discussed in Section 3.3.

### 3.2. Graph-based Entity Relation Embedding Module

To identify the correlations between the agent and its observed entities, the Graph-based Entity Relation Embedding Module takes $\tau_i \in \mathbb{R}^{m \times e}$ as input, where $m$ is the number of entities and $e$ represents the feature dimension of each entity within the embedding $\tau_i$. It then constructs an $m \times m$ Correlation Matrix $C \in \mathbb{R}^{m \times m \times 2e}$, as depicted in Fig. 3, where each entry $C_{k,j} \in C$ is the concatenation of $\tau_i[k]$ and $\tau_i[j]$, where $k, j \in m$. To facilitate GATv2 updates, $C$ must be transformed into an Attention Weight Matrix $\Omega \in \mathbb{R}^{m \times m}$ for performing node updates, as described in Section 2.2, where nodes in GATv2 correspond to the entity embeddings $\tau_i[k], k \in m$ in HGAP. This transformation is achieved with another Weight Transformation Matrix $\Gamma \in R^{m \times m \times 2e}$, where each entry of $\Omega$ is $\Omega_{i,j} = \Gamma_{i,j}^\top C_{i,j}$.

HGAP can then perform graph update operations as follows:

$$\tau_i'[k] = \sum_{j=1}^{e} \Omega_{k,j} \times \tau_i[j]. \tag{3}$$

As the dimensions of $\Omega$ are determined by the number of entities, adjustments are necessary when transitioning to new environments, thus preventing it from being fixed-sized. As a result, HGAP employs a hypernetwork, $H_{atten}$, to generate $\Gamma$, where each element of $\Gamma$ is computed as $\Gamma_{i,k} = H_{atten}(o^{i,k}), \forall k \in [1, m]$. The embedding $z_i$ is obtained by aggregating the updated entity embeddings $\tau_i'$ as follows:

$$z_i = Aggregate(\tau_k' | k \in [1, m]). \tag{4}$$

Our experimental results, as depicted in Fig. 4, validate the effectiveness of the Graph-based Entity Relation Embedding Module in enabling the agent to quickly identify enemies within a brief training duration. As training progresses, Fig. 4 demonstrates the efficiency in the agent's ability to enhance its proficiency in focusing on its enemies (given the agent's role as an attacking unit) and to effectively identify correlations within a short training period. The findings substantiate the effectiveness of the proposed graph-based module in facilitating more uniform and efficient training.

### 3.3. Action Generation Module

The function of the Action Generation Module is to utilize the embedding $z_i$ to derive state-action values for ego and engagement actions. An embedding $h_i$ generated by a GRU retains the temporal information for the agent, which can be used to derive the current action based on prior temporal cues. Since $z_i$ is obtained through an aggregation function, it ensures that the entry order of the entity embeddings in $\tau_i'$ does not affect the outcome. This further ensures that HGAP satisfies the PI property and can utilize $h_i$ to derive ego actions subsequently. Therefore, we employ a fully connected layer to map $h_i$ to the corresponding state-action values for ego actions as $Q_i(a_{ego}|o_i)$ for agent $i$. On the other hand, directly using $h_i$ to derive engagement actions without considering entity information and their embedding orders would not fulfill the requirement of PE. As a result, to derive the mapping relationship that links engagement actions with observed entities, HGAP adopts an approach inspired by Wang et al. (2020b), in which a pairwise function is used to generate the corresponding action-state values. Specifically, the state-action value for the $j^{th}$ engagement action is computed as $Q_i(a_{eng}|o_i)[j] = \tau_i'[j]^\top h_i$. The topological interpretation of this equation can be understood as the cosine angle between $h_i$ and the entries of $\tau'$. When they are proximate, the cosine angle nears zero, indicating a high degree of correlation and suggesting that the corresponding action should be given more consideration. The state-action values are processed through a Softmax layer, and then the action with the highest probability is selected for execution.

### 3.4. Loss Function

To optimize HGAP, we employ the temporal difference (TD) error (Sutton & Barto, 2018) as the objective, same as the approach employed in QMIX (Rashid et al., 2018). The loss function used during training can be formulated as follows:

$$L(\theta) = \sum_{i=1}^{b} [(y_i^{tot} - Q_{tot}(\tau; u, s; \theta))^2], \tag{5}$$

where $\theta$ and $\theta^-$ are the parameters of the agent and target networks, respectively. $\theta^-$ is periodically copied from $\theta$. b is the batch size, and $y_i^{tot} = r + max_{u'} Q(\tau; u', s'; \theta^-)$.

## 4. Experimental Results

In this section, we begin by detailing our experimental setups. This is followed by a comparison of HGAP against various state-of-the-art (SOTA) methods. Then, we provide a series of ablation studies to validate HGAP's effectiveness. These studies include applying HGAP within different MARL frameworks, assessing its transferability, and evaluating its comprehension of invisible entities under POMDP.

### 4.1. Experimental Setups

**Environments.** We evaluate the performance of HGAP on three representative MARL environments: Multiple Particle Environments (MPE) (Mordatch & Abbeel, 2017), SMAC (Vinyals et al., 2017), and SMACv2 (Ellis et al., 2023). MPE is tailored explicitly for MARL, wherein agents are represented as elementary particles within physics-based simulations. The inherent simplicity of these basic environments not only eliminates the complexities found in more realistic scenarios but also enables a concentrated examination of essential elements related to MARL agent interactions.

SMAC serves as an evaluation platform for cooperative MARL algorithms, which utilizes the strategy game StarCraft II as its foundation. The scenarios involve both agent and enemy units, with the objective of eliminating the enemy within a predefined duration. Agents that are guided by MARL algorithms interact with adversaries governed by the game's built-in AI heuristics. SMAC evaluates agent performance based on a global reward computed at each timestep, taking into account factors such as the number of defeated enemies, inflicted damage, and the agents' overall success. The scenarios in SMAC contain *Easy*, *Hard*, and *Super-Hard* difficulty levels defined by Vinyals et al. (2017).

SMACv2 is an evolved extension of SMAC and features three significant changes: (1) randomized initial positions, (2) randomized entity types, and (3) adjusted unit sight and attack ranges. These introduce greater randomness and diversity into the scenarios, and the purpose is to address the previous limitations in SMAC concerning the assessment of algorithmic robustness. In our experiments, we employ five independent random seeds across all the environments to ensure the robustness and reliability of our evaluations.

**Observation and Action Space.** In MPE, the action space for each agent encompasses six potential operations: no operation (no-op), pressing a button, and movements in four directions (up, right, down, and left). The observation space for these agents includes individual features (e.g., velocity and position) and positional information about other entities.

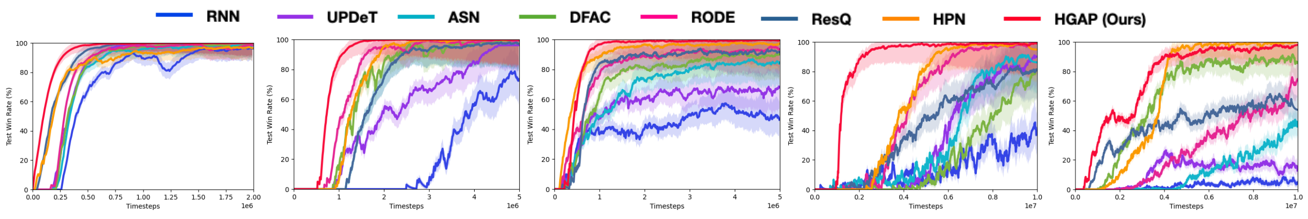
In the SMAC and SMACv2 environments, the action space for each agent is divided into two primary components: (i) stop, no-op, and the four directions of movement (north, south, east, and west), and (ii) attacks on the enemies. An agent can attack an enemy if two conditions are met: (1) the targeted enemy is alive, and (2) the enemy is both visible to the agent and falls within its attack range. The observation space in these environments includes the agent's features, such as the ability to move in four directions and its current health status. Furthermore, it includes features of other entities, such as visibility, health, normalized relative x-y coordinates, and distance from the agent. In adherence to the principles of POMDP, entities outside an agent's sight range are represented as zero in the corresponding observation.

**Baselines.** A wide range of CTDE-based MARL methodologies have been selected to evaluate different agent network design strategies. These include the recurrent neural network (RNN) (Hausknecht & Stone, 2015) agent, the attention-based UPDeT (Hu et al., 2021), and the DFAC (Sun et al., 2021) and ResQ (Shen et al., 2022) networks, which represent distributional RL strategies. In addition, RODE (Wang et al., 2020a) is included to represent role-based methodologies. Considering previous PI and PE approaches, ASN (Wang et al., 2020b) and HPN (Hao et al., 2023) have also been incorporated. All these agents employ QMIX as the factorization network for a fair comparison.

