applied sciences

MDPI

Article

DDWCN: A Dual-Stream Dynamic Strategy Modeling Network for Multi-Agent Elastic Collaboration

Zhenduo Meng, Xitai Na *, Tuo Wang, Jinglong Liu and Wenda Wang

School of Electronic and Information Engineering, Inner Mongolia University, Hohhot 010021, China; mzd123358@163.com (Z.M.); 32256149@mail.imu.edu.cn (T.W.); jingloog666@gmail.com (J.L.); 32256158@mail.imu.edu.cn (W.W.)

* Correspondence: 111991009@imu.edu.cn

Abstract

In the domain of multi-agent reinforcement learning, conventional algorithms such as VDN, QMIX, and QTRAN have exhibited favorable performance in static task scenarios. However, these algorithms encounter challenges due to their limited capacity to model elastic collaboration scenarios, wherein the number of agents and the state-action space undergo dynamic changes over time. To address this limitation, this paper proposes a novel multi-agent collaboration mechanism: the Dual-Stream Dynamic Weight Compensation Network (DDWCN). This method employs a dual-stream action modeling network for the classification and processing of actions. It integrates an information compensation network and a dynamic weight fusion network. These additions have been demonstrated to enhance the network's robustness and generalization capabilities for complex collaborative tasks. Extensive experimentation on a variety of benchmark tasks in StarCraft II has validated the effectiveness of DDWCN. The findings indicate that this approach exhibits commendable scalability and reliability in authentic multi-agent collaboration scenarios, underscoring its extensive applicability.

Keywords: multi-agent; dual-stream action modeling network; information compensation network; dynamic weight fusion network

check for updates

Academic Editor: Juan Francisco De Paz Santana

Received: 22 July 2025
Revised: 14 August 2025
Accepted: 17 August 2025
Published: 20 August 2025

Citation: Meng, Z.; Na, X.; Wang, T.; Liu, J.; Wang, W. DDWCN: A Dual-Stream Dynamic Strategy Modeling Network for Multi-Agent Elastic Collaboration. *Appl. Sci.* **2025**, *15*, 9164. https://doi.org/10.3390/app15169164

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) is a method that applies the principles of reinforcement learning to multi-agent systems. As multi-agent systems are increasingly applied in real-world scenarios, such as autonomous driving and intelligent transportation systems [1,2], drone swarms [3], and multi-robot collaboration [4], MARL algorithms have attracted growing attention from researchers, becoming a focal point in both academia and industry [5]. In comparison with single-agent systems, multi-agent systems encounter challenges such as strategic instability, environmental non-stationarity, and high-dimensional state-action spaces due to multi-agent games and collaboration. In order to address these challenges and improve the efficiency and stability of multi-agent training, researchers initially proposed two solutions: independent learning [6] and centralized learning [7]. Independent learning regards other agents as components of the environment and adjusts its own strategy without reliance on the strategies employed by other agents. In centralized learning, all agents are regarded as a unified entity, with all agents' actions, states, and reward information being processed collectively by a central controller. This controller optimizes strategies based on the global context. However, both independent

learning and centralized learning have their respective drawbacks. Independent learning is vulnerable to disruptions caused by environmental non-stationarity, which can affect the convergence speed and stability. Conversely, centralized learning is characterized by significant communication overhead, limited scalability, and a susceptibility to dimensionality explosion issues.

In order to achieve a more balanced integration of the benefits inherent in both of these approaches while concomitantly circumventing their respective disadvantages, a novel framework known as Centralized Training and Decentralized Execution (CTDE) has been put forth [8]. CTDE represents a synthesis of centralized learning and independent learning, integrating the decentralized approach of independent learning with the information-sharing benefits of centralized learning, while addressing the stability and coordination challenges inherent in both approaches. It is currently the most practical and scalable mainstream paradigm. During the training phase, CTDE enables agents to share global information, thereby promoting the collaborative optimization of multi-agent collective strategies. During the execution phase, it enables agents to make decisions based on their own environment and state, thereby maintaining their autonomous decision-making capabilities and promoting the optimization of single-agent strategies. Consequently, CTDE instigates a virtuous cycle, thereby balancing the coordination and scalability of multi-agent systems.

Due to its excellent performance characteristics, CTDE has attracted widespread attention from researchers in the relevant academic fields. Inspired by the CTDE framework, Sunehag P et al. [9] proposed a value decomposition network (VDN), which allows each agent to explore using a greedy strategy and then linearly sums the rewards of each agent to obtain a global reward value. Rashid T et al. [10] proposed the QMIX algorithm, which is based on VDN. This algorithm forces each agent's direction to align with the group's overall direction. This, in turn, improves algorithm performance. Furthermore, the extant literature [11] demonstrates the capacity to incorporate monotonic functions within a fusion network. However, it should be noted that the aforementioned methods are only applicable to simple linear scenarios and perform poorly in discrete, complex relational scenarios. In addressing the intricate challenges posed by discrete environments, Son K et al. [12] proposed the QTRAN algorithm, drawing inspiration from QMIX. The proposed approach involves a decomposition of the global value from the group level to the individual level. This process enables the dissolution of the linear scenario constraints imposed by VDN and QMIX, thereby enhancing the adaptability of multi-agent algorithms. Wang J et al. [13] proposed the QPLEX algorithm, which employs a dual-duel network architecture to decompose the joint value function. The algorithm encodes IGM principles into the neural network architecture to achieve efficient value function learning. Rashid et al. [14] have demonstrated an enhancement to the training objective of QMIX, proposing the WQMIX algorithm. This algorithm utilizes a weighted strategy for the TD error of Q_tot during training, thereby ensuring a more stable and robust training process.

The aforementioned MARL algorithm is designed based on value decomposition, while the MARL algorithm based on the actor-observer model also demonstrates good performance. Foerster J et al. [15] proposed the COMA algorithm, which is based on the actor-observer model. This algorithm was developed to address the challenge of credit allocation among multiple agents. This algorithm independently evaluates the contribution of each agent's actions and designs joint Q-values based on the magnitude of contributions to optimize policy gradient updates. Iqbal S et al. [16] proposed the MAAC algorithm, which models the states and actions of other agents through an attention mechanism to enhance agent performance. Yu C et al. [17] proposed the MAPPO algorithm, which updates strategies using centralized evaluation and policy constraints to ensure robustness during training. The HAPPO algorithm, proposed by Kuba JG et al. [18], is a

sophisticated system that optimizes strategies for diverse types of agents, with the objective of enhancing the collaborative capabilities of heterogeneous agents. In their seminal work, Mahajan A et al. [19] proposed the MAVEN algorithm, a pioneering system that integrates collective knowledge from the group into collective strategy learning. This innovative approach has demonstrated remarkable efficacy in collaborative tasks, underscoring its potential for addressing complex problems in collaborative settings. Wang T. et al. [20] proposed the RODE algorithm, which introduces a decoupling mechanism in the actor-critic structure with the objective of enhancing the algorithm's generalization ability and training sample utilization. Haarnoja et al. [21] proposed the SAC algorithm, which introduced maximum entropy reinforcement learning methods. These methods incorporate state-action entropy into the policy objective, thereby enhancing the agent's exploration capabilities and efficiency. Furthermore, the SAC algorithm promotes diversity in multi-agent strategies. Peng B. et al. [22] proposed the FACMAC algorithm, which employs non-monotonic decomposition for efficient modeling of the joint value function and uses a centralized policy gradient estimator to optimize the entire joint action space. In their seminal work, Bo Y et al. [23] proposed a novel framework for the RECO algorithm. This framework is predicated on the reuse of experience, and it employs a hierarchical experience pool mechanism. This mechanism is designed to enhance exploration through strategic reward redistribution and experience reuse. Zhao Z et al. [24] introduced the concept of latent interactions. Their proposed method derives weights from historical information to improve the accuracy of value estimation.

