

Article

QMIX-GNN: A Graph Neural Network-Based Heterogeneous Multi-Agent Reinforcement Learning Model for Improved Collaboration and Decision-Making

Taiyin Zhao * , Tian Chen and Bing Zhang

Laboratory of Intelligent Collaborative Computing, School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China; 202421090329@std.uestc.edu.cn (T.C.); zhangbingcode@163.com (B.Z.)

* Correspondence: tyzhao@uestc.edu.cn

Abstract: In multi-agent reinforcement learning, the fully centralized approach suffers from issues such as explosion of the joint state and action spaces, leading to performance degradation. On the other hand, the fully decentralized approach relies on agents that focus solely on maximizing their own rewards, making effective collaboration difficult and complicating adaption to scenarios that require cooperation among multiple agents. The Centralized Training and Decentralized Execution (CTDE) framework combines both fully centralized and fully decentralized approaches. During the training phase, a virtual central node receives the observations and actions of all agents for training, while during the execution phase each agent makes decisions based only on its own observations. However, in this framework the agents do not fully consider the information of other agents or the complex interactions between them during execution, which affects the correctness of their decisions. Therefore, this paper proposes a heterogeneous multi-agent reinforcement learning model based on graph neural networks, which we call QMIX-GNN. This model efficiently and flexibly handles input data of different dimensions, enabling the fusion of heterogeneous multi-agent information and providing this fused information to the agents. In turn, this allows them to perceive more comprehensive information and improve the correctness of their decisions. Experimental results demonstrate that the QMIX-GNN model performs better than other methods on complex multi-agent collaborative tasks.

Keywords: deep learning; reinforcement learning; multi-agent reinforcement learning; graph neural network; heterogeneity



Academic Editors: João M. F. Rodrigues and Pedro Couto

Received: 16 February 2025

Revised: 25 March 2025

Accepted: 28 March 2025

Published: 30 March 2025

Citation: Zhao, T.; Chen, T.; Zhang, B. QMIX-GNN: A Graph Neural Network-Based Heterogeneous Multi-Agent Reinforcement Learning Model for Improved Collaboration and Decision-Making. *Appl. Sci.* **2025**, *15*, 3794. <https://doi.org/10.3390/app15073794>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Multi-Agent Reinforcement Learning (MARL) has emerged as a powerful framework for developing autonomous agents that can interact, collaborate, and compete in dynamic environments. As a subset of reinforcement learning (RL), MARL involves multiple agents learning simultaneously, sharing information, and making decisions based on their local observations and interactions with others [1–5]. The significance of MARL lies in its ability to model and solve complex real-world problems that require cooperation, coordination, and competition among agents across various domains, including information security [6,7], resource management [8–10], building information modeling [11], and autonomous systems. In these fields, agents must adapt to continuously changing conditions, which makes MARL a promising approach for developing intelligent systems capable of navigating uncertain and dynamic environments.

One of the key challenges in MARL is balancing the tradeoff between centralized and decentralized approaches [12]. In a fully centralized approach, agents share global information, which can alleviate coordination problems but may result in the explosion of joint state and action spaces, leading to inefficiency and scalability issues as the environment's complexity increases. On the other hand, in fully decentralized approaches, agents make independent decisions based solely on their local observations, focusing on maximizing their individual rewards. While this simplifies decision-making, it can hinder collaboration and coordination, particularly in environments where joint actions are essential for optimal outcomes. The Centralized Training and Decentralized Execution (CTDE) framework seeks to address this tradeoff by allowing agents to be trained with global information while making independent decisions during execution. However, this approach still faces the limitation of agents not having access to comprehensive information about other agents during execution, which limits their ability to make fully informed decisions and diminishes overall system performance.

Graph Neural Networks (GNNs) have emerged as a powerful tool for capturing the complex relationships and dependencies between agents in multi-agent systems [13,14]. Recent advances in GNN-based MARL have demonstrated the potential of graph structures to model agent interactions. For instance, ref. [15] proposed GNNComm-MARL by utilizing the graph attention network (GAT) [16] to enhance observability, mitigate non-stationarity, and improve scalability in multi-agent systems, while ref. [17] introduced MARGIN by extending the idea of mutual information optimization from the graph domain to multi-agent systems. In addition, ref. [18] leveraged GNNs to process both local and global information, further enhancing model performance.

Building upon these developments, we propose QMIX-GNN, a heterogeneous MARL model that integrates GNNs to improve collaboration and coordination among agents. The key novelties of our approach compared to existing methods are as follows:

- **Adaptive Information Fusion:** QMIX-GNN introduces a GNN-based information infusion mechanism that aggregates data from multiple agents, allowing for enhanced team perception and decision-making.
- **Heterogeneous Information Processing:** Our model incorporates a dedicated module that handles the diverse observation and action spaces of heterogeneous agents, facilitating seamless communication and collaboration.
- **Scalability on Complex Tasks:** By integrating these modules within the CTDE framework, QMIX-GNN achieves significant improvements in both performance and convergence speed on complex multi-agent coordination tasks.

The structure of this paper is as follows: Section 1 provides an introduction to the problem domain, outlining the motivation and background for our work; Section 2 details necessary information, including the existing limitations, proposed approach, and a comprehensive overview of the data and data processing techniques used in this study; Section 3 describes the experimental setup, including both hardware and software configurations, presents the experimental results, and includes an ablation study to evaluate the contributions of each module; Section 4 offers a discussion of the methodology, exploring the implications and potential improvements of our approach; finally, Section 5 concludes the paper with a summary of our findings.

2. Materials and Methods

2.1. Preliminaries

2.1.1. Reinforcement Learning

RL is a feedback-based learning paradigm in which agents interact with an environment in order to maximize their cumulative long-term rewards through trial and error. Unlike supervised learning, RL does not rely on predefined datasets, instead allowing agents to learn optimal decision-making policies through continuous interaction. An RL problem can be modeled using a Markov Decision Process (MDP) $G = \langle S, A, P, R \rangle$, where S represents the state space, A represents the action space, P denotes the state transition function, and R is the reward function [19]. At each time step, the agent observes a state $s_t \in S$ and selects an action $a_t \in A(s_t)$. This results in a new state s_{t+1} and reward r_{t+1} , guiding the agent's policy π to optimize the expected cumulative reward.

Figure 1 illustrates this through a schematic diagram of the interaction between the agent and the environment. Based on the observed state s_t , the agent selects an action a_t to execute, where $a_t \in A(s_t)$ and $A(s_t)$ represents the set of actions available to the agent in state s_t . After executing the action, the agent receives a numerical reward r_{t+1} and observes a new environmental state s_{t+1} after one time step. Figure 1 illustrates the interaction between the agent and the environment, where the agent makes decisions based on its observations and the environment provides feedback in the form of rewards and state transitions. The agent's objective is to maximize its long-term return by learning a policy that selects the best actions over time.