To further demonstrate the adaptability of HGAP, a series of ablation studies are conducted. First, HGAP is integrated into action-critic frameworks, which include MADDPG (Lowe et al., 2017), DA-MADDPG (Ye et al., 2023), and PIC (Liu et al., 2019). Second, QMIX is replaced with QPLEX (Wang et al., 2021a) to verify the robustness of HGAP's performance across different factorization networks. Lastly, HGAP is implemented in a policy-based method, specifically MAPPO (Yu et al., 2022). We also evaluate HGAP alongside other PI/PE methodologies. For methods considering only the PI property, we include DA-MADDPG, Set (Lee et al., 2019), and GNN (Liu et al., 2019). Methods incorporating both PI and PE properties include UPDeT, ASN, and HPN. All experiments are evaluated across five distinct random seeds, with shaded areas in the figures representing the training variance. The hyperparameters for each method are configured based on the optimal settings reported in their respective original papers, with their detailed information available in Appendix F.

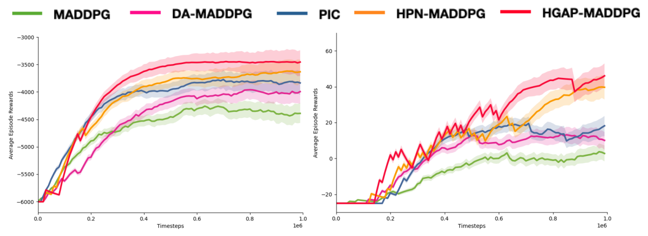
### 4.2. Agent Network Comparison

We explore a variety of agent network design strategies for CTDE-based MARL methods, all of which utilize the QMIX value factorization network for training. The results of this experiment are illustrated in Fig. 5. It is observed that agents (i.e., HGAP and HPN) considering both PI and PE properties achieve higher test win rates compared to the other baselines, particularly outperforming those based on distributional RL and role-based strategies. Moreover, as the complexity of the scenarios increases, HGAP exhibits a more rapid rate of convergence compared to the other



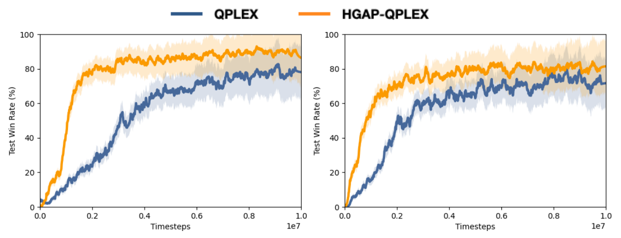
(a) 1c3s5z (E) (b) 3s_vs_5z (H) (c) 5m_vs_6m (H) (d) 3s5z_vs_3s6z (S) (e) zerg_10_vs_11 (v2)

*Figure 5.* Test win rate comparison for different CTDE-based MARL approaches in SMAC and SMACv2 for comparing the agent network design strategies. (E), (H), (S) stand for *Easy*, *Hard*, and *Super-Hard* scenarios in SMAC, and (v2) denotes the scenario from SMACv2.



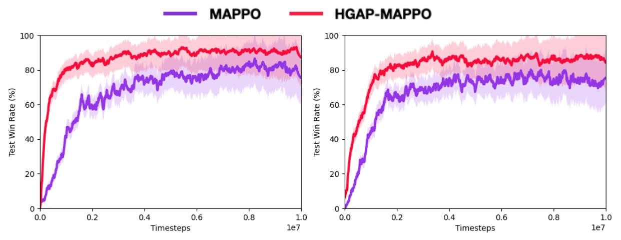
(a) Navigation (N=10) (b) Predator-Prey (N=3)

*Figure 6.* Test win rate comparison for MADDPG variants in MPE for validating HGAP's adaptability to actor-critic based methods.



(a) protoss_10_vs_11 (b) terran_20_vs_23

*Figure 7.* Test win rates of QPLEX & HGAP-QPLEX in SMACv2.



(a) terran_10_vs_11 (b) zerg_20_vs_23

*Figure 8.* Test win rates: MAPPO & HGAP-MAPPO in SMACv2.

baselines. This indicates that HGAP is effective and robust across a range of scenarios, even the more challenging ones from SMACv2, which features increased stochasticity and difficulty. Additional results are offered in Appendix G.1.

## 4.3. Ablation: Adaptability of HGAP

In this section, the integration of HGAP with multiple MARL methods is presented to verify HGAP's adaptability.

### 4.3.1. Actor-Critic Network Adaptability

We first explore the application of HGAP within the MADDPG framework, which results in a new variant termed HGAP-MADDPG. This analysis compares HGAP-MADDPG with the existing MADDPG on MPE. The learning curves presented in Fig. 6 demonstrate that HGAP-MADDPG achieves superior performance when benchmarked against HPN-MADDPG (Hao et al., 2023), PIC, DA-MADDPG, and MADDPG. We examine HGAP-MADDPG applied in large population environments in Appendix G.2.

### 4.3.2. Factorization Network Adaptability

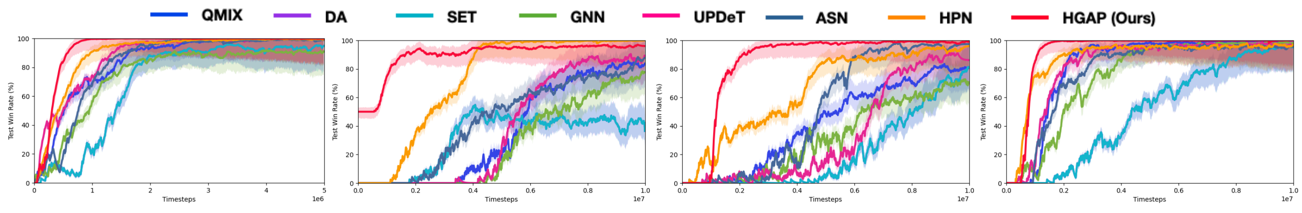
We next examine the efficacy and adaptability of HGAP in conjunction with the QPLEX framework within the SMACv2 environments. This variant is termed HGAP-QPLEX. The results of this validation are illustrated in Fig. 7. Based on these results, it can be observed that integrating HGAP with QPLEX enhances the test win rates and achieves better efficiency than the conventional QPLEX configuration. Further detailed results and discussions are presented in AppendixG.3. This substantiates HGAP's potential to improve the robustness and effectiveness of other factorization networks in complex, dynamic environments.

### 4.3.3. Policy-based Network Adaptability

Lastly, the integration of HGAP within the standard MAPPO learning framework is evaluated in the context of SMACv2, and this variant is termed HGAP-MAPPO. The test win rates are depicted in Fig. 8. The results suggest that HGAP enhances MAPPO and can yield improved performance and training efficiency in policy-based approaches. This finding demonstrates HGAP's adaptability and effectiveness in improving policy-based multi-agent learning strategies. For additional analysis results, please refer to AppendixG.4.

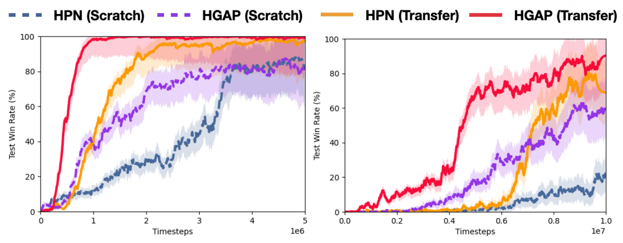
## 4.4. Ablation: PI and PE Baselines Comparison

In this section, we compare HGAP with several baselines that consider PI and PE. The PI-only baselines include: (i) DA-MADDPG (DA), which enhances MADDPG by shuffling the input order for additional training data, as described by (Ye et al., 2023); (ii) Set, which applies the Set Transformer to QMIX's state-action value functions for entity embedding aggregation (Zaheer et al., 2017); and (iii) GNN, which implements GNN (Liu et al., 2019) on its state-action value functions within QMIX. The other baselines that consider both PI and PE (i.e., UPDeT, ASN, and HPN) are adopted without modification from their respective original



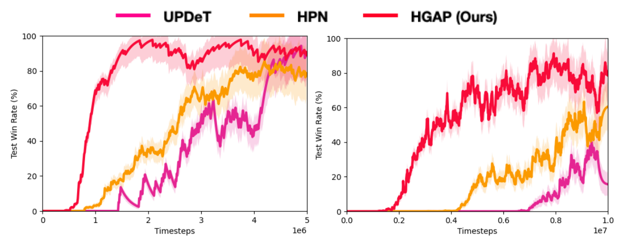
(a) 10m_vs_11m (H) (b) 6h_vs_8z (S) (c) corridor (S) (d) MMM2 (S)

*Figure 9.* Test win rates for the PI/PE baselines in SMAC. A detailed introduction to the PI/PE baselines can be found in Appendix D.



(a) (b)

*Figure 10.* Transfer learning comparison between HPN and HGAP in SMAC. The dashed lines represent the training of agents from scratch in the target environments without pretraining, while the solid lines indicate agents pre-trained in one scenario and then transferred to the target scenario. (a) 5m_vs_6m (H) → 10m_vs_11m (H); and (b) 3s5z (H) → 3s5z_vs_3s6z (S).



(a) 3s5z (H) (b) corridor (S)

*Figure 11.* A comparison of UPDeT, HPN, and HGAP in SMAC under special setting that the enemies have full environmental information, while the agent and allies are constrained by POMDP.

publications. The results evaluated in SMAC, shown in Fig. 9, indicate that the performance of HGAP surpasses HPN and the other baselines. It can be observed that considering both PI and PE in the agent network design can indeed enhance training performance compared with the baselines that do not consider both, such as DA, Set, and GNN. HGAP's superior performance over HPN and the others is attributed to its design, which effectively captures inter-entity relationships, thus providing more comprehensive information for HGAP to perform decision-making.