These MARL algorithms demonstrate efficacy in evaluating training effects; however, their performance is frequently deficient in scenarios of elastic collaboration, wherein the number of agents and the extent of the state-action space undergo fluctuations over time. In order to address the challenges posed by increasingly complex elastic collaboration scenarios, researchers have proposed numerous solutions. Bo Liu [25] proposed the coach-agent architecture, which dynamically distributes strategies to each agent and uniformly coordinates the global team configuration. This architecture is particularly well-suited to scenarios where the number of agents or the capabilities of individual agents change, as it addresses these scenarios more effectively. Wang W. et al. [26] developed a dynamic agent digital network (DyAN) to process network inputs in a dynamic manner, with the capacity to adapt to fluctuations in the number of agents and to accommodate elastic collaboration scenarios. In their seminal work, Tang et al. [27] proposed a novel few-shot learning algorithm that addresses the challenges posed by the dynamic nature of multi-agent systems. In scenarios where agents may temporarily exit or join during the training phase, this algorithm enables rapid knowledge transfer from existing agents to newly joined agents. This, in turn, accelerates the convergence of multi-agent strategies in elastic collaboration scenarios.

Although the aforementioned algorithms exhibit satisfactory performance in elastic collaboration scenarios, they lack the capability to classify and process different categories of agent actions, resulting in insufficient adaptability to diverse environments and leaving ample room for further improvement. Moreover, in MARL algorithms, the modeling approach of actions directly influences the agents' ability to compete and collaborate. Conventional MARL methods tend to treat elastic collaboration tasks as static scenarios, indiscriminately processing Q-values for different types of actions. When the number of agents decreases due to death or changes in action space, traditional methods typically overlook the dynamic variations introduced by agent failures. Instead, they continue to process the invalid actions generated by the deceased agents and produce new, invalid Q-values to fill the state-action space. This not only leads to computational redundancy but also introduces decision noise, thereby undermining the accuracy of agent decision-making.

In response to the aforementioned challenges, a new multi-agent collaboration mechanism is proposed. This mechanism is referred to as the DDWCN, and it is based on the limitations of conventional MARL algorithms. DDWCN categorizes agent actions into two distinct categories: dimension-varying attack actions and dimension-invariant non-attack actions. To process these actions independently, DDWCN establishes two separate data stream networks. The model introduces two novel mechanisms: a weight fusion scheduling mechanism and an information compensation mechanism. The former applies dynamic weighting to action information, while the latter applies residual correction to action semantics. The introduction of DDWCN is intended to enhance the efficiency of action information processing in multi-agent systems and to promote stability and speed of strategy convergence in dynamic collaboration scenarios.

The core contributions of this study are summarized as follows, forming a coherent framework from problem formulation to practical validation:

- Innovative algorithmic formulation. Targeting the modeling limitations of heterogeneous actions in multi-agent systems, this study proposes a novel MARL framework named DDWCN. The core of DDWCN lies in the design of a Dual-stream Action Modeling Network (DAMN), which explicitly separates different action types. This separation not only reduces interference from redundant actions but also enhances the precision of policy learning under complex cooperation dynamics;
- Adaptive value fusion mechanism. To further improve the adaptability of action value integration, a Dynamic Weight Fusion Network (DWFN) is constructed. DWFN dynamically fuses Q-values from the two action branches by computing scenario-dependent fusion weights. This mechanism enables the model to maintain decision consistency and learning stability across diverse environmental conditions;
- Information compensation enhancement. Recognizing the potential loss introduced by value fusion, an Information Compensation Network (ICN) is developed to enhance the integrity of Q-value representations. ICN compensates for fusion-induced value distortion through residual-based correction, thus preserving key action semantics and improving the expressiveness of joint decision-making;
- Comprehensive empirical validation. The proposed framework is thoroughly evaluated on the StarCraft II benchmark. Through comparative experiments with representative MARL algorithms, DDWCN demonstrates consistent advantages in convergence speed, robustness, and adaptability, validating its practical potential in elastic collaborative environments.

Following a series of empirical trials, the findings indicated that, in comparison with conventional multi-agent algorithms, DDWCN exhibited superior performance in diverse scenarios in StarCraft II [28] (e.g., 8m, 2s3z, 3s5z), demonstrating higher win rates and enhanced stability.

2. Method Structure

In this section, we put forth a novel MARL research paradigm and introduce the DDWCN algorithm. First, DAMN is constructed and analyzed, with the workflow and theoretical foundations of DAMN being systematically elucidated. Consequently, we propose DWFN, a module capable of adaptively weighting different types of actions based on environmental changes. Furthermore, the ICN has been developed to address the issue of missing Q-value information during weight fusion, thereby enhancing the overall system's robustness. To comprehensively understand the operational principles and structural design of DDWCN, this paper outlines the architecture of its three core submodules, thereby unveiling its underlying mechanisms and deepening the understanding of its functional logic and implementation pathways.

2.1. Dual-Stream Action Modeling Network

In scenarios involving elastic collaboration, the demise of an agent not only results in alterations in the number of agents but also in the size and dimensions of the state-action space. In the context of multi-agent tasks, the action sets of agents can be categorized into two distinct classifications: environment-oriented action sets and agent-oriented action sets. In StarCraft II, environmental actions are exhibited as non-attack actions by agents, including, but not limited to, upward movement, leftward movement, and cessation of movement. In StarCraft II, actions directed at other AI units are often offensive, such as attacking AI unit A or AI unit B. Therefore, the action set can be divided into non-offensive and offensive action sets. The former refers to the set of actions through which the AI interacts with the environment. Throughout the training process, the size and dimensions of this set remain constant, regardless of the death of AI units. The latter refers to the set of actions that the current agent performs when interacting with other agents. During the training phase, the size and dimension of this action set are subject to variation, contingent upon the number of agents. In this scenario, if a traditional action value network is used to treat non-attack actions and attack actions indiscriminately, the model will easily generate a large number of invalid or even incorrect action valuations. This will trigger strategy transfer among multiple agents and affect the accuracy of decision-making.

To address the aforementioned issues, DDWCN constructed the DAMN, which separately constructed two different action value modeling networks for attack actions and non-attack actions to generate dynamic Q-values adapted to different types of actions. The structure of DAMN is illustrated in Figure 1.

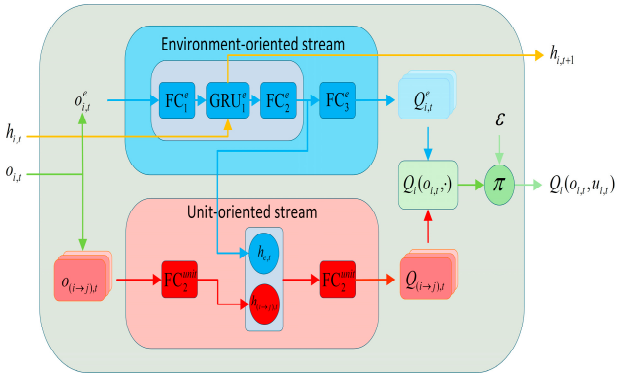


Figure 1. DAMN structure diagram.

DAMN employs a two-pronged approach by constructing two distinct network flows for two distinct types of action sets: environment-oriented and agent-oriented. Environment-oriented flows are predicated on environment-oriented observations. The model under consideration consists of three fully connected layers FC^e_j (j = 1, 2, 3) and a Gated Recurrent Unit GRU^e_1 [29]. The Q-value for environment-oriented actions is the result of this process and is expressed as follows:

$$Q^e_{i,t}(o^e_{i,t}, h_{i,t}, \cdot), h_{i,t+1} = \mathrm{NN}^e_i(o^e_{i,t}, h_{i,t}) \tag{1}$$

$Q^e_{i,t}$ represents the value when intelligent agent i performs environment-oriented actions, that is, non-attack action values. In addition, $o^e_{i,t}$ represents the environment-oriented observation values observed by intelligent agent i from the environment. The hidden vector of the next-time step is represented by $h_{i,t+1}$, and NN^e_i is an abbreviation for environment-oriented flow network.