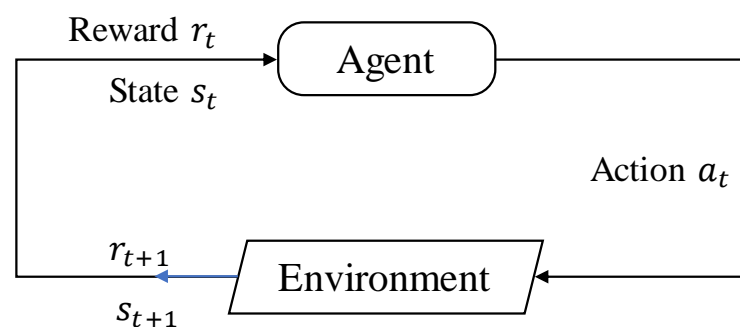


Figure 1. Interaction between the agent and the environment.

Q-learning is a foundational reinforcement learning algorithm that seeks to learn the optimal action–value function, commonly denoted as the Q-function, by iteratively updating Q-values based on the rewards received from the environment. Deep Q-Networks (DQNs) build upon this framework by combining Q-learning with deep neural networks to address the instability and high-dimensionality problems of traditional Q-learning. In a DQN, the Q-function is approximated by a deep neural network and the Q-values are updated using the Bellman equation. A DQN consists of two networks: the current network and a target network, which usually has the same architecture as the current network but is updated less frequently. This approach stabilizes the training process and reduces the variance of the target values.

2.1.2. Multi-Agent Reinforcement Learning

MARL is a branch of RL in which each agent selects actions based on its own observations of the environment and receives feedback in the form of rewards or penalties. The actions of all agents influence one another and collectively determine the overall system performance.

In MARL, agents operate in a Decentralized Partially-Observable Markov Decision Process (Dec-POMDP) [20] $\langle I, S, \{A_i\}, P, \{O_i\}, R, \gamma \rangle$ with partial observations $o_i \sim O_i(s)$. The CTDE paradigm [21,22] addresses the challenges of non-stationarity and partial observations:

- Centralized Training: Joint policies are learned using the global information (e.g., all agents' observations).
- Decentralized Execution: Agents act independently using local observations.

Key CTDE-based algorithms include:

- COMA [23]: Counterfactual baselines are used for credit assignment by marginalizing individual agents' actions while keeping others fixed.
- QMIX [22]: The joint Q-value Q_{tot} is decomposed by a monotonic mixing network; this ensures that $\frac{\partial Q_{tot}}{\partial Q_i} \geq 0$, enabling decentralized argmax operations.
- QTRAN [24]: QMIX's monotonicity constraint is relaxed through a transformed joint Q-value

$$Q_{tot} = \sum_i Q_i + V(s) + \max_{\mathbf{u}'} \left(\sum_i Q'_i - Q'_{tot} \right), \quad (1)$$

where $V(s)$ is a state-dependent baseline.

2.1.3. Graph Neural Networks

GNNs process relational data by passing messages between nodes. For multi-agent systems modeled as graphs $G = (V, E)$, with agents as nodes V and interactions as edges E :

- Graph Convolutional Networks (GCNs) [25]: Aggregate neighbor features Z with spectral convolution, as follows:

$$Z^{(l+1)} = \sigma \left(\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} Z^{(l)} W^{(l)} \right), \quad (2)$$

where $\hat{A} = A + I$ is the adjacency matrix with self-loops, \hat{D} is the degree matrix, σ is the activation function, and $W^{(l)}$ are learnable weights.

- GAT [16]: Computes the adaptive attention weights α_{ij} between nodes i and j :

$$\alpha_{ij} = \text{softmax}_j \left(\text{LeakyReLU} \left(\mathbf{a}^T [\mathbf{W}h_i | \mathbf{W}h_j] \right) \right), \quad (3)$$

where \mathbf{a} and \mathbf{W} are learnable parameters. GAT uses attention-based edge updates to handle dynamic graphs, which are common in MARL.

2.2. Existing Issues

While CTDE has proven effective in homogeneous multi-agent systems [22], its application to heterogeneous scenarios faces critical limitations:

- Partial Observability Constraints: In decentralized execution, agents lack access to global state information. Existing CTDE methods (e.g., QMIX, COMA) assume that agents can implicitly coordinate through shared value functions; however, this fails when the agents exhibit heterogeneous observation spaces or roles.
- Static Interaction Modeling: Traditional value factorization approaches [24] decompose joint Q-values under the assumption of fixed monotonicity, ignoring dynamic agent relationships.
- Heterogeneous Information Fusion: Agents with diverse sensor modalities (e.g., LiDAR vs. cameras) generate multimodal data with varying structures and scales. Simple concatenation or averaging of observations can lose critical relational patterns [26], while manually designed fusion rules lack adaptability.

To address these challenges, we model heterogeneous multi-agent systems as graphs. GCNs [25] apply fixed spectral filters, assuming homogeneous nodes and static topology. This is inadequate for our dynamic heterogeneous graph setting. In contrast, GAT [16] computes adaptive edge weights via attention mechanisms. These insights motivate our QMIX-GNN framework, which integrates GAT-based relational reasoning with monotonic value factorization, as detailed in Section 2.3.

2.3. Proposed Method

We propose a GNN-based heterogeneous MARL model called QMIX-GNN, the structure of which is shown in Figure 2. The proposed model consists of three parts: an information fusion network, an agent network, and a mixing network. The information fusion network is built based on GNNs, with the purpose of integrating the observation information from multiple agents into team-level information, which is then provided to the agents for their use. The agent network, hereafter referred to as the Recurrent Neural Network (RNN), is composed of Gated Recurrent Units (GRUs). The RNN is designed to capture the agents' historical observation information and team information, then generate the agents' action value functions. The mixing network is composed of a hypernetwork, which generates the global action value function based on the team information, and the agent's value function, while also ensuring that the global action value function maintains monotonicity with respect to the agent's action value function. The design details of these three components are introduced in the following subsections.