### 4.5. Transfer Learning Comparisons

The flexible architecture design of HGAP suggests that it can also be as a transferable framework, which allows HGAP to potentially expedite the adaptation of a pre-trained MARL policy to more complex scenarios with enhanced learning efficiency. To validate this, we conduct an experiment with HGAP and incorporat HPN as a baseline across various transfer learning scenarios. We evaluat the transferability of both HGAP and HPN using SMAC, with the results presented in Fig. 10. The findings indicate that HGAP achieves at least a 60% reduction in training time compared to training agents from scratch. This demonstrates HGAP's capability to efficiently transfer pre-trained policies to new and more challenging scenarios. Additional results on HGAP's transferability are available in Appendix G.5.

### 4.6. Competing Against Full-Information Opponents

We present an interesting experiment for a scenario in which the opponents have access to complete environmental information, while the agent and its allies operate under the POMDP constraint with limited sight range. As a result, certain entities remain unobservable, whereas the enemies enjoy an unlimited sight range. Many previous MARL studies have treated invisible entities as non-existent, which constrains their representation capability of the environmental information. In contrast, HGAP employs a graph-based representation strategy to enable the agent to implicitly infer potential information associated with these invisible entities. We compare HGAP, HPN, and UPDeT under such constraints. The results presented in Fig. 11 indicate that HGAP exhibits superior evaluation performance in such a setting due to its graph-based representation mechanism.

## 5. Conclusions

This study introduced HGAP, an agent network designed for MARL that employs a graph attention mechanism to encode the relationships among entities in an environment, while maintaining the properties of PI and PE. The graph-based embedding mechanism of HGAP further enables it to address the constraints of POMDP by inferring information from the graph. The experimental results conducted on MPE, SMAC, and SMACv2 validated HGAP's superiority over various CTDE-based MARL methodologies. Furthermore, HGAP's adaptability to different MARL framework designs is demonstrated, including its adaptability to actor-critic methods, a different value factorization frameworks, and a policy-based method. We also explored transfer learning scenarios to show that HGAP's architecture allows for the efficient transfer of a pre-trained policy to new or more challenging scenarios faster than the other baselines. Finally, we present a compelling experiment demonstrating that, even when enemies have full sight range and the agent and its allies operate under POMDP constraints, the graph-based embedding mechanism can still enable HGAP to outperform the baselines under this constrained and challenging setup.

## Acknowledgment

The authors gratefully acknowledge the support from the National Science and Technology Council (NSTC) in Taiwan under grant number MOST 111-2223-E-007-004-MY3 and the financial support from MediaTek Inc., Taiwan. The authors would also like to express their appreciation for the donation of the GPUs from NVIDIA Corporation and NVIDIA AI Technology Center (NVAITC). Moreover, the authors extend their gratitude to the National Center for High-Performance Computing (NCHC) in Taiwan for providing the necessary computational and storage resources.

## Impact Statement

This paper presents research aimed at utilizing graph representation and PI/PE properties to enhance agent policy learning in multi-agent reinforcement learning with better training efficiency. Our methodology can be easily integrated into various MARL approaches and offers better performance. While our research does not have any social consequences, the potential social impact is positive, as it can contribute to advancements in robotic applications.

## Limitation

One of the limitations of the current HGAP framework could be the absence of coach-based methodologies to enhance agent learning and performance. The integration of a coach agent that provides critiques or strategic advice during training could potentially improve the learning process and the overall performance of the agents. Furthermore, incorporating human-in-the-loop coaching, where human experts offer real-time guidance, and curriculum learning techniques that gradually increase task complexity under a coach's supervision, may facilitate more structured agent learning. However, realizing effective coach-based learning within the HGAP framework would require the development of new architectures and algorithms to seamlessly integrate coaching feedback and demonstrations. Techniques such as imitation learning, inverse reinforcement learning, or hybrid coach-guided reinforcement learning approaches need to be explored to fully leverage the potential benefits of coach-based methods. By addressing these limitations and combining HGAP with coach-based methods, a crucial future direction of this work is to improve agent capabilities, boost performance in complex environments, enable more efficient multi-agent decision-making, and foster collaboration between artificial agents and human domain experts. This direction holds promise for tackling challenging real-world problems and will be a focus of our future work.

## References

Bao, L., Ma, B., Chang, H., and Chen, X. Masked graph attention network for person re-identification. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2019.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks? In *Int. Conf. on Learning Representations (ICLR)*, 2022.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proc. Conf. on Neural Information Processing Systems Workshop (NeurIPSW)*, 2014.

Dosovitskiy, A., Beyerm, L., Kolesnikov, A., Weissenborn, D., Zhai, X., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. on Learning Representations (ICLR)*, 2021.

Ellis, B., Cook, J., Moalla, S., Samvelyan, M., Sun, M., and et al. SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. arXiv:2212.07489, 2023.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Proc. Conf. on Artificial Intelligence (AAAI)*, 2018a.

Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H. S., and et al. Stabilising experience replay for deep multi-agent reinforcement learning. *arXiv:1702.08887*, 2018b.

Hao, J., Hao, X., Mao, H., Wang, W., Yang, Y., et al. Boosting multiagent reinforcement learning via permutation invariant and permutation equivariant networks. In *Int. Conf. on Learning Representations (ICLR)*, 2023.

Hausknecht, M. J. and Stone, P. Deep recurrent Q-learning for partially observable mdps. In *Proc. Conf. on Artificial Intelligence (AAAI)*, 2015.

Hu, S., Zhu, F., Chang, X., and Liang, X. UPDeT: Universal multi-agent reinforcement learning via policy decoupling with transformers. In *Int. Conf. on Learning Representations (ICLR)*, 2021.

Iqbal, S. and Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2019.

Jiang, J. and Lu, Z. Learning attentional communication for multi-agent cooperation. In *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*, 2018.

Jiang, J., Dun, C., Huang, T., and Lu, Z. Graph convolutional reinforcement learning. In *Int. Conf. on Learning Representations (ICLR)*, 2020.

Kim, D., Moon, S., Hostallero, D., Kang, W. J., Lee, T., et al. Learning to schedule communication in multi-agent reinforcement learning. In *Int. Conf. on Learning Representations (ICLR)*, 2019.

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

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., et al. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *Prof. Int. Conf. on Machine Learning (ICML)*, 2019.

Liu, I. J., Yeh, R. A., and Schwing, A. G. PIC: Permutation invariant critic for multi-agent deep reinforcement learning. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2019.

Lowe, R., WU, Y., Tamar, A., Harb, J., Abbeel, P., O., et al. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. In *Int. Conf. on Learning Representations (ICLR)*, 2019.

Meng, L., Wen, M., Yang, Y., Le, C., Li, X. Y., et al. Offline pre-trained multi-agent decision transformer, 2022.

Mordatch, I. and Abbeel, P. Emergence of grounded compositional language in multi-agent populations. *arXiv:1703.04908*, 2017.

Oliehoek, F. A. and Amato, C. *A Concise Introduction to Decentralized POMDPs*. Springer Publishing Company, Incorporated, 2016.

Oliehoek, F. A., Spaan, M. T. J., and Vlassis, N. Optimal and approximate Q-value functions for decentralized pomdps. *arXiv:1111.0062*, 2008.

Park, C., Lee, C., Bahng, H., Tae, Y., Jin, S., et al. ST-GRAT: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed, 2022.

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2018.

Rashid, T., Farquhar, G., Peng, B., and Whiteson, S. Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. 33, 2020.

Shen, S., Qiu, M., Liu, J., Liu, W., Fu, Y., et al. ResQ: A residual Q function-based approach for multi-agent reinforcement learning value factorization. In *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*, 2022.

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2019.

Son, K., Ahn, S., Reyes, R. D., Shin, J., and Yi, Y. Qtran++: Improved value transformation for cooperative multi-agent reinforcement learning. *arXiv:2006.12010*, 2020.

Sun, W. F., Lee, C. K., and Lee, C. Y. DFAC framework: Factorizing the value function via quantile mixture for multi-agent distributional q-learning. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2021.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., and et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proc. Int. Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction, Second Edition*. MIT Press, Cambridge, MA, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., et al. Attention is all you need. In *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and et al. Graph attention networks. In *Int. Conf. on Learning Representations (ICLR)*, 2018.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., et al. Starcraft II: A new challenge for reinforcement learning. arXiv:1708.04782, 2017.

Wang, J., Ren, Z., Liu, T., Yu, Y., and Zhang, C. QPLEX: duplex dueling multi-agent q-learning. In *Int. Conf. on Learning Representations (ICLR)*, 2021a.

Wang, T., Dong, H., Lesser, V., and Zhang, C. ROMA: Multi-agent reinforcement learning with emergent roles. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2020a.

Wang, T., Gupta, T., Mahajan, A., Peng, B., Whiteson, S., et al. RODE: Learning roles to decompose multi-agent tasks. In *Int. Conf. on Learning Representations (ICLR)*, 2021b.

Wang, W., Yang, T., Liu, Y., Hao, J., Hao, X., et al. From few to more: Large-scale dynamic multiagent curriculum learning. In *Proc. Conf. on Artificial Intelligence (AAAI)*, 2019.

