ADAPT: Adaptive Decentralized Architecture with Perception-Aligned Training for Structural Generalization in Multi-Agent RL

Zhixiang Zhang[1,2], **Shuo Chen**[2*], **Yexin Li**[2], **Feng Wang**[1*]

[1]School of Computer Science, Wuhan University
[2]State Key Laboratory of General Artificial Intelligence, BIGAI, Beijing, China
zhangzhixiang@whu.edu.cn, chenshuo@bigai.ai, liyexin@bigai.ai, fengwang@whu.edu.cn

Abstract

Multi-agent reinforcement learning (MARL) excels in cooperative and competitive tasks, but most architectures are tied to fixed input-output sizes and require retraining when the number of perceptible or controllable objects changes. While structural generalization techniques mitigate this, they rely on centralized training, raising concerns about scalability and privacy. We propose ADAPT, the first framework to support structural generalization under a decentralized training and decentralized execution (DTDE) paradigm. Every agent adopts an object-centric view, encoding each observed object into a feature vector and aggregating them into a variable-length set representation. To enable each agent to infer task-level contexts from this dynamic input independently, we propose a dynamic-consistency loss that enforces spatio-temporal alignment between context representations and observed environmental dynamics. Agents then condition their policies on the inferred contexts to make locally aligned decisions. For zero-shot transfer, we propose FINE (Foresight INdex for multi-agEnt), a metric that considers Q-value overestimation and enables cross-policy comparison of long-term impact, facilitating effective policy transfer. Experiments show that ADAPT surpasses existing DTDE methods and outperforms CTDE baselines in zero-shot generalization.

1 Introduction

Multi-agent reinforcement learning (MARL) has become a core technique for solving complex cooperative and competitive problems such as autonomous traffic control (El-Tantawy, Abdulhai, and Abdelgawad 2013; Chu et al. 2019), drone-swarm coordination (Venturini et al. 2021; Hou et al. 2023; Shi et al. 2025), real-time strategy games (Zhou et al. 2021), and multi-robot search-and-rescue (Lei et al. 2024; Hou et al. 2023). In these domains, agents must perceive and act upon a varying set of objects—vehicles, drones, adversaries, or resources—while jointly maximizing team performance. Despite notable empirical successes, mainstream MARL architectures assume fixed-size observation and action spaces—each neural network is hard-wired to accept a preset number of input features regarding the information of perceived objects and to emit a predetermined number of actions corresponding to operations upon objects. This rigid design poses a fundamental obstacle to structural generalization. When the environment's object count differs from the one seen during training, the trained network no longer matches the input, output, or both dimensions, and must be redesigned and retrained. One might naively over-provision the network with very large input and output sizes and pad or mask unused entries, but this (i) wastes computational resources in small scenarios and (ii) still breaks down when the actual number of objects in the environment exceeds the maximum size the network was designed to handle.

Recent methods aim to endow MARL with structural generalization. Representative works include OPT (Liu et al. 2024) and UPDeT (Hu et al. 2021), which leverage Transformers to process variable-length entity sets; FlickerFusion (Koh et al. 2025), which randomly drops entities to maintain a constant input budget; REFIL (Iqbal et al. 2021), which dynamically factorizes the joint action space across agent subsets; and MATTAR (Qin et al. 2024), which alternates fixed-size training batches to enhance transferability. However, they all adopt the centralized training with decentralized execution (CTDE) paradigm, which often incurs substantial communication overhead, raises privacy concerns, and scales poorly with an increasing number of agents.

To overcome the rigidity of fixed-structure architectures and the inherent limitations of the CTDE paradigm, we aim to design an MARL framework that achieves structural generalization with the DTDE paradigm. Specifically, we seek to enable each agent to independently perceive and adapt to variable numbers of objects in its local environment, while learning representations that generalize across scenarios of different scales and complexities. Our key design is an object-centric view, where each perceived entity, such as a teammate, opponent, or environmental object, is treated as a distinct object and encoded as an individual token.

On the input side, every agent aggregates its encoded token into a variable-length set. A gated recurrent unit (GRU) then processes this set of tokens to produce a set of hidden vectors, which are subsequently aggregated by an attention module into a weighted context representation. This aggregated representation serves as the agent's task-level input, supporting decision-making under diverse and dynamic settings. To ensure that this learned representation faithfully captures the spatio-temporal dynamics of the environment, we further introduce a dynamic consistency (DC)

*Corresponding author(s).

loss. This objective minimizes the Huber distance among (i) changes in raw observations, (ii) changes in the GRU hidden state (capturing temporal dynamics), and (iii) changes in the attention-weighted context (capturing spatio-temporal dynamics). In doing so, we force the inferred context to remain aligned with the environment's evolution.