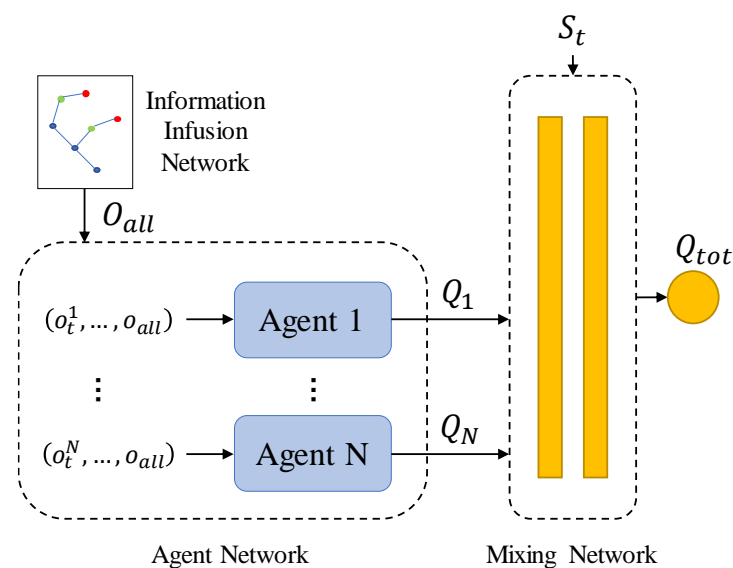


Figure 2. Structure of QMIX-GNN.

2.3.1. Agent Network

The agent network is designed to compute each agent's local Q-value. It consists of three parts in total, as shown in Figure 3:

- A projection matrix is used to map the states of heterogeneous agents into a consistent state space.
- Shared-bottom parameters accelerate the model's learning speed and improve its generalization ability.
- A tower structure enables different types of agents to learn differentiated policies, helping them to adapt to various action spaces.

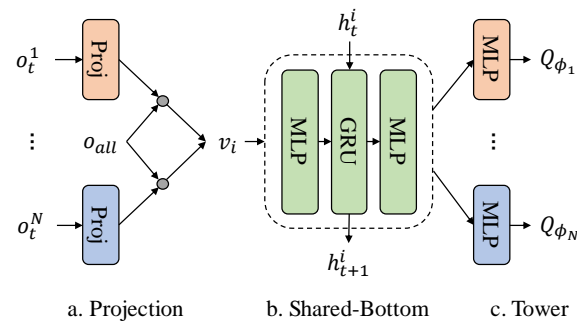


Figure 3. Structure of the agent network.

In MARL, the feature spaces and actions of heterogeneous agents may be inconsistent, necessitating the processing of this heterogeneous information. For cases where the feature spaces are inconsistent, a transformation matrix is constructed for each type of agent to map the agents' heterogeneous features into a common feature space. The projection process is shown in Equation (4):

$$o'_i = \mathbf{M}_{\phi_i} \cdot o_i, \quad (4)$$

where ϕ_i represents the type of agent i , \mathbf{M} is the projection matrix, o_i is the agent's raw observation, and o'_i is the projected feature of the agent derived from its raw observation. With a projection matrix specific to each type, the features of any agent can be processed.

To address the issue of inconsistent action spaces among heterogeneous agents in MARL, we adopt the tower structure from multitask learning. This structure employs a shared bottom in order to share the lower-level parameters among multiple task models. At the same time, it uses independent towers to produce different outputs for the specific differences of each task [3]. This approach can significantly reduce the number of parameters and the computational complexity, improving the training speed and generalization ability of the model. To account for the differences among individual agents, independent towers can be utilized to generate distinct outputs to accommodate various action spaces, effectively modeling the agents' heterogeneity and enhancing the decision-making performance of the multi-agent system.

The shared bottom layer takes the aligned information projected from the agent along with team information o_{all} and outputs the aligned vector b_i , as follows:

$$b_i = \text{bott}(\text{concat}([o'_i, o_{all}])), \quad (5)$$

where bott represents the shared bottom layer common to all agents and concat denotes the concatenation operation. The method for obtaining o_{all} will be detailed later. In this paper, we employ an RNN to implement the shared bottom network with a tower structure designed for each agent to address the heterogeneous policy differences among agents and output distinct action spaces. Consequently, the individual Q-value of an agent can be expressed as

$$Q_i = \text{tow}_{\phi_i}(b_i), \quad (6)$$

where tow refers to the tower structure designed for agent i (implemented in this work as a Multi-Layer Perceptron (MLP) network) and Q_i denotes the Q-value of agent i . In conclusion, the overall process of handling the heterogeneous feature and action spaces across agents can be formulated as

$$Q_i = \text{tow}_{\phi_i}(\text{bott}(\text{concat}([\mathbf{M}_{\phi_i} \cdot o_i, o_{all}]))). \quad (7)$$

2.3.2. Information Infusion Network

As shown in Figure 4, the information infusion network consists of two modules: a graph construction module (GCM) and an information infusion module (IFM). The GCM builds a graph to model the agent relationships based on the agents' states, while the IFM processes the graph data with a GAT, integrates feature information from heterogeneous agents, and generates a unified team-level information vector accessible to each agent.

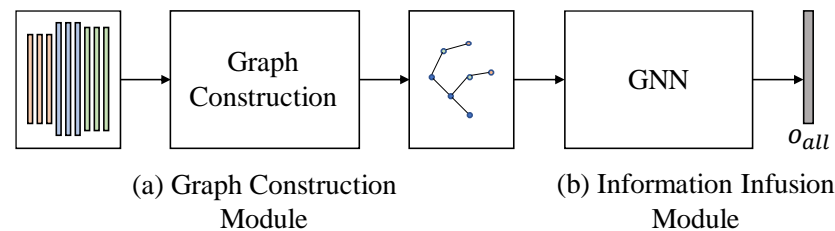


Figure 4. Structure of the information infusion network.

The experiments in this paper were conducted in the Starcraft Multi-Agent Challenge (SMAC) environment [27], where agents have fixed visual and attack radii. Agents connect to their k closest neighbors within their visual/attack range and form a k -nearest neighbor graph based on spatial proximity, as in Figure 5. Specifically, for each agent, we calculate pairwise Euclidean distances to visible agents, select the k nearest agents as neighbors, and establish bidirectional edges between neighbors.

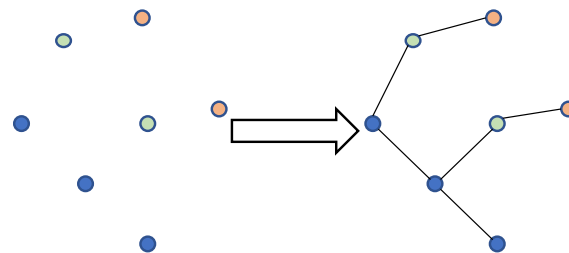


Figure 5. Establishing edges between agents and their neighboring agents to form a k -nearest neighbors graph.

After generating the graph, the GAT can be used to integrate the information of the agents. The multi-agent information fusion process is shown in Figure 6. Each agent first fuses the information from its k -neighbors, then the information from the agents is enriched with neighbor data and further fused to form the team information.

First, the similarity coefficient e between each agent and its k nearest neighbors is computed as follows:

$$e_{ij}^h = a_h^T \cdot [o'_i || o'_j], \quad (8)$$

where e_{ij}^h is the attention coefficient between nodes i and j for head h , a_h^T represents the learnable weight vector of attention head h , $||$ is the concatenation operator, and $[o'_i || o'_j]$ denotes the concatenation of the transformed feature matrices of nodes i and j .