The network flow for intelligent agent units utilizes the unit's observation value $o^e_{(i\to j),t}$ as an input, and its output is $Q_{(i\to j),t}$ for the intelligent agent unit's action, i.e., the

attack action Q-value. The distinction between unit-oriented observation value $o^e_{(i\to j),t}$ and environment-oriented observation value $o^e_{i,t}$ lies in their respective roles in describing the interaction information between the intelligent agent i and the target intelligent agent j, and the information observed by the intelligent agent i from the environment, respectively. The Q-value calculation for the intelligent agent unit is expressed as follows:

$$vector = [h_{e,t}, h_{(i\to j),t}] \tag{2}$$

$$Q_{(i\to j),t}(o_{(i\to j),t}, u_{(i\to j),t}) = \mathrm{FC}_2^{\mathrm{unit}}(vector) \tag{3}$$

vector is the concatenation of $h_{e,t}$ and $h_{(i\to j),t}$, where $h_{e,t}$ and $h_{(i\to j),t}$ are the outputs of the first stage of the environment-oriented flow and the agent-oriented flow, respectively. The second-layer network of the agent-oriented flow is represented by $\mathrm{FC}_2^{\mathrm{unit}}$. In conclusion, the DAMN generates two distinct types of Q-values: environment-oriented non-aggressive action Q-values and agent-oriented aggressive action Q-values. This action-type-aware modeling approach enhances the rationality and efficiency of handling different categories of actions in multi-agent policy learning.

2.2. Dynamic Weight Fusion Network

Although DDWCN constructs the DAMN to perform separate modeling for different types of actions and addresses the information expression bottleneck in elastic collaborative scenarios, the DAMN in DDWCN still exhibits coarse-grained fusion processing of aggressive and non-aggressive actions during the individual Q-value modeling stage, failing to reflect the significant differences in action semantic importance. This section proposes the DWFN based on action semantic differentiation with the objective of further enhancing action semantic differentiation and the robustness of joint Q-values. This network performs dynamic weight fusion processing on the Q-values of aggressive and non-aggressive actions, enhancing the fine-grained expression capability of individual Q-values during the nonlinear fusion process. This, in turn, strengthens action semantic differentiation and the robustness of joint Q-values.

In the context of StarCraft II, which is characterized by a high degree of heterogeneity in its task scenarios, individual agents are confronted with action sets that can be categorized into two distinct groups: the first category comprises attack actions, such as attacking Agent A or attacking Agent B, among others, while the second category encompasses non-attack actions, including movements such as moving left or right and halting. The impact of these two types of action sets on agent strategies varies significantly across different scenarios, and these differences evolve over time. However, in conventional MARL algorithms, there is often an oversight of the impact of differences in action semantic importance. Instead, a direct concatenation approach is used to simplify the processing of Q-values for different actions. This absence of differentiated processing based on semantic importance impedes the strategy's capacity to model essential behaviors, potentially diminishing the influence of critical action Q-values and resulting in the deterioration of multi-agent strategies. Consequently, this phenomenon affects the effectiveness and stability of agent decision-making.

To address this issue, the proposed DDWCN framework extends DAMN by incorporating the DWFN to achieve discriminative processing of heterogeneous action categories. The architectural details of DWFN are illustrated in Figure 2.

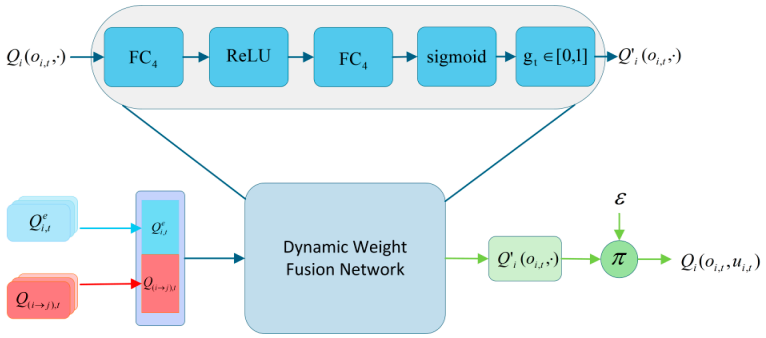


Figure 2. DWFN structure diagram.

The DWFN introduces a gate structure composed of neural networks that dynamically adjusts the influence of two types of actions on the joint strategy at the semantic level. In this context, the gate structure is defined as a neural network submodule specifically designed to generate dynamic fusion weights for attack and non-attack actions, rather than serving as a general-purpose network for feature extraction. This specialized design enables the gate to focus on capturing semantic importance differences between the two action types and to produce adaptive weights accordingly. It then outputs a normalized fusion weight value $g_t \in [0, 1]$, which represents the importance of non-attack actions in the overall strategy. This gate mechanism captures differences in the importance of attack and non-attack actions in different scenarios. It provides more accurate behavioral strategy guidance for subsequent strategies. The calculation process for the g_t value of the gated network is as follows:

$$g_t = \sigma(W_2 \cdot \mathrm{ReLU}(W_1 \cdot (Q^e_{i,t} + Q_{(i\to j),t}) + b_1) + b_2) \tag{4}$$

W_1 and W_2 are the weights of the DWFN and b_1 and b_2 are the corresponding bias values. $\mathrm{ReLU}(\cdot)$ is a nonlinear activation function that introduces nonlinearity to the features. $\sigma(\cdot)$ is the sigmoid function that normalizes the g_t value of the gate network to ensure that g_t remains within the range of 0 to 1. After extracting and processing the gate weight value g_t, the DWFN applies adaptive processing to the two action branches. This enables adaptive fusion processing based on different action Q-values, generating the corresponding fused action Q-values Q'_i. The Q'_i function expression is as follows:

$$Q'_i = g_t \cdot Q^e_{i,t} + (1 - g_t) \cdot Q_{(i\to j),t} \tag{5}$$

$Q^e_{i,t}$ denotes the Q-value for non-attack actions, while $Q_{(i\to j),t}$ represents the Q-value for attack actions. The aforementioned formula enables the gate mechanism to dynamically adjust the influence of the Q-values for the two types of actions on the global strategy via the g_t value. The closer the g_t value is to 1, the greater the influence of non-attack actions on the global strategy; the closer the g_t value is to 0, the greater the influence of attack actions on the global strategy; and when the g_t value approaches 0.5, the influence of the two types of actions is roughly equal. This DWFN enhances the joint value's ability to express different action values. It also enables multi-agent systems to adaptively schedule the influence of attack actions and non-attack actions when facing different environments and states. This improves the depth and flexibility of strategy selection.

2.3. Information Compensation Network

The DWFN has been demonstrated to adjust the Q-values of different types of actions based on differences in action semantic importance. This enhances the adaptability and scheduling capabilities of actions across different stages. However, dynamically processing

Q-values essentially constitutes an information compression process, which can result in information loss and overly strong action biases. This phenomenon is particularly salient during the model's initialization phase, prior to the attainment of optimal training conditions for g_t values. The consequence of these factors can be the emergence of unstable multi-agent strategies and diminished joint Q-value accuracy, consequently impacting the convergence of training decisions.

In order to address the aforementioned potential issues, DDWCN introduces the ICN to compensate for lost information, an approach inspired by residual networks [30]. The ICN employs a residual modeling approach, leveraging the differences in action Q-values before and after fusion. It utilizes this information to reconstruct the lost data by capturing the disparities between the Q-values and compensating for the relevant Q-values. The incorporation of the ICN module is primarily driven by two theoretical considerations. First, it is essential to accurately capture the discrepancies between Q-values before and after fusion and to compensate for the information contained within the fused Q-values, thereby mitigating potential information loss. Second, such compensation serves to regulate excessive variations in Q-value information across the transformation process, which could otherwise induce instability in policy learning, while ensuring the relative stability of the Q-value fusion mechanism.