Wang, W., Yang, T., Liu, Y., Hao, J., Hao, X., et al. Action semantics network: Considering the effects of actions in multiagent systems. In *Int. Conf. on Learning Representations (ICLR)*, 2020b.

Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., et al. Mean field multi-agent reinforcement learning. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2018.

Yang, Y., Hao, J., Liao, B., Shao, K., Chen, G., et al. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv:2002.03939*, 2020a.

Yang, Y., Wen, Y., Chen, Y., Wang, H., Shao, K., et al. Multi-agent determinantal Q-learning. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2020b.

Ye, Z., Chen, Y., Song, G., Yang, B., and Fan, S. Experience augmentation: Boosting and accelerating off-policy multi-agent reinforcement learning. In *Int. Conf. on Learning Representations (ICLR)*, 2023.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, T., et al. The surprising effectiveness of PPO in cooperative multi-agent games. In *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*, 2022.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., et al. Deep sets. In *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

## Appendix

This appendix aims to provide additional information that includes a summary of notations for the main manuscript, background materials for various value factorization method baselines, and the foundational concept of the Transformer. It also supplements the main manuscript with additional experimental results. Section A summarizes the notations used throughout the paper. More specifically, Section B offers an in-depth discussion on contemporary value factorization methodologies. In Section C introduces various aspects of MARL approaches. Furthermore, we also demonstrate some PI and PE MARL methodologies in Section D. Section E briefly describes the concept of the Transformer model and the attention mechanism. Section F details the hyperparameters employed in our experiments. Moreover, Section G provides additional experimental results to complement those presented in the main manuscript. Lastly, we provide more graph attention heatmaps to validate the effectness of GATv2 in Section H.

## A. Notation Table

To enhance clarity and facilitate understanding of the notations used throughout our paper, Table A1 provides a comprehensive summary of these notations.

*Table A1.* Notation table

| **Notation Descriptions** | | |
|---|---|---|
| | Dec-POMDPs Notations | |
| $\mathbf{N}$ | Set of agents in the environments | $i \in \mathbf{N}$ |
| $\mathbb{O}$ | Observation space | $o_i \in \mathbb{O}$ |
| $\mathbb{S}$ | Global state space | $s \in \mathbb{S}$ |
| $\mathbb{A}$ | Action space | $u_t^i \in \mathbf{A}_i, \forall i \in \mathbf{N}$ |
| $\mathbb{U}$ | Joint action space | $\mathbb{U} = \Pi_{i \in N} A_i$ |
| $\mathbf{T}$ | State transition function defining the probability from $s$ to $s'$ by taking $u$, where $u \in U$ | $P(s' \mid s, u)$ |
| $\mathbf{R}$ | Reward function | $\mathbf{R}(s, u)$ |

## B. Background of the Value Factorization Methods

In this section, we elaborate on the core concepts of the mixing networks that we have selected for comparison with our proposed methodology in the manuscript. These networks include VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2018), QTRAN (Son et al., 2019), Qatten (Yang et al., 2020a), and QPLEX (Wang et al., 2021a). The explanation aims to provide an in-depth comprehension of their function and enable a meaningful comparison with our proposed approach.

The notations used in this section are defined as follows. $N$ is the total number of agents and $i \in N$ represents the index of the agent. The joint history $h = \langle h_1, ..., h_N \rangle$ concatenates the action-observation history of each agent, where $h_i$ denotes the individual action-observation history for agent $i$. The joint action $u = \langle u_1, ..., u_N \rangle$ is taken with joint history $h$ to estimate the joint action-value function $Q_{jt}(h, u)$ at every timestep $t$, where $u_i$ is the individual action taken by agent $i$.

### B.1. Value Decomposition Network (VDN)

In VDN (Sunehag et al., 2018), the value factorization network, often referred to as the mixing network, operates under the assumption of *additivity* in the value function. It represents the total return, denoted as $Q_{jt}$, as a sum of individual utility functions. Each of these functions is derived from the corresponding agent's individual trajectory and action. This concept can be represented as follows:

$$Q_{jt}(h, u) = \sum_{i=1}^{N} Q_i(h_i, u_i)\,. \tag{6}$$

The contribution of VDN lies in decomposing complex joint learning problems into smaller, more manageable sub-problems. By value factorization, it simplifies the learning process of MARL agents.

## B.2. QMIX

QMIX (Rashid et al., 2018) extends the concept of value factorization networks from VDN and employs a network to estimate the joint action-value function $Q_{jt}$. This function is modeled as a non-linear combination of individual utility functions $Q_i(h_i, u_i)$.

The formulation of $Q_{jt}$ can then be expressed as follows:

$$Q_{jt}(h, u) = \Phi(Q_1(h_1, u_1), Q_2(h_2, u_2), ..., Q_N(h_N, u_N)), \tag{7}$$

where $\Phi$ represents the monotonic function following the constraint $\frac{\partial \Phi}{\partial Q_i} \geq 0$. The weights of the mixing network in QMIX originate from separate hypernetworks. Each hypernetwork accepts the state $s$ as input and generates the weights for a specific layer within the mixing network. The generated weights are also enforced to remain non-negative, which ensures that QMIX adheres to the monotonicity constraint. The outputs of the hypernetworks are subsequently reshaped into matrices of appropriate dimensions. On the other hand, the biases are generated using a similar approach but are not subjected to non-negativity constraints, as described in the original manuscript of QMIX.

## B.3. QTRAN

While VDN and QMIX have exhibited impressive performance in a number of MARL benchmarks, they face challenges when encountering functions that deviate from additivity or monotonicity. These constraints impose limitations on the representational capacity of the mixing network. In scenarios where the environment does not adhere to these constraints, VDN and QMIX often encounter difficulties in appropriately factorizing the joint Q-value. To address these difficulties, QTRAN (Son et al., 2019) proposed that a factorizable joint action-value function should conform to the following equation:

$$\sum_{i=1}^{N} Q_i(h_i, u_i) - Q_{jt}(h, u) + V_{jt}(h) = \begin{cases} 0, & u = \bar{u} \\ \geq 0, & u \neq \bar{u} \end{cases}, \tag{8}$$

where $V_{jt}(h) = \max_u Q_{jt}(h, u) - \sum_{i=1}^{N} Q_i(h_i, \bar{u}_i)$, and $\bar{u}_i$ represents the optimal local action for agent $i$. The joint state value function $V_{jt}(h)$ incorporates joint observation-action history information. This enhancement, driven by the partial observability from each agent, offers QTRAN an edge in terms of value representation capabilities, and therefore, better performance, over VDN and QMIX.

## B.4. Multi-Head Attention based Q-Value Mixing Network (Qatten)

QTRAN is introduced to ensure optimal decentralization. It preserves the additive assumption and mitigates the representational constraints associated with VDN and QMIX. However, QTRAN presents a computationally intractable optimization problem. Recently, modules based on attention mechanisms have exhibited outstanding performance across various fields. In Qatten (Yang et al., 2020a), the authors offer a theoretical analysis of the capacity of attention modules for joint action-state value representation. This consideration takes into account a scenario with a continuous action space and the absence of independent agents. Given these assumptions, it's possible to establish the existence of constants, denoted as $c(s)$, dependent on the state $s$, which plays as the bias term. The joint action-value function, $Q_{jt}$, can be expressed as:

$$Q_{jt}(h, \vec{u}) \approx c(s) + \sum_{k=1}^{K} w_k \sum_{i=1}^{N} \lambda_{i,k} Q_i(h_i, u_i), \tag{9}$$

where $\lambda_{i,k}$ represents the output from the $k$-th head of the attention layer, $k \in K$ denotes the index of the head in the attention layer utilized in Qatten, and $K$ is the total number of heads. In order to assign weight importance, Qatten attributes $w_k = |f^{NN}(s)|_k$ to different heads. Qatten uses hypernetworks to generate the weights associated with different attention heads in $w_k$, where these weights adjust the Q-values associated with different heads based on global states $s$. Instead of conducting self-attention between each pair of agents, Qatten transforms the global state information and joint actions into a query matrix and a key matrix for attention. It then concatenates the individual action-value functions and transforms them, resulting in the formation of a value matrix for attention. Qatten utilizes a multi-head attention mechanism within a mixing network that distinctly models each agent's influence on the overall system during the transformation of individual utilities into $Q_{jt}$.