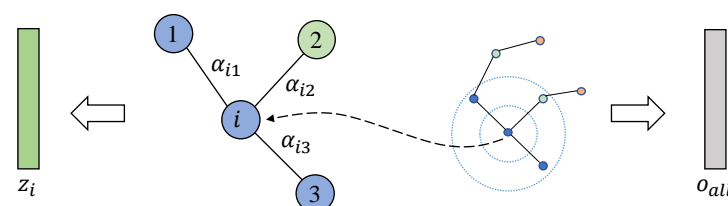


Figure 6. Information infusion of multiple agents.

The node features of the graph are the projected features o' . The edge relationships are stored in the adjacency matrix A , $A \in \mathbb{R}^{n \times n}$, and \mathcal{N}_i denotes the set of k -nearest neighbors for agent i . Then, the similarity coefficients are normalized via the softmax operation to obtain the attention coefficients α_{ij} between neighboring agents, ensuring that the sum of attention coefficients for each node equals 1. The computation is formulated as follows:

$$\alpha_{ij}^h = \text{softmax}_j(e_{ij}^h) = \frac{\exp(e_{ij}^h)}{\sum_{z \in \mathcal{N}_i} \exp(e_{iz}^h)}. \quad (9)$$

After obtaining the attention coefficients between agents, the features of each agent's k -nearest neighbors are aggregated through a weighted summation using these coefficients, resulting in the feature v that integrates neighborhood information. The computation is as follows:

$$v_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij}^h \cdot \mathbf{W} o'_j. \quad (10)$$

The fused features of the agent across multiple attention heads are obtained by concatenating the outputs of all heads, as shown in Equation (11):

$$z_i = \parallel_{h=1}^H v_i^h, \quad (11)$$

where H is the total number of attention heads. Then, a pooling operation is applied to the aggregated neighborhood features to derive the team-level feature vector o_{all} , which integrates the information from all agents. The computation is formalized as follows:

$$o_{all} = \frac{1}{N} \sum_{i=1}^N z_i. \quad (12)$$

In summary, the team-level feature o_{all} which integrates information from all agents can be described as follows:

$$o_{all} = \frac{1}{N} \sum_{i=1}^N \left(\parallel_{h=1}^H \left(\sum_{j \in \mathcal{N}_i} \frac{\exp(e_{ij}^h)}{\sum_{z \in \mathcal{N}_i} \exp(e_{iz}^h)} o'_j \right) \right). \quad (13)$$

2.3.3. Mixing Network

The structure of the mixing network aligns with QMIX [22], which monotonically aggregates the individual Q-values of agents into a joint team Q-value to enable multi-agent collaboration and coordination:

$$Q_{tot} = \text{Mix}(Q_1(o_1, o_{all}), Q_2(o_2, o_{all}), \dots, Q_N(o_N, o_{all}); s_t). \quad (14)$$

As illustrated in Figure 7, the mixing network comprises two subnetworks: the inference network (IN) and the parameter generation network (PGN). The IN is a feedforward neural network that takes the Q-values output by individual agent network as input and computes the team Q-value through weight matrices and bias vectors. The PGN is a hypernetwork that uses absolute global information s_t provided during training or observation o_{all} as its substitute to generate the weight matrices, which are constrained to be non-negative in order to satisfy monotonicity requirements, as well as the bias vectors for the inference network. It consists of multiple Fully Connected (FC) layers, with the final layer employing an absolute-valued activation function to enforce non-negative weights.

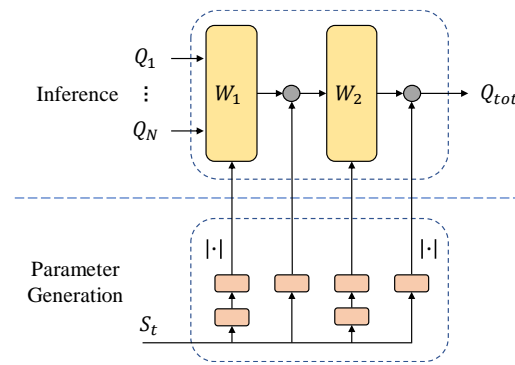


Figure 7. Structure of the mixing network.

For clarity of presentation, the important variables are summarized in Table 1.

Table 1. Important variables and their meanings.

Variable	Meaning
o	The agent's raw observation
ϕ	The agent's type
o'	The projected feature
b	The aligned vector output of Shared-Bottom
\mathbf{M}	The projection matrix
o_{all}	The team information
h	Attention head
H	The number of attention heads
N	The number of agents
\mathcal{N}	The number of k -nearest neighbors
a	Agents' actions
u	The joint action of agents
s	Agents' states
r	Agents' rewards
Q	Agents' Q-value
θ	Models' parameters
v	The feature that integrates neighbor information
z	The concatenated features

2.3.4. Implementations

QMIX-GNN employs an end-to-end training approach, with its overall training architecture aligning with DQN. The selected loss function is as follows:

$$\mathcal{L}(\theta) = \sum_{i=1}^b \left[(y_i^{tot} - Q_{tot}(\mathbf{o}, \mathbf{u}, s; \theta))^2 \right] \quad (15)$$

where b denotes the batch size sampled from the replay buffer, \mathbf{o} represents the joint observation of agents, \mathbf{u} is the joint action of the agents, s represents the absolute global state, θ represents the parameters of the estimated network, and y^{tot} is the output of the target network:

$$y^{tot} = r + \gamma \max_{u'} Q_{tot}(\mathbf{o}', \mathbf{u}', s'; \theta^-) \quad (16)$$

where θ^- stands for the parameters of the target network. The pseudocode of our QMIX-GNN is provided in Algorithm 1.