As illustrated in Figure 3, DDWCN introduces an explicit modeling residual signal compensation subnetwork, ICN, based on DWFN to restore the local high-dimensional information that is lost during the fusion stage. The subsequent calculation implementation process is delineated as follows. Initially, the residual value between the Q-value subsequent to dynamic weight fusion and the original Q-value must be calculated to accurately capture the discrepancy:

$$\Delta_i = Q'_i - Q_i \tag{6}$$

$$\hat{\Delta}_i = f_{comp}(\Delta_i) = W_r^2 \cdot \mathrm{ReLU}(W_r^1 \cdot \Delta_i + b_r^1) + b_r^2 \tag{7}$$

Q'_i denotes the Q-value subsequent to fusion, Q_i signifies the Q-value prior to fusion, and Δ_i represents the residual value of the Q-value before and after the fusion transformation. The compensation vector $\hat{\Delta}_i$, is derived from the residual value. The weight parameters of the ICN, W_r^1 and W_r^2, are defined accordingly, where they are randomly initialized at the start of training and subsequently optimized via backpropagation together with the other network parameters of DDWCN. The bias terms, b_r^1 and b_r^2, are similarly initialized to zero and updated through backpropagation during training. In order to guarantee the stability of the compensation magnitude scale, the compensation vector $\hat{\Delta}_i$, derived from the residual values, is uniformly standardized using LayerNorm [31], which is implemented in PyTorch (version 1.10.0, available at https://pytorch.org/, accessed on 10 July 2025).with the function calculation as follows:

$$\mathrm{H} = \lambda \cdot \mathrm{LayerNorm}(\hat{\Delta}_i) \tag{8}$$

$$Q_i^{final}(o_{i,t}, \cdot) = Q'_i(o_{i,t}, \cdot) + \mathrm{H} \tag{9}$$

among these, λ is the residual compensation coefficient, with a value range of [0, 1], used to control the correction magnitude of the residual compensation term on the Q_i^{final} value. The λ value is generally set to 0.2 for optimal results. This figure denotes the ultimate Q_i^{final} value subsequent to processing through DDWCN. DDWCN is characterized by the following functional advantages offered by the ICN module: The primary advantage of this approach is that it explicitly addresses information loss during the fusion process of Q-values, thereby maintaining information integrity during the Q-value transformation

process. A secondary benefit is that it improves the stability of policy convergence during the early stages of training, enhancing the model's robustness and environmental adaptability. Finally, it standardizes the compensation vector $\hat{\Delta}_i$ using LayerNorm to prevent overcompensation of the compensation vector, thereby enhancing the robustness of the training process.

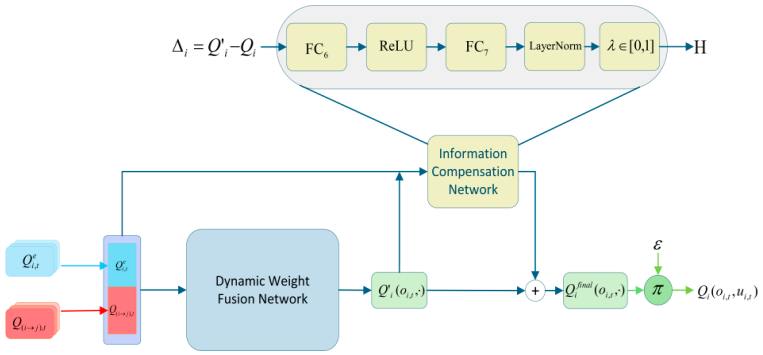


Figure 3. ICN structure diagram.

3. Comprehensive Structural Comparison Analysis

In the context of MARL, agents frequently encounter the challenge of modeling high-dimensional, highly variable actions in complex and dynamic collaborative scenarios. Addressing the challenge of modeling heterogeneous actions in high-dynamic environments has become a core difficulty in such scenarios [32,33]. To address these challenges, DDWCN employs a structural splitting approach, processing attack actions and non-attack actions in two separate streams, and introduces DWFN and ICN to significantly enhance the strategy's responsiveness to critical actions and the robustness of information representation. The following diagram in Figure 4 illustrates the overall structure of DDWCN.

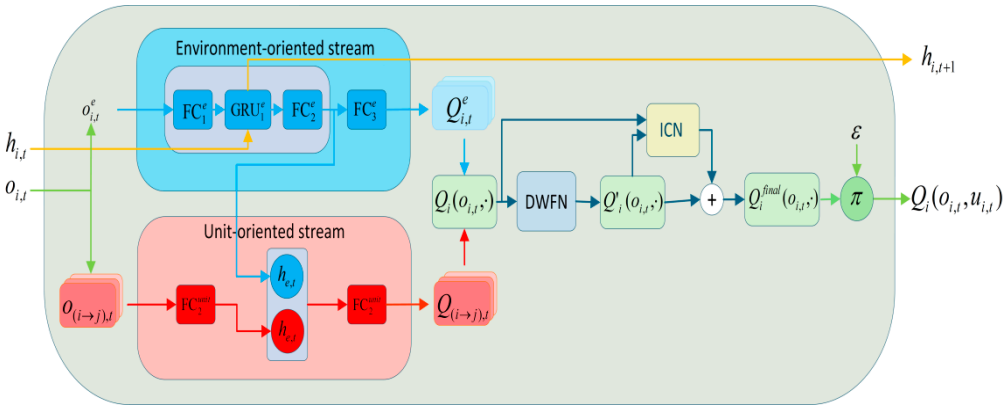


Figure 4. DDWCN structure diagram.

According to the DDWCN structural diagram presented above, a thorough analysis of the DDWCN algorithm can be conducted, and a comparison with traditional MARL algorithms can be made.

3.1. Action Modeling Path

It transitions from indiscriminate processing to divergent action modeling processing. Traditional MARL algorithms often use a unified processing method for action modeling, meaning all actions are incorporated into a unified policy network regardless of the task

or action type. The joint action value function in traditional MARL algorithms is usually represented as follows:

$$Q(s,a) = f(s,a) \tag{10}$$

$f(\cdot)$ represents the single neural network channel of the traditional MARL algorithm. This algorithm treats each action as a homogeneous element, disregarding the semantic differences and variability between different actions. While this uniform action modeling approach offers a certain degree of simplicity in action information processing, it often struggles to adapt to the semantic heterogeneity and variability of different action types in complex and dynamic collaborative scenarios. This limitation restricts the agent's ability to express strategies at a fine-grained level.

In order to overcome the action modeling bottleneck previously mentioned, the DDWCN algorithm innovates by explicitly splitting the action modeling path. DDWCN divides the processing into action semantic categories and constructs a dual-channel action modeling structure. This allows different types of action value functions to learn their specific semantic features. In terms of structure, this method effectively avoids information confusion and strategy conflicts. The action modeling method of the DDWCN algorithm is expressed as follows:

$$Q^{e}_{i,t} = f_1(o^{e}_{i,t}, h_{i,t}) \tag{11}$$

$$Q^{e}_{(i\to j),t} = f_2(h_{e,t}, h_{(i\to j),t}) \tag{12}$$

the former corresponds to the Q-value for non-aggressive actions, whose dimensionality remains unchanged regardless of the agent's death. In contrast, the latter represents the Q-value for aggressive actions, whose dimensionality varies dynamically with the presence or absence of other agents. The latter is the Q-value for aggressive actions, whose numerical dimension undergoes dynamic changes upon the death of the intelligent agent. $f_1(\cdot)$ and $f_2(\cdot)$ represent two different neural network channels of the Dual-stream action modeling network. The design of the flow modeling mechanism enables the intelligent agent to select different types of actions for dynamic information processing, thereby improving the adaptability and strategy accuracy of action processing.