On the output side, we again exploit the object-centric view. For every manipulable object, the agent feeds the object-specific slice of the context into a lightweight action head, which evaluates a fixed set of actions applicable to each object type, assigning scores that reflect their expected utility. The agent then selects the action-object pair with the highest score, effectively determining both which object to act upon and how. Since the action head's output size depends only on the per-object action set rather than the total number of objects, our policy naturally scales to environments of varying sizes and operates without reliance on other agents' internal states or observations.

After training, generalizing the learned policies to diverse scenarios remains a significant challenge. Since each agent acquires an independent policy during training, it is often unclear which policy should be assigned to which agent in a novel testing scenario. Instead of selecting at the policy level, we propose a novel metric—Foresight INdex for multi-agEnt (FINE)—to enable fine-grained action-level selection. Specifically, at each time step, all trained policies propose an action for their respective agents, and the action associated with the highest FINE score is selected for execution. This mechanism empowers agents to adopt the most foresighted behavior, thereby facilitating more effective decision-making in previously unseen scenarios.

To summarize, this paper makes four key contributions: (1) To the best of our knowledge, we propose the first DTDE MARL architecture that supports structural generalization. (2) We introduce a self-supervised loss that aligns the dynamics of latent representations with observations, enabling agents to capture the spatio-temporal structure of the environment. (3) We introduce FINE, a novel metric that measures the foresight impact of each action, enabling action selection during zero-shot transfer. (4) We conduct comprehensive evaluations on MPE and SMAC benchmarks, where our method surpasses the strongest DTDE baseline and achieves superior zero-shot generalization compared to state-of-the-art CTDE structural generalization approaches.

2 Related Work

In this section, we review the existing works of structural generalization and different learning paradigms of MARL.

Structural Generalization In real-world MARL tasks, the number of task-related objects is often dynamic, such as variable-sized swarms, dynamic traffic lanes, etc. Completing those tasks needs policies that remain valid as the object set grows or shrinks. Conventional networks tie their input and output dimensions to a single object budget, so any change demands architectural redesign and retraining, prompting research on structural generalization: reusing one set of parameters across arbitrary object counts. Prior efforts to endow MARL with structural generalization fall into two main categories. The first category leverages Transformers (Bahdanau, Cho, and Bengio 2014) to decouple network parameters from the exact number of objects. UPDeT (Hu et al. 2021) and OPT (Liu et al. 2024) fix only the per-object feature dimension and let the token sequence grow with the object count, which makes them suitable for multi-agent scenarios where the object set can be diverse and dynamic across different tasks. FlickerFusion (Koh et al. 2025) tackles the problem from another angle: during training, it randomly drops subsets of objects so that the total input budget remains constant, forcing the policy to stay robust to missing or newly appearing objects, though performance can suffer if at test time all objects prove important. The second category acts on the action side. REFIL (Iqbal et al. 2021) repeatedly factorizes the joint action space into random, smaller agent subsets, encouraging value decomposition networks to adapt to changing team sizes without altering the network architecture, whereas MATTAR (Qin et al. 2024) alternates among fixed population levels in a curriculum-style schedule, enabling one model to amortize knowledge across different group sizes. Despite alleviating the mismatch problem between network dimensions and environment object count, all of these methods are trained with the CTDE paradigm, relying on global experiences and centralized critics. This dependence introduces heavy communication, privacy risks, and scalability limits, motivating our move toward structural generalization with the DTDE paradigm.

Learning Paradigms The learning paradigms in MARL are commonly categorized as Centralized Training and Centralized Execution (CTCE), Centralized Training and Decentralized Execution (CTDE), and Decentralized Training and Decentralized Execution (DTDE) (Yuan et al. 2023). Among these, CTCE algorithms are relatively rare, as agents in most real-world or online environments typically have access only to partial and local observations during execution. CTDE has therefore become the prevailing choice: by giving agents access to centralized information or a shared critic during learning, it simplifies credit assignment and often yields higher sample efficiency, yet still deploys purely local policies. Widely used algorithms such as VDN (Sunehag et al. 2017), QMIX (Rashid et al. 2020), COMA (Foerster et al. 2018), and MAPPO (Yu et al. 2022) exemplify this paradigm. CTDE's reliance on global experiences, however, can be prohibitive when communication bandwidth is limited or privacy regulations restrict data sharing. In such situations, researchers turn to DTDE, where both optimization and decision-making are distributed. Classical DTDE baselines include independent Q-learning (Tampuu et al. 2017) and IPPO (de Witt et al. 2020). Recent work, such as I2Q (Jiang and Lu 2022), shows that agents can achieve near-optimal cooperation by learning local Q-functions guided by an implicitly modeled joint transition. However, the DTDE literature remains limited, underscoring the need for further advances in decentralized MARL.