Algorithm 1 QMIX-GNN**Input:** Hyperparameters.**Output:** Model parameters θ .

```

1:  $step = 0, \theta^- = \theta$ . ▷ Initialize
2: while  $step < step_{max}$  do
3:    $t = 0$  ▷ Initialize episode
4:    $s_0 \leftarrow initial\ state$  ▷ Obtain the initial state
5:   while  $s_t \neq terminal$  and  $t < episode\ limit$  do
6:      $\mathbf{o}_t \leftarrow$  observations for all agents from the environment
7:      $A_t \leftarrow K\text{-Nearest Neighbor Graph}$  ▷ Construct the graph for current timestep
8:      $\mathbf{o}_{all}^t \leftarrow GAT(\mathbf{o}_t, A_t)$  ▷ Infuse observation information from all agents
9:     for each agent  $i$  do
10:       $\epsilon = \text{epsilon-schedule}(step)$  ▷ Explore or exploit threshold
11:      if  $\text{random}(0, 1) < \epsilon$  then ▷ Explore or exploit
12:         $Q_i \leftarrow RNN(o_t^i, a_t^i, o_{all}^i)$ 
13:         $a_t^i = \arg \max_{a^j} Q_i(a^j)$ 
14:      else
15:         $a_t^i = \text{randint}(1, |\cup \phi_i|)$  ▷ Random exploration
16:      end if
17:    end for
18:     $r_t, s_{t+1} \leftarrow$  get reward and next state from the environment
19:     $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{o}_t, s_t, \mathbf{u}_t, r_t, s_{t+1}, A_t)\}$  ▷ Save data to replay buffer
20:     $t = t + 1, step = step + 1$ 
21:    if  $|\mathcal{D}| > \text{batch size}$  then
22:      batch  $\mathbf{b} \leftarrow$  random batch of episodes from  $\mathcal{D}$ 
23:      for each timestep  $t$  in each episode in batch  $\mathbf{b}$  do
24:         $\mathbf{o}_{all}^t \leftarrow GAT(\mathbf{o}_t, A_t)$ 
25:         $Q_{tot} = \text{Mix}(Q_1(o_t^1, u_t^1, o_{all}^t), \dots, Q_N(o_t^N, u_t^N, o_{all}^t); s_t; \theta)$ 
26:      end for
27:       $\Delta Q_{tot} = y^{tot} - Q_{tot}$ 
28:       $\Delta \theta = \nabla (\Delta Q_{tot})^2$ 
29:       $\theta = \theta - \alpha \Delta \theta$  ▷ Update model parameters
30:    end if
31:    if update-interval steps have passed then
32:       $\theta^- = \theta$ 
33:    end if
34:  end while
35: end while

```

3. Results

3.1. Experimental Platform

We conducted experiments on the SMAC to validate the effectiveness of our method. The SMAC is an open-source experimental platform for studying collaborative MARL based on Blizzard's real-time strategy game StarCraft II. In this game, players need to collect resources, build structures, and create units to defeat their opponents by conquering their territory and destroying their base. The game has two modes: macro mode and micro mode. Macro mode refers to high-level strategic tasks, such as resource collection, factory construction, and organizing army attacks on the enemy. Micro mode refers to more detailed control of individual units in small-scale skirmishes with the enemy. The SMAC platform currently focuses only on micro mode.

In micro mode, each agent is controlled individually, and can only access information such as the relative position, health, and shields of other agents within its line of sight. Alliances consist of multiple agents that battle against multiple enemy agents, requiring training of agents to fight against the built-in AI. The SMAC adopts a CTDE approach,

meaning that the environment provides global observations during training, but such information is not available to agents during execution.

The SMAC offers several scenarios, including homogeneous and heterogeneous ones. Homogeneous scenarios involve both sides having only one type of agent. For example, in the 3m (Marine) scenario, both allies and enemies have only Marines as agents. On the other hand, heterogeneous scenarios feature multiple types of agents with different skills and attributes. For instance, in the 2s3z (Stalker+Zealots) scenario, both sides consist of two Stalkers and three Zealots.

In this experiment, we used a computer with an Intel(R) Xeon(R) Gold 6330 CPU @ 2.00 GHz processor and an RTX 3090 GPU, running on the Ubuntu operating system. The experiments were conducted using the Python programming language and the PyTorch deep learning framework. The specific versions of the hardware and software used in the experiment are shown in Table 2.

Table 2. The hardware and software used in our experiments.

Name	Version
CPU	Intel(R) Xeon(R) Gold 6330 CPU @ 2.00 GHz
GPU	Nvidia GeForce RTX 3090
OS	Ubuntu 18.04.5 LTS
Language	Python 3.8.10
DL Framework	PyTorch 1.7.0

3.2. Experimental Configure

To comprehensively validate the performance of QMIX-GNN, experiments were conducted on six scenarios in the SMAC platform: 3m, 2s3z, 1c3s5z, 10m_vs_11m, MMM, and MMM2, which include both homogeneous and heterogeneous scenarios. The baseline models included COMA, QMIX, and QTRAN.

The specific information for the four scenarios used in the experiments is shown in Table 3:

Table 3. Specific information for the SMAC scenarios.

Scenario	Alliance	Enemy	Type	Difficulty
3m	Marine \times 3	Marine \times 3	Homogeneous	Easy
2s3z	Stalker \times 2 Zealot \times 3	Stalker \times 2 Zealot \times 3	Heterogeneous	Easy
1c3s5z	Colossus \times 1 Stalker \times 3 Zealot \times 5	Colossus \times 1 Stalker \times 3 Zealot \times 5	Heterogeneous	Medium
10m_vs_11m	Marine \times 10	Marine \times 11	Homogeneous	Medium
mmm	Medivac \times 1 Marauder \times 2 Marine \times 7	Medivac \times 1 Marauder \times 2 Marine \times 7	Heterogeneous	Hard
mmm2	Medivac \times 1 Marauder \times 2 Marine \times 7	Medivac \times 1 Marauder \times 3 Marine \times 8	Heterogeneous	Hard

The experimental scenario settings were as follows: the SMAC game difficulty was set to 7, with an action taken every eight steps, for a total of one million training steps. The discount factor was set to $\gamma = 0.99$ and the RMS optimizer is used. Exploration was carried out using the ϵ -greedy algorithm, where the exploration rate decay was from 1.0 to 0.05 over the first 50,000 steps and remained at 0.05 thereafter. The number of attention heads was set to 6, and the experience replay buffer size was set to 3000. The number of nearest neighbors was set to 5. The parameters of the evaluation network (evaluate net) were assigned to the target network (target net) at every 200 training steps. The reward mechanism used the default shaped reward provided by SMAC. The experimental parameters are shown in Table 4.

In terms of hyperparameter selection, while a comprehensive grid search would theoretically be ideal to determine the best settings, its computational cost was prohibitive given the scale of the experiments. Instead, we adopted a sequential tuning approach in which each hyperparameter was tuned individually while keeping the others fixed. This strategy allowed us to identify effective settings that balanced performance with computational efficiency.

Table 4. The experimental parameters.