3.2. Action Weight Fusion Processing

The evolution of processing methods has progressed from static to dynamic fusion scheduling. In conventional Markov random field (MARL) algorithms, there is an absence of dynamic differentiation between the two categories of action Q values. Conversely, the generated Q values are merely concatenated and processed statically. The Q-value processing method of the traditional MARL algorithm is as follows:

$$Q_i = Q^{e}_{i,t} + Q_{(i\to j),t} \tag{13}$$

In contrast, DDWCN adopts a dual-channel Q-value modeling structure that explicitly differentiates between distinct types of actions and assigns them independent Q-value representation channels. Furthermore, the framework incorporates a perception-driven DWFN, which replaces the static Q-value processing paradigm in traditional MARL algorithms with a dynamic, weight-aware fusion approach. DDWCN employs a neural network to perform nonlinear transformations on the Q-values of different action types and outputs a weight-based gating factor g_t, which is used to distinguish the relative importance of attack and non-attack actions under the current context. Using formula (5), it generates Q-values via dynamic weight fusion.

This dynamic perception method, based on action semantic weights, enables intelligent agents to make real-time adjustments based on the current environmental state and enemy-

friendly information. Consequently, this improves the precision of strategy control and the accuracy of behavior selection, thereby enhancing the decision-making capabilities of multi-agent systems.

3.3. Information Compensation Processing

This progression encompasses a transition from implicit compression to explicit compensation. In conventional MARL algorithms, Q values are managed in a relatively straightforward manner, necessitating only static concatenation processing without regard for information loss during the Q value transformation process. However, in the DDWCN algorithm, different types of Q-values need to be dynamically weighted and fused, and during the fusion process, some information loss may occur. In complex and dynamic collaborative scenarios, the loss of information can easily lead to decision-making errors by intelligent agents, thereby affecting the robustness of the strategy. In order to address the aforementioned issues, DDWCN introduces ICN to process Q-value information during the fusion process and uses formula (9) to compensate for the fusion Q-value information.

The design of the ICN allows the system to structurally compensate for critical Q-value information that may be compressed or weakened during the fusion process, based on the dynamic integration of the main Q-value stream. This effectively prevents information loss during policy learning and further enhances the accuracy and robustness of intelligent agent decision-making.

4. Simulation Experiments

In order to validate the performance of DDWCN in practical applications, it is necessary to select a classic elastic collaboration scenario and conduct comparative tests with other intelligent agent algorithms. Among these, StarCraft II is a classic elastic collaboration scenario, and its training environment meets our training and testing requirements. StarCraft II necessitates players to command a team and engage in combat with opponents to achieve victory in the battle. During combat, agents may face death, which can lead to changes in agent numbers. These changes, in turn, can affect agent actions, states, and other combat-related information. Consequently, we have selected it as our training environment to compare and test the performance of various multi-agent algorithms.

4.1. Experimental Setup

The training scenario was configured as a competitive setup between two teams. Our team was controlled using a set of representative MARL algorithms selected by us, including DDWCN, VDN [7], QMIX [8], and QTRAN [9]. The opposing team was operated using the built-in standard AI system of StarCraft II to ensure stable gameplay and consistent adversarial conditions, thereby providing a reliable experimental foundation for the evaluation of algorithm performance.

To minimize the influence of external interference and ensure that the multi-agent system operates within a highly standardized and reproducible experimental environment, this study adopts a control variable method to rigorously regulate the hyperparameters involved in the experimental process. This approach effectively isolates potential confounding factors, thereby enhancing the scientific validity and comparability of the experimental results. The specific hyperparameter configurations are detailed in Table 1.

At each discrete interval, the agent carries out the associated action in a decentralized manner and acquires the relevant reward value through interactions with the environment and other agents. When an agent inflicts damage on an enemy agent, the system rewards the agent with a positive incentive. In the event that an agent successfully eliminates another agent, the SMAC system responds by providing a greater positive reward value.

Furthermore, Double Q-learning [34] is introduced to address the issue of overestimated Q-values, with a target network established for interval updates.

Table 1. Experimental hyperparameter settings.

Settings	Name	Value
Training Settings	Size of Replay buffer D	5000 episodes
	Batch size b	32 episodes
	Testing interval	10,000 timesteps
	Target update interval	200 timesteps
	Maximum timesteps	2 million timesteps
	Exploration rate ε	1.0 to 0.05
	Discount factor γ	0.99
Network Settings	Hyper network unit	64
	GRU layer unit	64
	Optimizer	RMSProp
	RMSProp α_R	0.99
	RMSProp ε_R	0.00001
	Learning rate α	0.0005

In this study, the training process is organized in units of 10,000 environment interaction timesteps, which we refer to as the testing interval. After each such interval, a testing phase is conducted under the same environmental conditions as the training phase, except that the agents' parameters are fixed and no further learning occurs. During testing, agents act using a purely greedy policy, ensuring that the performance evaluation accurately reflects the policies learned up to that point without additional updates. In our experiments, the primary evaluation metric is the win rate. Win rate is defined as the ratio of the number of games won by our agents to the total number of evaluation games, expressed as a percentage. "Optimal performance" refers to the highest average win rate achieved during the entire training process. The training process employs GPU acceleration to expedite computation and improve overall efficiency.

4.2. Ablation Experiment

The objective of this section is to systematically evaluate the influence of each key module of DDWCN in the Q-value modeling process. To this end, two ablation experiments have been designed and conducted to examine the impact of DDWCN submodules and λ value on the overall DDWCN model. The primary focus of this study was to conduct ablation experiments on the following structures and their parameters: In the first experiment, the DAMN module is retained while DWFN and ICN are removed to assess their contributions to overall model performance. The second experiment focused on alterations in the information compensation intensity λ value within ICN. In order to eliminate the potential impact of other interfering factors, all ablation experiments were conducted within the 8m map environment of StarCraft II. All other model configurations were maintained at a consistent state with those of the primary DDWCN model, with only the target modules and parameters undergoing adjustment.

4.2.1. Module Ablation Experiment

In this set of experiments, ICNs ability to compensate for missing information is contingent on DWFN, indicating that ICN cannot exist independently of DWFN. Additionally, the construction of the DDWCN algorithm is predicated on DAMN, rendering its existence contingent on DAMN. Consequently, three training models were established to conduct ablation experiments on each module. First, the overall DDWCN model, which retains the

DAMN, DWFN, and ICN modules in its original DDWCN model; second, the DDWCN-NO model, which retains only the DAMN and DWFN modules and does not retain the ICN module; and third, the DAMN model, which does not retain the DWFN and ICN modules. A comparative analysis of these three models enables the discernment of the significance of each module in relation to the overall DDWCN model.

Figure 5 presents a comparison of the task win rate over time for the DDWCN, DDWCN-NO, and DAMN models in the classic StarCraft II 8m environment. An analysis of the win rate curves reveals that DDWCN exhibits optimal performance, demonstrating a substantial performance advantage from the initial stages of training. The win rate curve demonstrates a swift increase in the short term and maintains commendable robustness, showcasing excellent speed and stability. In contrast, the DAMN model, which is deficient in the DWFN and ICN modules, also demonstrates a rapid increase in the early stages; however, it exhibits significant volatility and unstable strategy fluctuations. A comparison of DDWCN-NO with the previously mentioned two models reveals a conspicuous performance deficit. The module under scrutiny is the ICN module, which is removed in the process. Only the combination of DAMN and DWFN modules is retained, resulting in excessive changes in Q-values before and after fusion daily only when the training process reached approximately one-fifth of its duration. Furthermore, the overall convergence speed was found to be insufficient, reflecting limitations in both the rapid adjustment and efficient processing of weights. This, in turn, leads to unstable strategy outputs. During the early stages of training, the win rate of DDWCN-NO remains at a low level and only gradually increases in the mid stages, eventually achieving a moderate degree of stability and robustness.