### B.5. DuPLEX Dueling Multi-Agent Q-Learning (QPLEX)

The insufficiency of the joint value function class can detrimentally impact performance and pose a potential risk of training instability in offline settings. Several approaches have emerged to address this structural limitation. Among these methods, QTRAN employs two soft regularizations aiming at harmonizing the choice of greedy actions between joint and individual value functions. Nevertheless, due to computational constraints, the implementation of these regularizations is inherently approximate, relying on heuristics and providing no concrete guarantee of Individual-Global-Minimum (IGM) consistency. To deal with this issue, QPLEX (Wang et al., 2021a) introduces an alternative strategy that incorporates a duplex dueling network architecture to facilitate the factorization of the joint action-value function into individual utility functions. It formulates the joint action-value function $Q_{jt}$ as follows:

$$\begin{aligned} Q_{jt}(h,u) &= V_{jt}(h) + A_{jt}(h,u) \\ &= \sum_{i=1}^{n} Q_i(h_i,u_i) + \sum_{i=1}^{n} (\lambda_i(h,a)-1)A_i(h_i,u_i), \end{aligned} \tag{10}$$

$$\lambda_i(h,u) = \sum_{k=1}^{K} \lambda_{i,k}(h,u)\phi_{i,k}(\tau)v_k(h), \tag{11}$$

where $\lambda_i(h,u)$ represents the positive importance weights, which are generated through a multi-head attention layer using global state information and joint actions as inputs. $K$ denotes the number of heads, and $v_k(h) > 0$ signifies a positive key for each head. The QPLEX architecture comprises two key components: (i) individual utility functions $Q_i(h_i,u_i)$, each characterized by a recurrent network dedicated to a particular agent, and (ii) a duplex dueling network that merges these individual utility functions to create a joint action-value function $Qjt$, while conforming to the advantage-based IGM constraint. The architecture employs a dot-product operation between the importance weight $\lambda_i(h,u)$ and the corresponding individual advantage function $A_i(h_i,u_i)$. This is subsequently added to the joint state value function $V_{jt}(h) = \sum_{i=1}^{N} V_i(h)$, where $V_i(h) = \omega_i(h)V_i(h_i) + b_i(h)$. Please note that both $\omega_i(h)$ and $b_i(h)$ are the outputs of a multilayer perceptron (MLP) that uses the global state $s$ as input. This MLP acts as a transformation mechanism for converting individual $V_i(h_i)$ to $V_i(h)$.

## C. Introduction to Various MARL approaches

In recent years, the landscape of Multi-Agent Reinforcement Learning (MARL) has been enriched by the introduction of numerous methodologies, each exhibiting proficiency in various dimensions. This subsection aims to systematically categorize these methodologies into three principal classes, namely: Actor-Critic approaches, Value Function Factorization approaches, and Role-Based approaches.

### C.1. Actor-Critic approaches

In multi-agent reinforcement learning, it is difficult to estimate the value function since each agent sees the environment as constantly changing. A recent method, called multi-agent deep deterministic policy gradient (Lowe et al., 2017), addresses this by using a centralized critic. This critic combines the observations and actions of all agents. Similarly, other approaches like (Foerster et al., 2018a;b; Iqbal & Sha, 2019; Jiang & Lu, 2018; Kim et al., 2019; Yang et al., 2018; Ye et al., 2023) also use centralized critics to manage the changing environment.

### C.2. Value function factorization approaches

Recent advancements in multi-agent reinforcement learning often use the Centralized Training with Decentralized Execution (CTDE) framework, which is effective for scalability and real-world application. The key element in CTDE is the factorization network, which allocates the overall reward to each agent based on their collective decisions. VDN (Sunehag et al., 2018) was the first to employ factorization networks, using a summation method, but it had limitations in representing the capacity of joint action values. QMIX (Rashid et al., 2018) improved on this by introducing a monotonic constraint and using hypernetworks for greater capacity. However, QMIX struggles with representing nonmonotonic joint action value functions. To address this, various approaches have been proposed: QTRAN (Son et al., 2019) and QTRAN++ (Son et al., 2020) use soft regularizations to align individual and joint value function decisions; WQMIX (Rashid et al., 2020) applies a weighted projection emphasizing better joint actions; Qatten (Yang et al., 2020a) utilizes self-attention layers to understand joint action correlations; and QPLEX (Wang et al., 2021a) introduces a dueling structure [cite] for duplex action-value functions, redefining the IGM principle with an advantage-based approach. Another research area within value factorization networks is distributional RL. DMIX and DDN (Sun et al., 2021), extending QMIX and VDN, can distribute stochastic joint state-action values into individual agent utilities, adhering to the DIGM principle. ResQ (Shen et al., 2022) further innovates by using masking for non-monotonic mean joint value functions, fulfilling both IGM and DIGM principles without limitations in representation.

### C.3. Role-based approaches

Role-based learning offers a scalable approach to multi-agent learning by breaking down complex tasks into distinct roles. ROMA (Wang et al., 2020a) integrates the concept of roles into MARL, allowing agents with similar duties to share their learning experiences. Additionally, it incorporates two regularizers to ensure that roles are identifiable through specific behaviors and are specialized for particular sub-tasks. Building on this idea, RODE (Wang et al., 2021b) proposes that discovering roles can be more straightforward by initially dividing the joint action spaces based on the functionality of actions, instead of learning roles from the beginning.

## D. Introduction to Permutation Invariant (PI) and Permutation Equivariant (PE) approaches in MARL

### D.1. Data Augmentation

To minimize environmental interactions, DA-MADDPG (Ye et al., 2023) introduced a data augmented version of MADDPG. This method creates additional training data by rearranging the order of inputs and then updates the model using this new data. While effective, this approach demands more computational resources and takes longer. Moreover, the generated data, having identical information to the original, should yield the same Q-value. However, training a function sensitive to input order to produce the same output for differently ordered inputs is inefficient.

### D.2. Set Representation

DeepSet (Zaheer et al., 2017) and Set Transformer (Lee et al., 2019) present a range of neural architectures that are Policy Invariant (PI) and are designed for learning set representations. In these architectures, each element $e_i$ is individually mapped to a latent space using a shared embedding layer $\phi(e_i)$. The latent representations obtained are then combined through a PI pooling layer. This layer is key to maintaining the PI characteristic of the entire function. For example, the function $f(E) = \Gamma(\sum_{i=1}^{k} \phi(e_i))$, where $\Gamma$ can be any function. This approach ensures that the output is invariant to the order of the input elements.

### D.3. Graph Neural Network

Graph Neural Networks (GNNs) (Wang et al., 2019; Battaglia et al., 2018; Jiang et al., 2020) offer a different perspective compared to set-based approaches. Instead of treating an agent's observation as a simple set, GNNs view these observations as having topological relationships among the features of different entities. Just like set-based methods, GNNs also employ shared embedding and pooling layers. These layers are adapted to learn functions specifically on graphs. These approaches allow for a more nuanced understanding of the relationships and interactions between different elements in the agent's observations.

### D.4. PE Functions

In the realm of deep learning, there has been research exploring the effectiveness of Policy Evaluation (PE) functions in problems involving graphs (Maron et al., 2019). However, within the field of multi-agent reinforcement learning (MARL), there are comparatively fewer studies that utilize the PE property. A notable exception is the Action Semantics Network (ASN) (Wang et al., 2020b), which investigates the varied impacts of different action types. Nevertheless, ASN doesn't explicitly focus on the PE property in its approach. This indicates a potential area for further exploration in MARL, leveraging the PE property to enhance understanding and performance.

## E. Background of Transformer

Transformer (Vaswani et al., 2017) is an attention-based model used in a wide range of deep learning frameworks and has demonstrated excellent performance (Meng et al., 2022; Dosovitskiy et al., 2021). A typical Transformer model consists of two sub-modules: an encoder and a decoder. The encoder generally maps the input space to a latent space, and the decoder employs the resultant latent embedding to generate a sequence of target outputs. The most crucial component in a Transformer is the attention layer, which can be formulated as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}\,, \tag{12}$$

where $\mathbf{Q}$ is the query embedding, $\mathbf{K}$ denotes the key embedding, and $\mathbf{V}$ represents the value embedding. The three embeddings can be learned via training. The scaling factor $d_k$ represents the dimension of the key embedding for normalizing the dot-product of the query and key embeddings.

## F. Hyperparameter Settings

Table A2 presents the hyperparameter settings used in our experiments, with each component optimized according to its respective reference literature. For both our proposed Transfermer and the baseline methods, each training curve illustrated in the main manuscript is based on five independent runs using different random seeds.

*Table A2.* Hyperparameter settings.