Parameter	Value
SMAC Difficulty	7 (Hard)
Episode	1,000,000
Discount factor γ	0.99
Parameter update step	200
Attention head	6
K-nearest neighbor	5
Replay buffer size	3000

3.3. Experimental Results

Table 5 illustrates the average winning rate during the last final 100k training steps for all models across the four scenarios:

- In the 3m scenario, QTRAN performs the best, followed by QMIX-GNN and QMIX, which show similar results. COMA performs slightly worse, with a winning rate of only around 76%, indicating relatively poor performance.
- In the 2s3z scenario, our QMIX-GNN has a slight decrease in winning rate compared to QMIX (a 2.6% drop). The reason for this could be that the 2s3z scenario is relatively simple, allowing agents to make good decisions based on their own observations. The introduction of information fusion increases the complexity of learning, which slows down the agent's learning process. In this scenario, QTRAN converges slowly but eventually reaches a winning rate of about 71%, while COMA struggles to converge, with a winning rate of only around 20%.
- In the 1c3s5z and mmm scenarios, QTRAN performs poorly, with a winning rate of around 30%, while COMA fails to converge. QMIX-GNN and QMIX achieve relatively high winning rate, with QMIX-GNN showing varying degrees of improvement over QMIX, suggesting that in more complex scenarios the agents need to focus more on information from other agents in order to achieve better cooperation.

Table 5. Comparison of winning rates.

Winning Rate	3m	2s3z	1c3s5z	10m_vs_11m	mmm	mmm2
COMA	0.778	0.203	0.285	0.252	0.015	0.008
QTRAN	0.991 *	0.720	0.713	0.717	0.316	0.167
QMIX	0.956	0.969	0.846	0.855	0.877	0.542
QMIX-GNN	0.967	0.957	0.900	0.892	0.928	0.655
Comparison with QMIX	0.011	−0.012	0.054	0.037	0.051	0.113

* The best results are highlighted in bold.

The training processes for the four scenarios are shown in Figure 8, while the training steps required for each model to reach an 80% winning rate are shown in Table 6:

- In the 3m scenario, all four models converge relatively quickly and maintain a stable winning rate, although COMA only manages to maintain a relatively low winning rate.
- In the 2s3z scenario, both QMIX and QMIX-GNN converge quickly and maintain a high winning rate, but QMIX has a more stable convergence curve with smaller fluctuations, while COMA exhibits an unstable convergence curve and performs poorly. In comparison, QTRAN converges slowly.
- In the 1c3s5z, 10m_vs_11m, mmm, and mmm2 scenarios, QTRAN converges slowly and COMA fails to converge. Both QMIX and QMIX-GNN show good convergence performance, with QMIX-GNN converging significantly faster than QMIX and demonstrating better overall training results.

Table 6. Number of training episodes to reach an 80% winning rate.

Episodes	3m	2s3z	1c3s5z	10m_vs_11m	mmm
COMA	820k	∞	∞	∞	∞
QTRAN	150k	∞	∞	∞	∞
QMIX	140k	300k	730k	680k	800k
QMIX-GNN	120k	410k	380k	420k	640k

3.4. Ablation Study

To evaluate the impact of the proposed modules in QMIX-GNN, we designed an ablation study in the 1c3s5z scenario to analyze the contributions of each component to overall model performance. Specifically, we compared three variants: the original QMIX model, our proposed QMIX-GNN, and QMIX-HT, i.e., QMIX with our heterogeneous information processing structures but without the GNN-based information fusion module. The training curves are shown in Figure 9 and a performance comparison is provided in Table 7.

Table 7. Performance comparison from the ablation study.

Model	QMIX	QMIX-HT	QMIX-GNN
Winning rate	0.834	0.861	0.913
Improvement	-	0.027	0.079

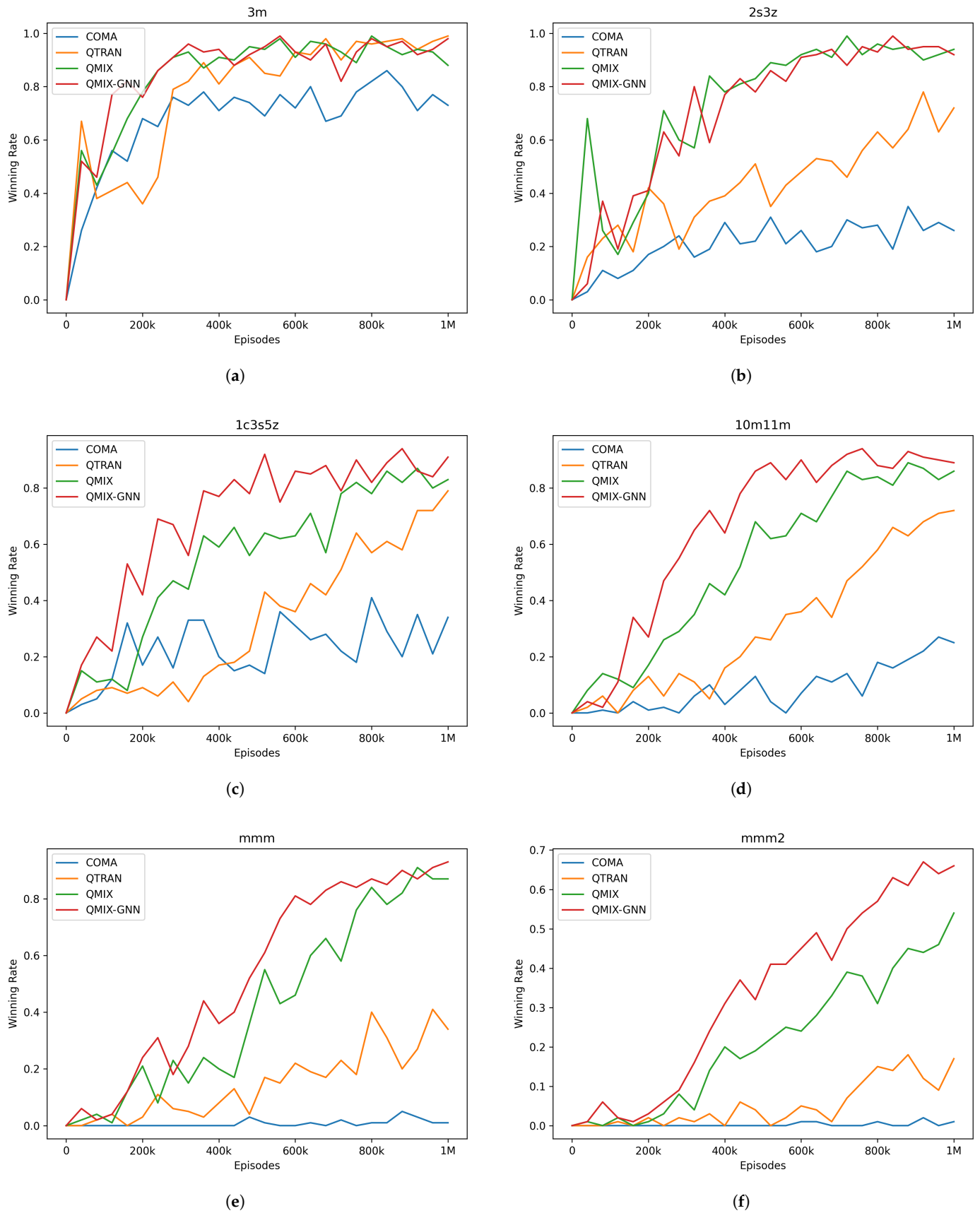


Figure 8. Training results for the six scenarios: (a) 3m, (b) 2s3z, (c) 1c3s5z, (d) 10m_vs_11m, (e) mmm, and (f) mmm2.