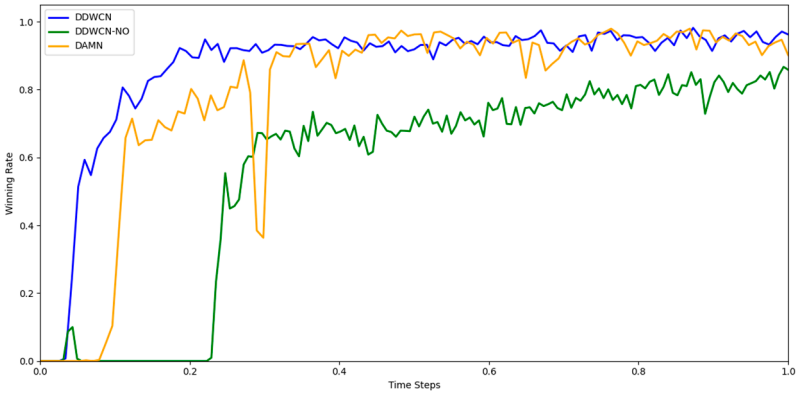


Figure 5. Comparison of DDWCN, DDWCN-NO, and DAMN experiments.

Overall, all three models achieve policy convergence and demonstrate promising learning performance. However, the comparison between DDWCN and DAMN indicates that integrating DWFN and ICN into DAMN-based Q-value modeling is essential for ensuring adaptability and robustness in elastic collaboration scenarios. Moreover, the analysis of DDWCN-NO versus DAMN reveals that removing the ICN module—while retaining only DWFN and DAMN—can result in uncompensated information loss; thereby undermining policy stability.

The overall structural comparison results demonstrate that DDWCN exhibits superior policy learning efficiency and robustness compared to DDWCN-NO and DAMN in the 8m scenario. Moreover, each submodule plays an irreplaceable role and constitutes an indispensable component of the overall algorithm.

4.2.2. Parameter Ablation Experiment

In order to minimize interference, the parameter ablation experiment was still conducted in the 8m environment of StarCraft II. To this end, four groups of λ value parameter models for the ICN were configured for the purpose of comparative experimentation. The primary objective of these experiments was to investigate the impact of varying λ value compensation strengths on the overall DDWCN model. The λ values were set to 0.1, 0.2, 0.4, and 1, respectively, to perform information compensation on the DWFN module. By comparing the training performance of models with different λ values, we can clearly identify the impact of varying compensation strengths on the overall DDWCN model.

Figure 6 illustrates the trend of the win rate of our intelligent agent over time under different λ values using the DDWCN algorithm. The objective of this figure is to evaluate the impact of different λ values on the overall performance of the DDWCN strategy. According to the overall trend, when λ is set to 0.2, the DDWCN-0.2 model rapidly converges in the early stages and exhibits a relatively stable strategy. In the event that λ is set to 0.4, the training performance of the DDWCN-0.4 model is marginally suboptimal. Nevertheless, its absolute win rate can still be maintained at approximately 90% during the final convergence phase. The model's robustness is deficient, exhibiting substantial variations during the initial training phase and unstable strategy learning. When λ is set to 0.1, the DDWCN-0.1 model is unable to effectively compensate for information differences between Q-values before and after dynamic weight fusion. This results in significant fluctuations in the win rate curve and poor decision convergence. The strategy learning process demonstrates a deficiency in robustness. When λ is set to 1, the DDWCN-1.0 model displays information overcompensation. Despite the win rate curve demonstrating a consistent upward trend during the training phase, the rate of increase is gradual, and the model exhibits suboptimal training performance.

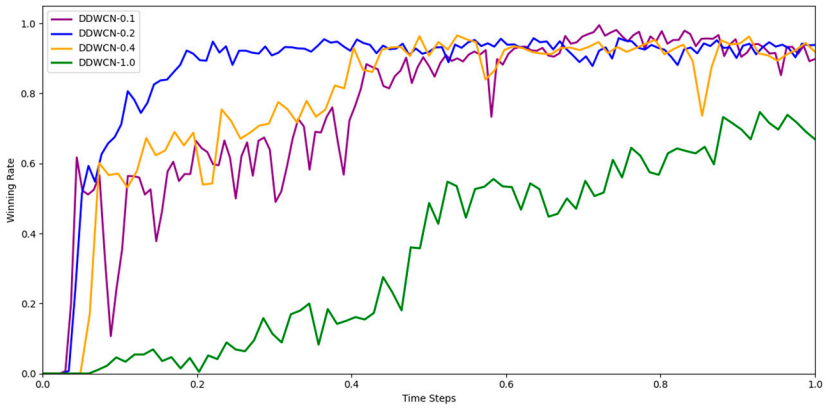


Figure 6. Comparison of DDWCN experiments between different λ values.

In summary, the λ value of ICN plays a crucial role in the overall DDWCN model. Selecting an appropriate λ value enhances the fine-grained representation of Q-values, thereby improving the convergence speed and stability of the strategy. For example, when λ is set to 0.2, the DDWCN model performs optimally. In contrast, an excessively high or low λ compensation ratio may disrupt the Q-value representation, leading to structural imbalance or inadequate compensation, which ultimately reduces the performance of multi-agent systems in highly dynamic, elastic collaboration scenarios. For instance, when λ is set to 0.1 or 1, the model exhibits suboptimal performance.

4.3. Comparative Experiments

The following section is dedicated to the systematic evaluation of the training performance of DDWCN in comparison to other MARL algorithms in a variety of environments.

To this end, three comparative experiments have been designed and conducted to assess the training effectiveness of DDWCN in various heterogeneous collaborative scenarios. In order to eliminate the potential impact of extraneous variables, all comparative experiments are conducted within the 8m, 2s3z, and 3s5z map environments of StarCraft II. Among these, the 8m scenario represents a homogeneous collaborative environment in which all agents are of the same type and exist in large numbers. The 2s3z scenario represents a simple heterogeneous elastic collaboration environment, where agents are of different types but relatively small in number, and the cooperative structure is less complex. The 3s5z scenario corresponds to a complex heterogeneous elastic collaboration environment, characterized by a larger number of agents of multiple types, more intricate interactions, and higher coordination difficulty. During the training phase, we ensure consistency in map configurations and adjust the MARL algorithms exclusively, thereby minimizing the influence of extraneous factors. In addition to performance in individual scenarios, the experiments are also designed to evaluate the generalization ability of each algorithm across heterogeneous tasks with varying agent compositions and complexity levels.

4.3.1. 8m Scenario

In order to validate the performance of DDWCN in homogeneous elastic collaboration scenarios, four MARL methods were selected for comparison in an 8m scenario: DDWCN, VDN, QMIX, and QTRAN. The comprehensive performance of the DDWCN algorithm and other MARL algorithms was evaluated in terms of strategy convergence speed, stability, and final performance.

As demonstrated in Figure 7, DDWCN exhibits commendable performance on the 8m map. It demonstrates effective convergence during the initial training phase, as evidenced by the rapid and stable convergence of the win rate curve. This ultimately leads to a win rate exceeding 90%, indicative of its high performance. VDN achieves a certain degree of convergence during the strategy learning process and exhibits an upward trend in win rate during the middle stage of training. However, its convergence speed is relatively slower, and the final stable win rate remains around 80%, which is lower than that of DDWCN, reflecting a performance gap in overall effectiveness. Conversely, QMIX demonstrates substantial volatility during strategy learning, manifesting as pronounced variations in the win rate curve during convergence. This suggests a deficiency in robustness. QTRAN demonstrates suboptimal performance in comparison to other algorithms, manifesting considerable variations from the initial training stages and exhibiting a gradual convergence rate, necessitating enhanced performance.

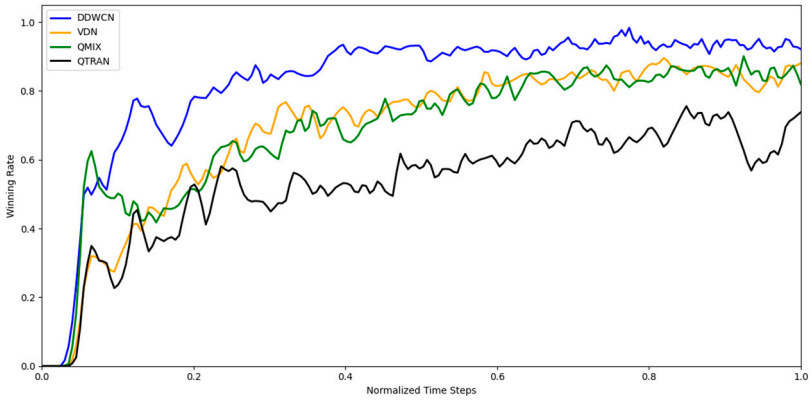


Figure 7. Comparison of 8m experimental algorithms.