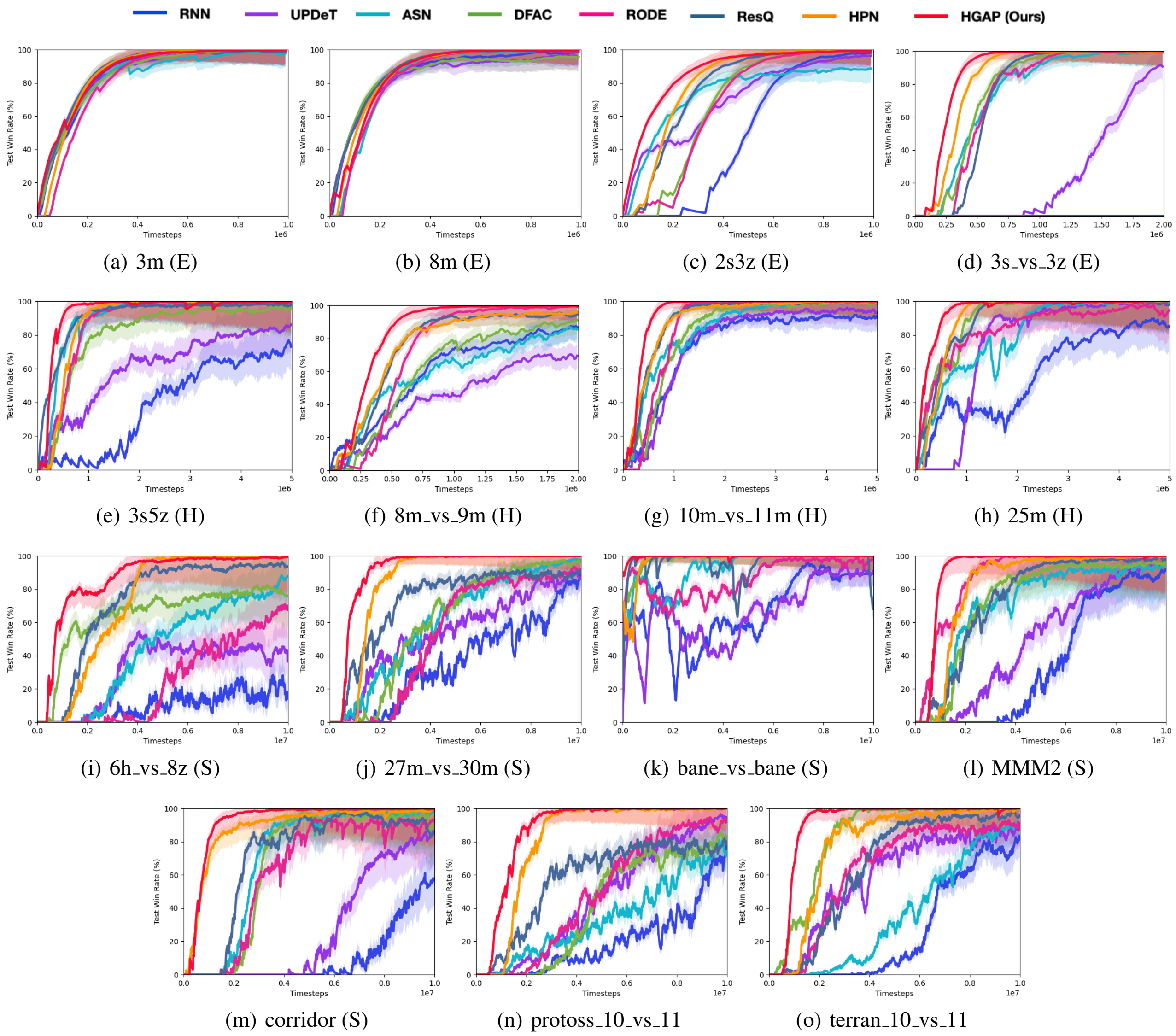
| Hyperparameters | Value |
|---|---|
| Batch size | 32 |
| Buffer size | 5,000 |
| Learning rate | 0.0005 |
| Gamma | 0.99 |
| Epsilon start | 1.0 |
| Epsilon end | 0.05 |
| Update target network interval | 10,000 |
| RNN hidden dimension | 64 |
| UPDeT embedding size | 32 |
| UPDeT attention heads | 3 |
| UPDeT attention layers | 2 |
| HPN embedding size | 64 |
| HGAP embedding size | 64 |
| HGAP attention layers | 1 |
| QMIX hypernet embedding | 64 |
| QMIX mixing embedding dimension | 32 |
| QTRAN opt loss | 1 |
| QTRAN nopt loss | 0.1 |
| QPLEX hypernet embedding | 64 |
| QPLEX mixing embedding dimension | 32 |
| QPLEX advantage hypernet embedding | 64 |
| QPLEX advantage hypernet layers | 3 |

## G. Additional Experimental Results

In this section, we provide supplemental evidence to substantiate the effectiveness of our methodology. In Section G.1, we present results from additional SMAC scenarios to demonstrate the superior performance of HGAP over the other agent network baselines discussed in the main manuscript. In Section G.2, we show that HGAP is capable of delivering better performance with a larger population compared with other approaches. Section G.3 and G.4 showcases additional experimental results that validate the adaptivity of HGAP with different MARL frameworks, including QPLEX and MAPPO respectively. Finally, Section G.5 provides experimental results to validate the transferability of HGAP in more cases.

## G.1. Additional Agent Networks Comparison

In this section, we present additional comparisons of the training curves in addition to those presented in Fig 5 of the main manuscript. These comparisons contrast the performance of HGAP with the other agent network baselines. In Fig. A1, we can notice that the HGAP agents rise faster and achieve higher win rates than the baseline methods. These results thus provide strong evidence for the effectiveness of the HGAP architecture in comparison to the other agent network designs. We also provide a qualitative evaluation results for these baselines in difference benchmarks in Table A3.



(a) 3m (E) (b) 8m (E) (c) 2s3z (E) (d) 3s_vs_3z (E)

(e) 3s5z (H) (f) 8m_vs_9m (H) (g) 10m_vs_11m (H) (h) 25m (H)

(i) 6h_vs_8z (S) (j) 27m_vs_30m (S) (k) bane_vs_bane (S) (l) MMM2 (S)

(m) corridor (S) (n) protoss_10_vs_11 (o) terran_10_vs_11

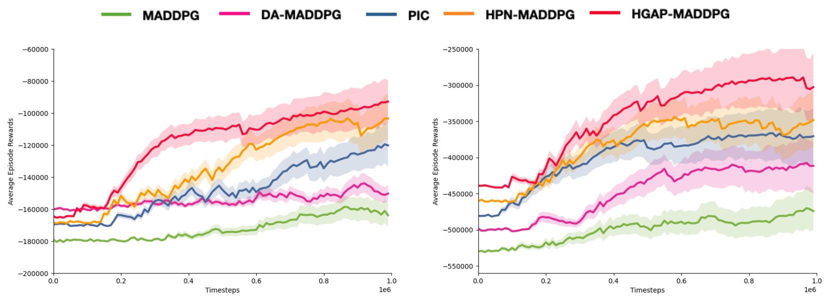
*Figure A1.* Test win rate comparison for different SOTA approaches in SMAC (Vinyals et al., 2017) and SMACv2 (Ellis et al., 2023). (E), (H), (S) stand for *Easy*, *Hard*, and *Super-Hard* scenarios.

*Table A3.* The averaged test win rates for all agent networks and mixing networks.

| **Environments** | | | **Vanilla** | | **Role-based** | **Quantile Fac** | | **PI/PE** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Map Name** | Easy | **RNN** | **UPDeT** | **RODE** | **DFAC** | **ResQ** | **ASN** | **HPN** | **HGAP** | **Timesteps** |
| SMAC | 3m | Easy | 96 | 98 | **100** | **100** | **100** | 99 | **100** | **100** | **1e6** |
| | 8m | Easy | 98 | 99 | 99 | **100** | **100** | **100** | **100** | **100** | **1e6** |
| | 3s_vs_3z | Easy | 99 | 98 | **100** | 99 | **100** | **100** | **100** | **100** | **2e6** |
| | 2s3z | Easy | 93 | 96 | 98 | 97 | 99 | 92 | **100** | **100** | **2e6** |
| | 1c3s5z | Easy | 95 | 97 | 99 | 98 | **100** | 94 | **100** | **100** | **2e6** |
| | 5m_vs_6m | Hard | 62 | 73 | 96 | 94 | 93 | 86 | 98 | **100** | **5e6** |
| | 8m_vs_9m | Hard | 85 | 77 | 96 | 93 | 98 | 88 | **100** | **100** | **2e6** |
| | 10m_vs_11m | Hard | 88 | 92 | 97 | 99 | **100** | 94 | **100** | **100** | **5e6** |
| | 25m | Hard | 79 | 83 | 92 | 91 | 98 | 88 | 99 | **100** | **5e6** |
| | 3s_vs_5z | Hard | 73 | 88 | 94 | 96 | 99 | 93 | **100** | **100** | **5e6** |
| | 3s5z | Hard | 75 | 81 | 91 | 93 | 96 | 88 | **100** | **100** | **5e6** |
| | MMM | Hard | 67 | 73 | 82 | 89 | 94 | 79 | 99 | **100** | **5e6** |
| | 27m_vs_30m | Super Hard | 53 | 69 | 72 | 83 | 89 | 77 | 96 | **100** | **1e7** |
| | 3s5z_vs_3s6z | Super Hard | 45 | 83 | 84 | 88 | 91 | 79 | 97 | **100** | **1e7** |
| | 6h_vs_8z | Super Hard | 27 | 43 | 73 | 84 | 94 | 59 | 98 | **100** | **1e7** |
| | bane_vs_bane | Super Hard | 77 | 82 | 89 | 92 | 95 | 86 | 99 | **100** | **1e7** |
| | corridor | Super Hard | 57 | 74 | 89 | 86 | 92 | 82 | 96 | **99** | **1e7** |
| | MMM2 | Super Hard | 81 | 84 | 96 | 93 | 95 | 88 | 97 | **100** | **1e7** |
| SMACv2 | protoss_5_vs_5 | - | 64 | 71 | 80 | 84 | 89 | 77 | 94 | **98** | **1e7** |
| | protoss_10_vs_11 | - | 53 | 62 | 71 | 80 | 86 | 73 | 91 | **96** | **1e7** |
| | protoss_20_vs_23 | - | 57 | 61 | 74 | 79 | 83 | 71 | 92 | **100** | **1e7** |
| | terran_5_vs_5 | - | 59 | 67 | 81 | 87 | 87 | 72 | 92 | **96** | **1e7** |
| | terran_10_vs_11 | - | 47 | 54 | 74 | 81 | 82 | 63 | 89 | **96** | **1e7** |
| | terran_20_vs_23 | - | 41 | 55 | 75 | 79 | 85 | 67 | 90 | **94** | **1e7** |
| | zerg_5_vs_5 | - | 53 | 63 | 79 | 83 | 88 | 71 | **94** | 92 | **1e7** |
| | zerg_10_vs_11 | - | 46 | 51 | 70 | 77 | 81 | 62 | 91 | **93** | **1e7** |
| | zerg_20_vs_23 | - | 49 | 57 | 69 | 79 | 83 | 63 | 89 | **95** | **1e7** |

### G.2. Large Population Environment

Previous MARL methodologies have often neglected to assess their ability to maintain consistent performance as the population size increases. In light of this, we have conducted an ablation study incorporating varying numbers of entities to ascertain that the HGAP surpasses other MARL approaches. Fig. A2 presents the comparative results across different entity counts. Leveraging the PI and PE properties, HGAP exhibits convergence to superior training performance in comparison to alternative methods. Notably, as the number of entities increases, the training performance gap between HGAP and other methods widens, providing additional empirical support for the significance of the PI and PE properties.