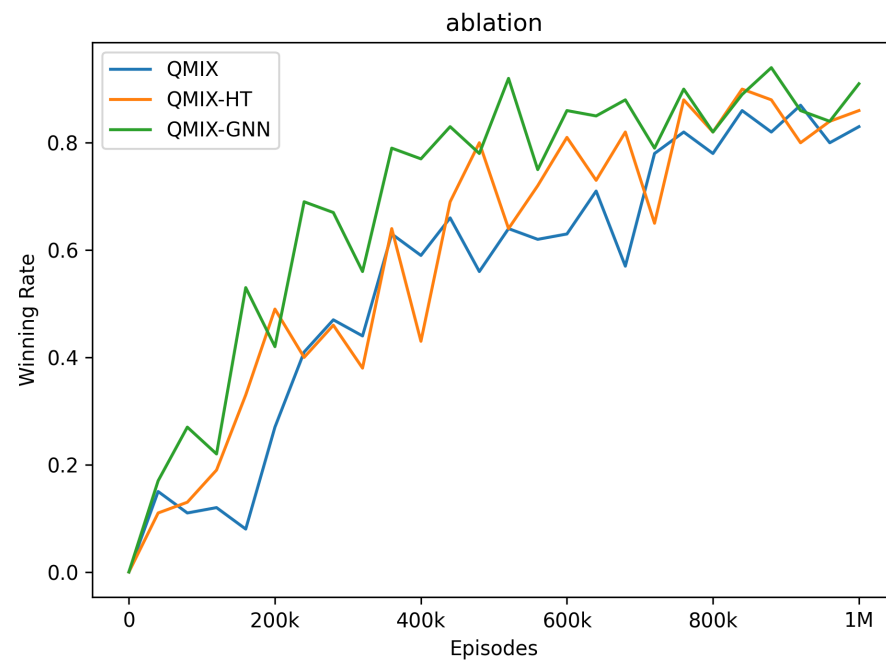


Figure 9. Training curves from the ablation study.

The experimental results of the ablation study indicate that QMIX-HT improves the winning rate by 0.027 over the baseline QMIX. More notably, QMIX-GNN shows a winning rate improvement of 0.079 compared to QMIX and 0.052 compared to QMIX-HT, as well as faster convergence. These findings demonstrate that both the heterogeneous information processing structures and the GNN-based information fusion module play significant roles in enhancing inter-agent collaboration and contribute positively to model performance.

4. Discussion

In this section, we discuss the limitations of QMIX-GNN and some potential future work directions:

- In simpler scenarios such as 3m and 2s3z, both QMIX-GNN and QMIX achieve strong performance, indicating that the advantages of our approach are less pronounced in low-complexity settings. However, the benefits of QMIX-GNN become more evident as task complexity increases. In medium-difficulty scenarios such as 1c3s5z and 10m_vs_11m, our method converges faster than QMIX, demonstrating superior sample efficiency. In challenging scenarios such as MMM and MMM2, QMIX-GNN consistently outperforms QMIX, underscoring its effectiveness in handling complex coordination tasks. This performance trend can be attributed to our information infusion mechanism, which enhances agent coordination by leveraging structured relational data. While this mechanism significantly improves performance in complex environments, it introduces additional computational overhead that may not be necessary in simpler tasks. Additionally, our use of a k -nearest graph based on inherent SMAC features such as visual and attack radii imposes certain constraints on QMIX-GNN's applicability. Future work could explore adaptive strategies that dynamically adjust the degree of information infusion based on task complexity, which would enhance flexibility and reduce unnecessary learning overhead in less demanding scenarios.

- Our proposed QMIX-GNN is fundamentally reliant on the effective fusion of information across multiple agents to enable collaborative decision-making. Scenarios in which agents operate in complete isolation, such as decentralized systems lacking communication channels, are outside the primary scope of our work; nevertheless, this assumption imposes practical limits on the application of QMIX-GNN. In environments where robust inter-agent communication and information sharing cannot be guaranteed, the underlying information fusion mechanism may be ineffective, potentially leading to degraded performance. Future work could explore the design of lightweight or privacy-aware fusion strategies that would simplify the fusion architecture or incorporate privacy-preserving mechanisms while still enabling collaborative information sharing. This would help to extend the practical applicability of QMIX-GNN to a broader range of scenarios.
- Although SMAC presents certain challenges, the current infusion mechanism in QMIX-GNN is specifically designed around the unique features of the SMAC, as mentioned above. As a result, while our experiments demonstrate promising performance in this context, the effectiveness of QMIX-GNN in more complex environments remains to be validated. Because the SMAC does not support macro mode, extending our evaluation to such settings is left for future work. In addition, future research could broaden the applicability of QMIX-GNN by focusing on development of a more general infusion mechanism that is adaptable to a wider range of environments.

5. Conclusions

To address the issue of agents in the Centralized Training and Decentralized Execution (CTDE) framework struggling to make correct decisions based solely on their own observations, this paper proposes a heterogeneous Multi-Agent Reinforcement Learning (MARL) model based on Graph Neural Networks (GNNs), which we name QMIX-GNN. QMIX-GNN effectively handles the inconsistent feature and action spaces of heterogeneous agents through the use of matrix projection and a tower-like structure, ensuring that diverse agent types can be processed appropriately. In addition, by employing GNNs to fuse information across multiple agents, our model endows each agent with a more comprehensive view of the environment during the execution phase, resulting in significantly enhanced decision-making quality.

Experimental results in the SMAC environment demonstrate that QMIX-GNN outperforms existing methods, particularly in complex scenarios where collaboration and coordination are critical. Ablation studies further validate the contributions of our proposed information infusion and heterogeneous information processing modules. While the current work is based on the SMAC framework, which inherently influences the design of our infusion mechanism, our findings can provide valuable insights into methods for improving scalability and performance in heterogeneous environments.

Future work may focus on refining the information fusion mechanism to adaptively adjust to varying task complexities and exploring privacy-aware fusion strategies to broaden the applicability of our method in scenarios with decentralized or communication-constrained agents. We believe that these enhancements will further bridge the gap between theoretical advancements and real-world applications, offering a promising direction for future research into multi-agent reinforcement learning.

Overall, our proposed QMIX-GNN model not only addresses key limitations in existing CTDE approaches but also establishes a robust foundation for developing more effective and scalable MARL systems in heterogeneous environments.