The DDWCN algorithm exhibits favorable convergence and robustness in an 8m homogeneous elastic collaboration scenario, accompanied by commendable generalization

capabilities. Its performance surpasses that of other prevalent MARL algorithms. The experimental results presented herein corroborate the theoretical and practical advantages of the structure proposed by DDWCN in homogeneous elastic collaboration scenarios.

4.3.2. 2s3z Scenario

In order to verify the performance of DDWCN in heterogeneous elastic collaboration scenarios, four MARL algorithms were selected for comparison: DDWCN, VDN, QMIX, and QTRAN. These algorithms were evaluated in the 2s3z scenario to ascertain their performance.

As demonstrated in Figure 8, DDWCN demonstrates the optimal overall performance. The win rate curve demonstrates marked improvement in the early stages, attains rapid convergence, and maintains adequate robustness during the subsequent strategy convergence process, thereby preserving the win rate at the highest possible level. The VDN algorithm exhibits notable proficiency in strategy learning, as evidenced by its win rate curve, which shows an initial fluctuation followed by a gradual upward trend during the training phase. However, the rate of increase in its win rate curve is relatively slow, and it lacks the ability to converge quickly. A comparison of the QMIX algorithm to the VDN algorithm reveals that QMIX demonstrates a heightened degree of sensitivity in convergence and a faster response time. However, it is important to note that QMIX also exhibits substantial volatility and lacks robust stability. In comparison, QTRAN demonstrates substandard performance, exhibiting consistently low win rates. The system's rapidity and stability in strategy convergence are notable, yet it appears to face challenges in adapting to complex, elastic, heterogeneous collaborative scenarios.

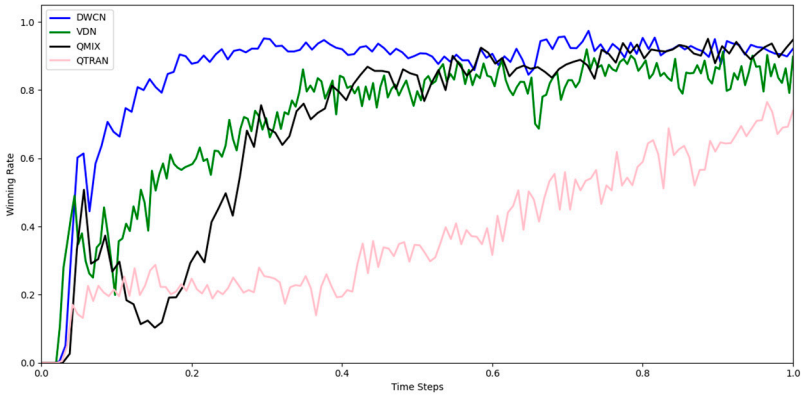


Figure 8. Comparison of 2s3z experimental algorithms.

In summary, DDWCN has been demonstrated to be a superior algorithm in comparison to other MARL algorithms. It is particularly well-suited to heterogeneous elastic collaboration scenarios and has been shown to exhibit significant advantages in convergence speed, stability, and final win rate. These findings serve to verify its excellent performance.

4.3.3. 3s5z Scenario

In order to provide further validation of the performance of DDWCN in more complex heterogeneous elastic collaboration scenarios, DDWCN, VDN, QMIX, and QTRAN—four MARL algorithms—were selected for comparison in the 3s5z scenario to verify their performance.

As illustrated in Figure 9, the win rate comparison curves for the four algorithms—DDWCN; VDN; QMIX; and QTRAN—are presented; enabling a direct performance comparison among them. A thorough examination of the experimental results reveals that DDWCN demonstrates the most optimal performance, exhibiting a swift escalation in

its win rate during the initial training stages, followed by a consistent and sustained enhancement in its performance metrics. This observation signifies that DDWCN consistently attains the highest win rate and manifests the most advanced strategy learning capabilities. QMIX ranks second in overall strategy learning performance, and its steadily rising win rate during the early training phase indicates a certain ability for information integration and stable policy execution. VDN demonstrates suboptimal overall performance, characterized by slow convergence and poor robustness, suggesting insufficient modeling capability when dealing with highly complex collaborative tasks. QTRAN demonstrates the poorest performance, with its win rate barely exceeding 10% throughout the training process. This suggests the presence of convergence difficulties or strategy degradation.

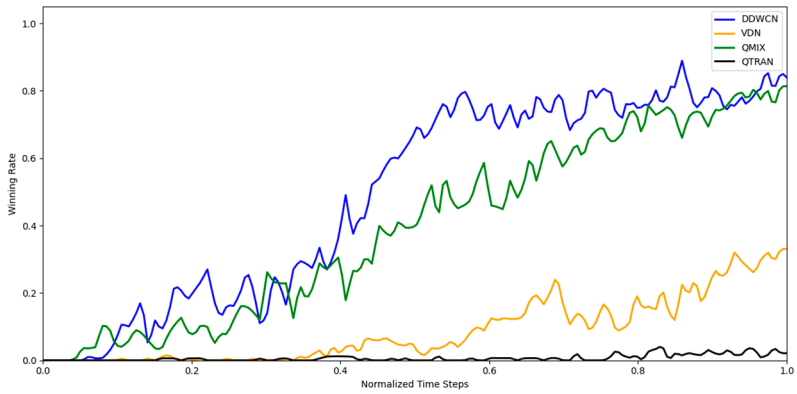


Figure 9. Comparison of 3s5z experimental algorithms.

In summary, DDWCN demonstrates significant performance advantages over existing mainstream algorithms in the 3s5z scenario. This outcome substantiates the algorithmic efficacy of DDWCN in intricate, heterogeneous elastic collaboration scenarios, particularly with regard to augmenting the collaborative decision-making capabilities and robustness of multi-agent systems. These findings further demonstrate that DDWCN is capable of maintaining strong generalization performance even in the most challenging heterogeneous elastic collaboration environments.

5. Conclusions

The present paper is concerned with the fundamental challenges encountered by MARL in elastic collaboration scenarios, including alterations in the number of agents and weighted heterogeneous actions. The paper proposes a DDWCN algorithm. The algorithm has been innovatively constructed based on three modules, which are designed to address the dynamic changes in elastic collaboration scenarios. The first module is the DAMN module, which separately processes attack actions and non-attack actions. It adapts to the dynamic changes in action values in elastic collaboration scenarios. The second module is the DWFN module, which can adaptively adjust the influence levels of attack actions and non-attack actions based on the state semantics of heterogeneous actions. The third module is the ICN module, which uses residual signals as guidance to capture the difference values before and after dynamic weight fusion. This enables adaptive compensation of action modeling information. The construction of these three modules enhances DDWCNs ability to distinguish critical actions, thereby improving the speed and robustness of agent strategy learning from the action semantic level.

A large number of experimental results demonstrate that DDWCN exhibits outstanding performance in highly dynamic elastic collaboration scenarios, achieving high win rates and good stability during the strategy convergence phase, thereby demonstrating the effectiveness of DDWCN in the strategy learning process. At the same time, the design of

ablation experiments further verifies the influence of each submodule, further supporting the scientific nature and necessity of the submodule structure. It is worth noting that DDWCNs dynamic adjustment strategy exhibits significant sensitivity to the value of the information compensation coefficient λ. In the event that λ exceeds acceptable limits, it may lead to instances of excessive information compensation, thereby impeding the modeling of primary actions and consequently reducing the velocity and stability of strategy convergence. In the event that λ is inadequate, it has the potential to diminish the efficacy of the ICN, resulting in a diminution of heterogeneous action information during the fusion process. This, in turn, can give rise to fluctuations in strategy learning. Consequently, future research endeavors may be expanded in the following domains:

- Adaptive parameter adjustment: To further enhance the adaptability and robustness of the DDWCN algorithm, it is possible to incorporate adaptive mechanisms from reinforcement learning to enable dynamic adjustment of the key parameter λ.
- Scenario generalization and deployment: The DDWCN algorithm may be further extended to a broader range of virtual and real-world scenarios, such as drone formations and autonomous vehicle coordination, to continually validate its practicality and cross-environment adaptability.