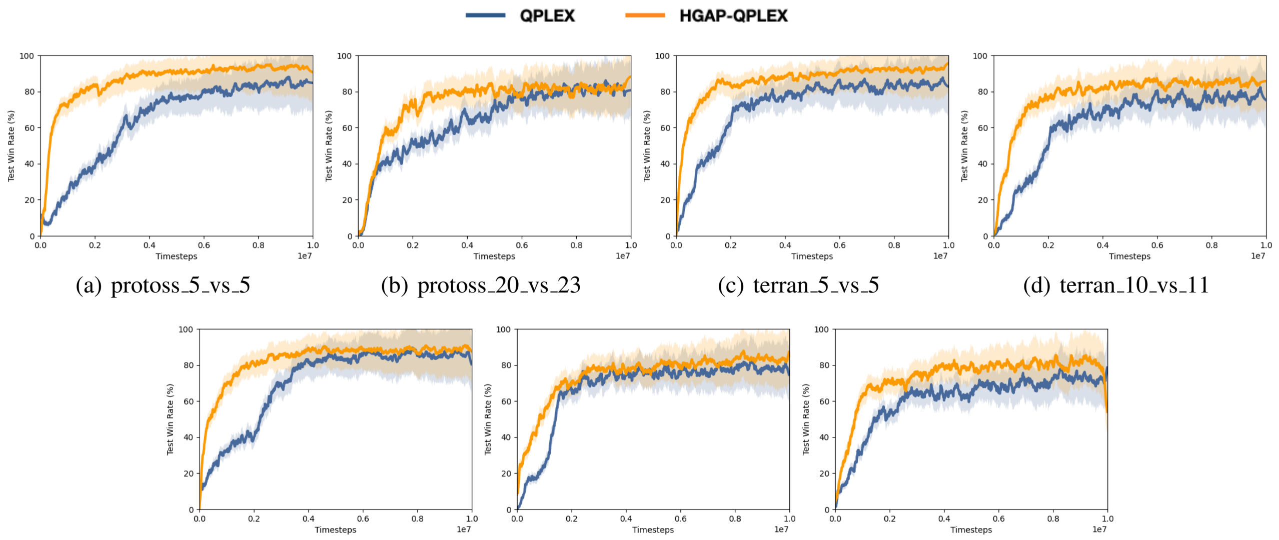


(a) Navigation (N=100) (b) Predator-Prey (N=200)

*Figure A2.* Test win rate comparison for MADDPG variants in Navigation with larger population.

### G.3. Additional Comparison of HGAP with QPLEX

To illustrate the applicability of the HGAP integrated with the value factorization method, additional experimental results are presented and elaborated upon in Section 4.3.2. The corresponding outcomes are graphically depicted in Fig. A3. The findings distinctly indicate that the HGAP agent, featuring PI and PE properties, effectively enhances training performance compared with a recurrent-based network.



(a) protoss_5_vs_5 (b) protoss_20_vs_23 (c) terran_5_vs_5 (d) terran_10_vs_11

(e) zerg_5_vs_5 (f) zerg_10_vs_11 (g) zerg_20_vs_23

*Figure A3.* Test win rate comparison for QPLEX and HGAP-QPLEX in SMACv2 (Ellis et al., 2023).

### G.4. Additional Comparison of HGAP with MAPPO

In order to assess the adaptability of the HGAP within policy-based MARL approaches, as expounded in Section 4.3.3, additional experimental results comparing the original MAPPO and HGAP-MAPPO are presented in Fig. A4. The outcomes consistently affirm that an actor network imbued with PI and PE properties effectively mitigates the occurrence of redundant learning scenarios wherein agents encounter identical information.

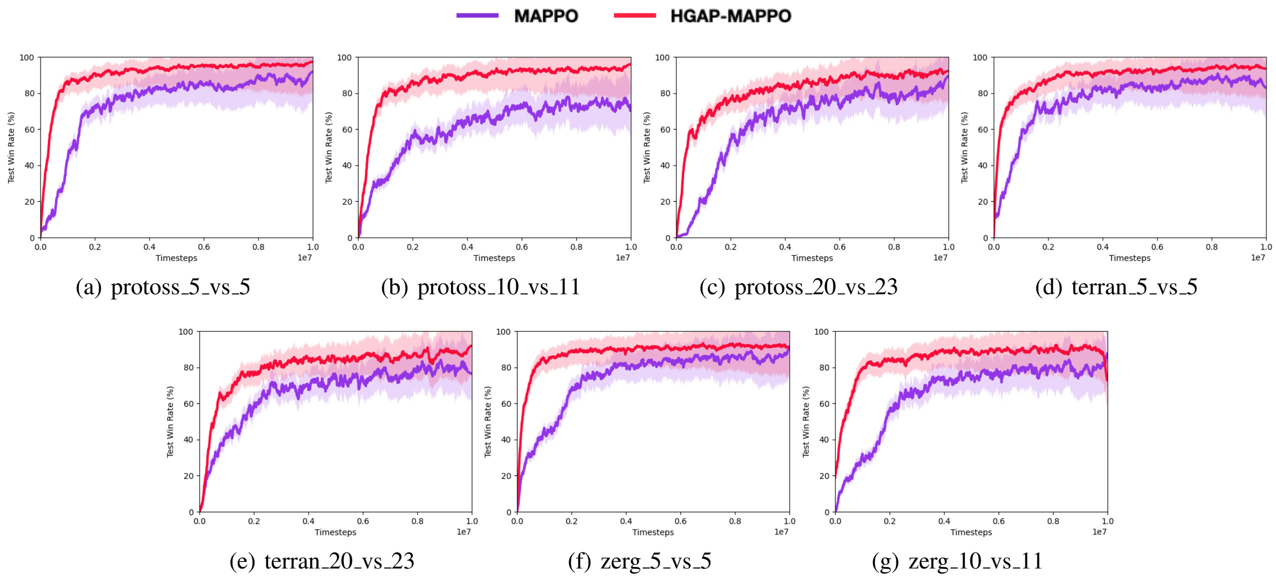


*Figure A4.* Test win rate comparison for MAPPO and HGAP-MAPPO in SMACv2 (Ellis et al., 2023).

## G.5. Additional Transferability Comparison between HGAP and HPN

As discussed in Section 4.5, we have explored the comparative transferability between HPN and HGAP. In order to support the argument that HGAP exhibits consistent adaptability across various transfer scenarios, additional experimental results for these settings are showcased in Fig. A5. The findings substantiate the notion that HGAP possesses superior transferability compared to HPN, highlighting the capacity of graph-based representations to retain and apply learned strategies in novel scenarios.

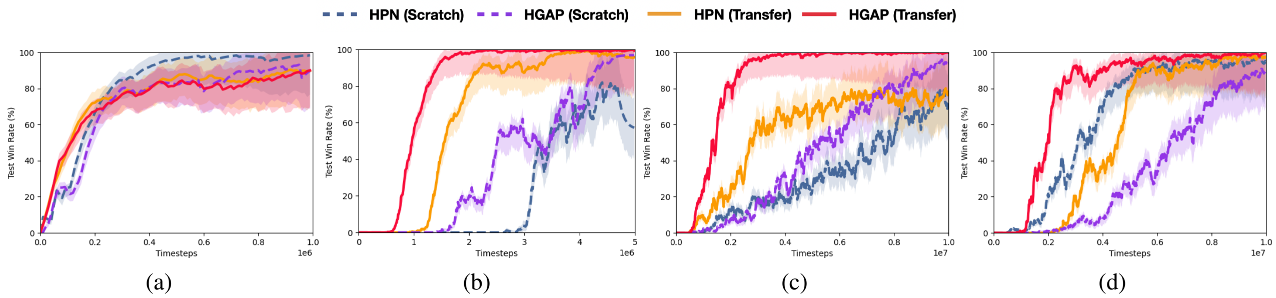
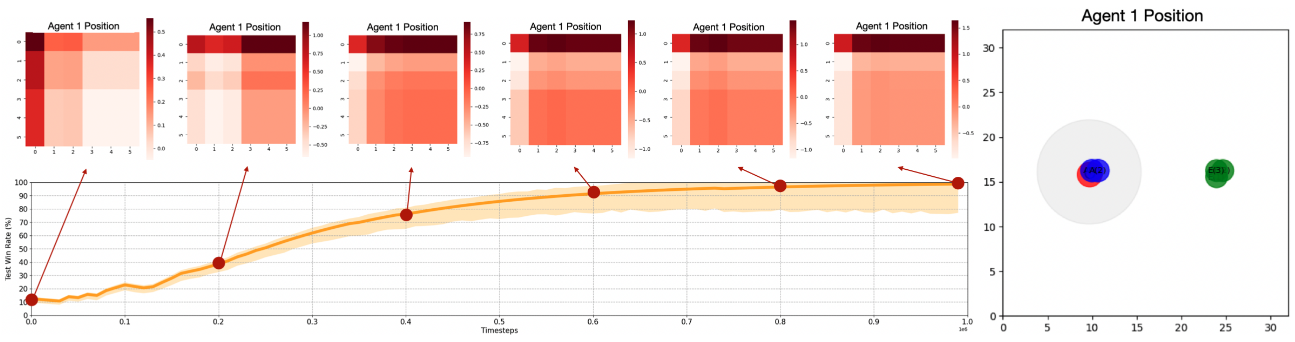