Author Contributions: Conceptualization, T.Z. and B.Z.; methodology, T.Z. and B.Z.; software, B.Z. and T.C.; validation, T.Z., B.Z. and T.C.; formal analysis, B.Z.; investigation, T.Z.; resources, T.Z.; data curation, B.Z.; writing—original draft preparation, B.Z.; writing—review and editing, T.Z. and B.Z.; visualization, B.Z. and T.C.; supervision, T.Z.; project administration, T.Z.; funding acquisition, T.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Chengdu Key Research and Development Support Program “Jie Bang Gua Shuai” Project under grant number 2023-JB00-00012-GX.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All experiments are conducted on the StarCraft Multi-Agent Challenge platform (<https://github.com/oxwhirl/smac>, accessed on 10 June 2023).

Acknowledgments: We would like to express our sincere gratitude to everyone who contributed to the development of this paper for their valuable support and insightful feedback.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MARL	Multi-Agent Reinforcement Learning
GNN	Graph Neural Network
GAT	Graph Attention Network
GRU	Gated Recurrent Unit
RNN	Recurrent Neural Network

References

1. Ning, Z.; Xie, L. A survey on multi-agent reinforcement learning and its application. *J. Autom. Intell.* **2024**, *3*, 73–91. [\[CrossRef\]](#)
2. Gronauer, S.; Diepold, K. Multi-agent deep reinforcement learning: A survey. *Artif. Intell. Rev.* **2021**, *55*, 895–943. [\[CrossRef\]](#)
3. Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv* **2017**, arXiv:1706.05098.
4. Wen, Y.; Qian, F.; Guo, W.; Zong, J.; Peng, D.; Chen, K.; Hu, G. VSP Upgoing and Downgoing Wavefield Separation: A Hybrid Model-Data Driven Approach. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 5908014. [\[CrossRef\]](#)
5. Qian, F.; Pan, S.; Zhang, G. *Tensor Computation for Seismic Data Processing: Linking Theory and Practice*; Earth Systems Data and Models Series; Springer: Berlin, Germany, 2025.
6. Kong, G.; Chen, F.; Yang, X.; Cheng, G.; Zhang, S.; He, W. Optimal Deception Asset Deployment in Cybersecurity: A Nash Q-Learning Approach in Multi-Agent Stochastic Games. *Appl. Sci.* **2024**, *14*, 357. [\[CrossRef\]](#)
7. Wilson, A.; Menzies, R.; Morarji, N.; Foster, D.; Mont, M.C.; Turkbeyler, E.; Gralewski, L. Multi-Agent Reinforcement Learning for Maritime Operational Technology Cyber Security. *arXiv* **2024**. [\[CrossRef\]](#)
8. Du, J.; Yu, A.; Zhou, H.; Jiang, Q.; Bai, X. Research on Integrated Control Strategy for Highway Merging Bottlenecks Based on Collaborative Multi-Agent Reinforcement Learning. *Appl. Sci.* **2025**, *15*, 836. [\[CrossRef\]](#)
9. Garcia-Cantón, S.; Ruiz de Mendoza, C.; Cervelló-Pastor, C.; Sallent, S. Multi-Agent Reinforcement Learning-Based Routing and Scheduling Models in Time-Sensitive Networking for Internet of Vehicles Communications Between Transportation Field Cabinets. *Appl. Sci.* **2025**, *15*, 1122. [\[CrossRef\]](#)
10. Mamond, A.W.; Kundroo, M.; Yoo, S.e.; Kim, S.; Kim, T. FLDQN: Cooperative Multi-Agent Federated Reinforcement Learning for Solving Travel Time Minimization Problems in Dynamic Environments Using SUMO Simulation. *Sensors* **2025**, *25*, 911. [\[CrossRef\]](#) [\[PubMed\]](#)
11. Zhang, C.; Zhou, X.; Xu, C.; Wu, Z.; Liu, J.; Qi, H. Automatic Generation of Precast Concrete Component Fabrication Drawings Based on BIM and Multi-Agent Reinforcement Learning. *Buildings* **2025**, *15*, 284. [\[CrossRef\]](#)
12. Sharma, P.K.; Fernandez, R.; Zaroukian, E.; Dorothy, M.; Basak, A.; Asher, D.E. Survey of recent multi-agent reinforcement learning algorithms utilizing centralized training. In *Proceedings of the Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*; SPIE: Online Only, FL, USA, 2021; Volume 11746, pp. 665–676.

13. Xiao, L.; Wu, X.; Wu, W.; Yang, J.; He, L. Multi-channel attentive graph convolutional network with sentiment fusion for multimodal sentiment analysis. In Proceedings of the ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 23–27 May 2022; pp. 4578–4582.
14. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [[CrossRef](#)] [[PubMed](#)]
15. Liu, Z.; Zhang, J.; Shi, E.; Liu, Z.; Niyato, D.; Ai, B.; Shen, X. Graph Neural Network Meets Multi-Agent Reinforcement Learning: Fundamentals, Applications, and Future Directions. *IEEE Wirel. Commun.* **2024**, *31*, 39–47. [[CrossRef](#)]
16. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
17. Ding, S.; Du, W.; Ding, L.; Zhang, J.; Guo, L.; An, B. Multiagent Reinforcement Learning With Graphical Mutual Information Maximization. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, 1–10. [[CrossRef](#)]
18. Huang, J.; Su, J.; Chang, Q. Graph neural network and multi-agent reinforcement learning for machine-process-system integrated control to optimize production yield. *J. Manuf. Syst.* **2022**, *64*, 81–93. [[CrossRef](#)]
19. Leibfried, F.; Grau-Moya, J. Mutual-information regularization in Markov decision processes and actor-critic learning. In Proceedings of the Conference on Robot Learning, Graz, Austria, 9–12 July 2020; pp. 360–373.
20. A. Oliehoek, F.; Amato, C.A. *A Concise Introduction to Decentralized POMDPs*; Springer: Berlin, Germany, 2016.
21. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
22. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **2020**, *21*, 1–51.
23. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
24. Son, K.; Kim, D.; Kang, W.J.; Hostallero, D.E.; Yi, Y. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 5887–5896.
25. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
26. Zhu, C.; Dastani, M.; Wang, S. A survey of multi-agent deep reinforcement learning with communication. *Auton. Agents-Multi-Agent Syst.* **2024**, *38*, 4. [[CrossRef](#)]
27. Samvelyan, M.; Rashid, T.; De Witt, C.S.; Farquhar, G.; Nardelli, N.; Rudner, T.G.; Hung, C.M.; Torr, P.H.; Foerster, J.; Whiteson, S. The starcraft multi-agent challenge. *arXiv* **2019**, arXiv:1902.04043.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.