In summary, the DDWCN algorithm achieves separate processing of different types of actions, dynamic weight control, and information compensation at the theoretical structural level. Furthermore, the experimental results substantiate the model's capacity for adaptability and resilience, thus establishing a novel research framework for the implementation of MARL algorithms in elastic collaboration scenarios. Moreover, the observed stability of the win rate across multiple independent runs, conducted under identical settings, indicates that the coordinated design of the DAMN, DWFN, and ICN modules inherently enhances the reliability and efficiency of policy learning in dynamic collaborative environments. Beyond the tested StarCraft II environments, the proposed approach can be readily scaled to large-scale multi-agent collaboration tasks, such as multi-UAV cooperative reconnaissance, distributed sensor networks, and dynamic resource allocation in communication systems.

Author Contributions: Conceptualization, Z.M. and X.N.; methodology, Z.M. and X.N.; software, Z.M.; validation, Z.M. and X.N.; formal analysis, Z.M.; investigation, Z.M.; resources, Z.M. and X.N.; data curation, Z.M.; writing—original draft preparation, Z.M.; writing—review and editing, X.N.; supervision, X.N.; project administration, X.N.; visualization, Z.M.; theoretical guidance and critical revision, X.N.; experimental coordination and parameter adjustment, T.W. and J.L.; partial result verification and technical support, W.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Natural Science Foundation of Inner Mongolia Autonomous Region, China (Grant No. 2025LHMS06003). The project is led by Associate Professor Xitai Na from the School of Electronic Information Engineering, Inner Mongolia University, and conducted under the Cluster Intelligence Control and Information Processing Laboratory at Inner Mongolia University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors gratefully acknowledge the School of Electronic Information Engineering, Inner Mongolia University, for providing continuous academic support and high-performance computing resources, as well as the constructive suggestions, professional discussions,

and technical assistance from colleagues, peers, and support staff, all of which greatly contributed to the smooth implementation and successful completion of this study.

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

Abbreviations

The following abbreviations are used in this manuscript:

MARL	Multi-Agent Reinforcement Learning
CTDE	Centralized Training and Decentralized Execution
DDWCN	Dual-stream Dynamic Weight Compensation Network
DAMN	Dual-stream Action Modeling Network
DWFN	Dynamic Weight Fusion Network
ICN	Information Compensation Network
SMAC	StarCraft Multi-Agent Challenge

References

1. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access* **2020**, *8*, 58443–58469. [CrossRef]
2. Dinneweth, J.; Boubezoul, A.; Mandiau, R.; Espié, S. Multi-agent reinforcement learning for autonomous vehicles: A survey. *Auton. Intell. Syst.* **2022**, *2*, 27. [CrossRef]
3. Xia, B.; Mantegh, I.; Xie, W. Decentralized UAV swarm control: A multi-layered architecture for integrated flight mode management and dynamic target interception. *Drones* **2024**, *8*, 350. [CrossRef]
4. Wang, L.; Liu, G. Research on multi-robot collaborative operation in logistics and warehousing using A3C optimized YOLOv5-PPO model. *Front. Neurorobot.* **2024**, *17*, 1329589. [CrossRef]
5. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Trans. Cybern.* **2020**, *50*, 3826–3839. [CrossRef]
6. Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, USA, 27–29 June 1993; pp. 330–337.
7. Boutilier, C. Planning, learning and coordination in multiagent decision processes. In Proceedings of the TARK 1996, De Zeeuwse Stromen, The Netherlands, 11–13 March 1996; pp. 195–210.
8. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6382–6393.
9. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Graepel, T. Value-decomposition networks for cooperative multi-agent learning. *arXiv* **2017**, arXiv:1706.05296. [CrossRef]
10. 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.
11. Dugas, C.; Bengio, Y.; Bélisle, F.; Nadeau, C.; Garcia, R. Incorporating functional knowledge in neural networks. *J. Mach. Learn. Res.* **2009**, *10*, 1239–1262.
12. 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 (ICML), Long Beach, CA, USA, 9–15 June 2019; pp. 5887–5896.
13. Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; Zhang, C. Qplex: Duplex dueling multi-agent Q-learning. *arXiv* **2020**, arXiv:2008.01062. [CrossRef]
14. Rashid, T.; Farquhar, G.; Peng, B.; Whiteson, S. Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 10199–10210.
15. 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. No. 1.
16. Iqbal, S.; Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019; pp. 2961–2970.
17. Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of PPO in cooperative multi-agent games. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24611–24624.
18. Kuba, J.G.; Chen, R.; Wen, M.; Wen, Y.; Sun, F.; Wang, J.; Yang, Y. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv* **2021**, arXiv:2109.11251. [CrossRef]
19. Mahajan, A.; Rashid, T.; Samvelyan, M.; Whiteson, S. MAVEN: Multi-agent variational exploration. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 7613–7624.

20. Wang, T.; Gupta, T.; Mahajan, A.; Peng, B.; Whiteson, S.; Zhang, C. RODE: Learning roles to decompose multi-agent tasks. *arXiv* **2020**, arXiv:2010.01523. [CrossRef]
21. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
22. Peng, B.; Rashid, T.; Schroeder de Witt, C.; Kamienny, P.A.; Torr, P.; Böhmer, W.; Whiteson, S. FACMAC: Factored multi-agent centralised policy gradients. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12208–12221.
23. Yang, B.; Gao, L.; Zhou, F.; Yao, H.; Fu, Y.; Sun, Z.; Ren, H. A coordination optimization framework for multi-agent reinforcement learning based on reward redistribution and experience reutilization. *Electronics* **2025**, *14*, 2361. [CrossRef]
24. Zhao, Z.; Zhang, Y.; Wang, S.; Zhou, Y.; Zhang, R.; Chen, W. Assisted-value factorization with latent interaction in cooperate multi-agent reinforcement learning. *Mathematics* **2025**, *13*, 1429. [CrossRef]
25. Liu, B.; Liu, Q.; Stone, P.; Garg, A.; Zhu, Y.; Anandkumar, A. Coach-player multi-agent reinforcement learning for dynamic team composition. In Proceedings of the International Conference on Machine Learning (ICML), Virtual, 18–24 July 2021; pp. 6860–6870.
26. Wang, W.; Yang, T.; Liu, Y.; Hao, J.; Hao, X.; Hu, Y.; Gao, Y. From few to more: Large-scale dynamic multi-agent curriculum learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 7293–7300.
27. Tang, X.; Xu, J.; Wang, S. Transferable multi-agent reinforcement learning with dynamic participating agents. *arXiv* **2022**, arXiv:2208.02424. [CrossRef]
28. Samvelyan, M.; Rashid, T.; De Witt, C.S.; Farquhar, G.; Nardelli, N.; Rudner, T.G.; Whiteson, S. The StarCraft multi-agent challenge. *arXiv* **2019**, arXiv:1902.04043. [CrossRef]
29. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555. [CrossRef]
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
31. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450. [CrossRef]
32. Liang, J.; Miao, H.; Li, K.; Tan, J.; Wang, X.; Luo, R.; Jiang, Y. A Review of Multi-Agent Reinforcement Learning Algorithms. *Electronics* **2025**, *14*, 820. [CrossRef]
33. Oroojlooy, A.; Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *Appl. Intell.* **2023**, *53*, 13677–13722. [CrossRef]
34. Van Hasselt, H. Double Q-learning. *Adv. Neural Inf. Process. Syst.* **2010**, *23*, 2613–2621.

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.