*Figure A5.* Transfer learning comparison between HPN and HGAP in SMAC (Vinyals et al., 2017). (a) 3m (E) $\rightarrow$ 8m (E). (b) 1c3s5z (E) $\rightarrow$ 3s_vs_5z (H). (c) 8m (E) $\rightarrow$ 27m (S). (d) 3s5z_vs_3s6z (S) $\rightarrow$ MMM2 (S).

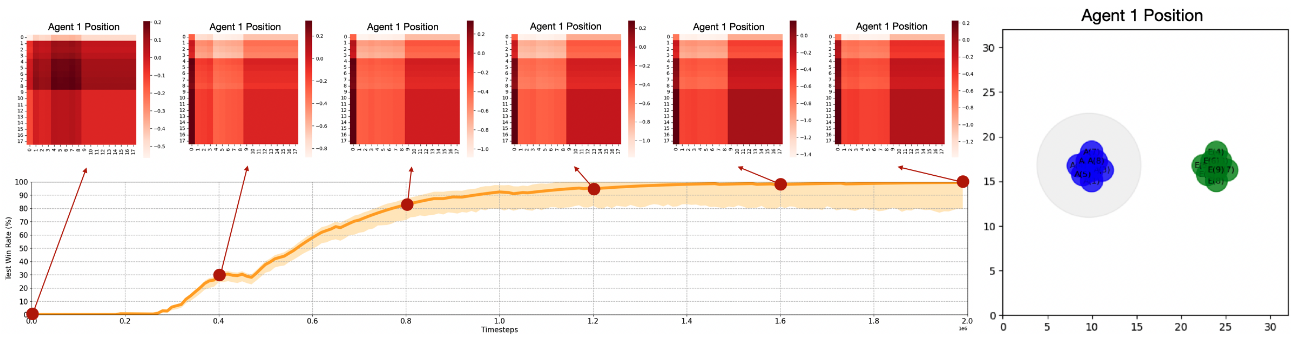
## H. Graph Attention Heatmap Variation

In this subsection, additional training heatmaps depicting the performance of HGAP across various SMAC maps are presented. The outcomes demonstrate the rapid acquisition of policy knowledge by HGAP within a brief training duration, resulting in a consistent and effective graph representation. Notably, the visual analysis reveals that agents exhibit a heightened focus on adversaries, directing increased attention to enemies within close regions.



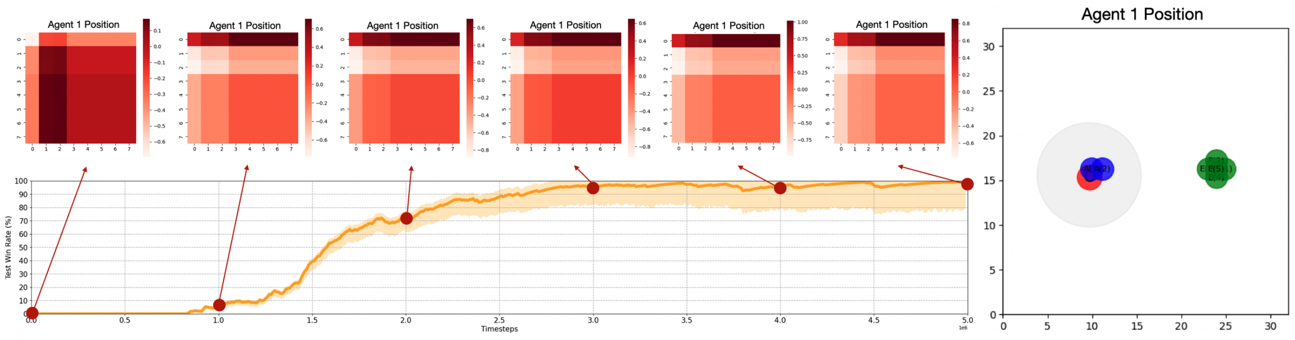
(a) (b)

*Figure A6.* (a) Attention heatmap variation during training in SMAC (Vinyals et al., 2017) 3m map at the initial state. (b) The initial position for all allies and enemies in 3m map.



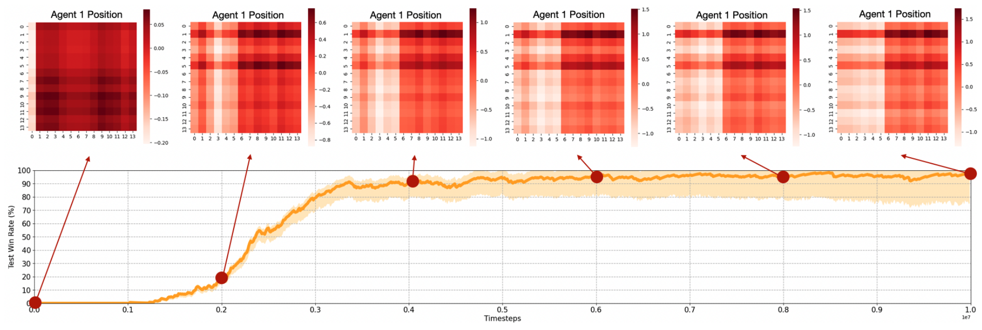
(a) (b)

*Figure A7.* (a) Attention heatmap variation during training in SMAC (Vinyals et al., 2017) 1c3s5z map at the initial state. (b) The initial position for all allies and enemies in 1c3s5z map.

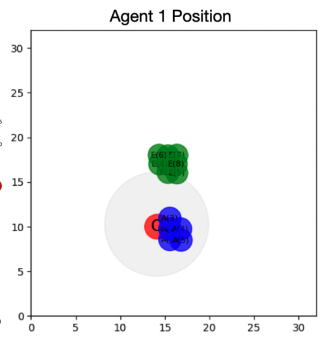


(a) (b)

*Figure A8.* (a) Attention heatmap variation during training in SMAC (Vinyals et al., 2017) 3s_vs_5z map at the initial state. (b) The initial position for all allies and enemies in 3s_vs_5z map.

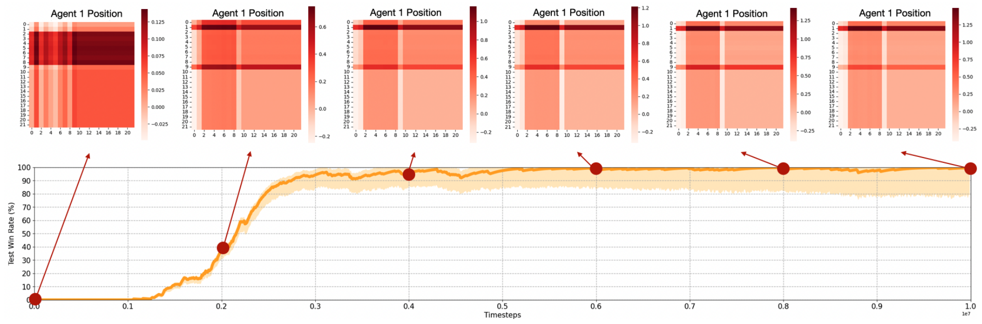


(a)

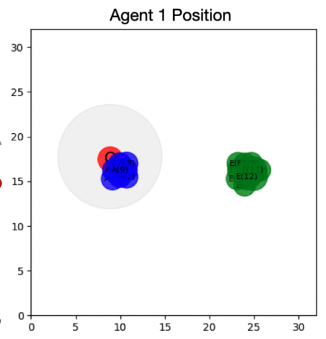


(b)

*Figure A9.* (a) Attention heatmap variation during training in SMAC (Vinyals et al., 2017) 6h_vs_8z map at the initial state. (b) The initial position for all allies and enemies in 6h_vs_8z map.



(a)



(b)

*Figure A10.* (a) Attention heatmap variation during training in SMAC (Vinyals et al., 2017) MMM2 map at the initial state. (b) The initial position for all allies and enemies in MMM2 map.