3 Preliminaries

In this section, we formally model the cooperative MARL problem as a decentralized partially observable Markov de-

cision process with objects, referred to as **object-centric Dec-POMDP**.

Object-centric Dec-POMDP is defined by the tuple $\langle \mathcal{E}, \mathcal{T}, \mathcal{S}, \mathcal{U}, P, r, \mathcal{O}, \gamma \rangle$, where $\mathcal{E}$ denotes the set of *objects* in the environment. Each object $e \in \mathcal{E}$ is associated with a state s^e, and the global state of the environment is given by $s = \{s^e \mid e \in \mathcal{E}\} \in \mathcal{S}$. The object set $\mathcal{E}$ includes both *agents* and *non-agent entities*, such as landmarks, uncontrollable enemies, manipulable tools, etc., allowing a unified representation of all environment elements. At each time step t, agent k receives an individual partial observation $\mathbf{O}_t^k \in \mathcal{O}$ and selects an action $u_t^k \in \mathcal{U}$, together forming the joint action $\mathbf{u}_t = (u_t^1, \ldots, u_t^n)$. After executing $\mathbf{u}_t$, the environment transitions from state s_t to s_{t+1} according to the state transition function $P(s_{t+1} \mid s_t, \mathbf{u}_t)$. All agents share a common reward function $r(s_t, \mathbf{u}_t)$. Each agent k maintains an action-observation history $\tau^k \in \mathcal{T}$ and learns an individual policy $\pi^k(u_t^k \mid \tau_t^k)$ to jointly maximize the expected discounted return $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$, where $\gamma \in [0, 1)$ is the discount factor. We denote by $Q_t^k(\tau_t^k, u_t^k)$ the individual action-value function of agent k at time step t, representing the expected return given the agent's current history and action.

To train the individual action-value function Q^k, a common approach is to apply independent Q-learning, where each agent k updates Q^k based solely on its local experiences $(\mathbf{O}_t^k, u_t^k, r_t, \mathbf{O}_{t+1}^k)$. However, in decentralized multi-agent settings, the environment dynamics from each agent's perspective are non-stationary due to the simultaneous policy updates of other agents, which hinders convergence and degrades learning performance.

To address the non-stationarity in decentralized learning, we follow the ideal transition modeling approach introduced in I2Q (Jiang and Lu 2022). Each agent k independently learns the following components: (1) $Q^k(\tau_t^k, u_t^k)$: the individual action-value function estimating the expected return of taking action u_t^k under history τ_t^k; (2) $Q_{\text{ss}}^k(\tau_t^k, \tau_{t+1}^k)$: a transition value function measuring the expected return of transitioning from history τ_t^k to τ_{t+1}^k; (3) $V^k(\tau_t^k)$: the value function estimating the expected return starting from history τ_t^k. The training minimizes three mean squared error (MSE) losses:

$$\begin{aligned}
&\text{MSE}[V^k(\tau_t^k), \overline{Q}_{\text{ss}}^k(\tau_t^k, \tau_{t+1}^k)] \ \rightarrow \text{update } V^k, \\
&\text{MSE}[Q_{\text{ss}}^k(\tau_t^k, \tau_{t+1}^k), \overline{V}^k(\tau_{t+1}^k)] \rightarrow \text{update } Q_{\text{ss}}^k, \\
&\text{MSE}[Q^k(\tau_t^k, u_t^k), \lambda \overline{V}^k(\tau_{t+1}^k) + (1-\lambda)\overline{Q}^k(\tau_t^k, u_t^k)] \\
&\qquad\qquad \rightarrow \text{update } Q^k,
\end{aligned}$$

where $\overline{V}^k$, $\overline{Q}^k$, and $\overline{Q}_{\text{ss}}^k$ are target networks, and $\lambda \in [0, 1]$ balances the influence of $\overline{V}^k$ and $\overline{Q}^k$. According to (Jiang and Lu 2022), Q_{ss}^k estimates the long-term return of transitioning from state s to s' under the optimal joint policy, which decouples the Q-value learning from the non-stationary transitions caused by other agents, improving the learning stability. Moreover, all learning components take local action-observation history τ^k as input, which facilitates fully decentralized policy learning.

4 Method

We begin by introducing the notations and input representations used in our object-centric formulation. We then describe the core architecture and training losses of our proposed **ADAPT** framework. Finally, we present the action selection strategy designed for zero-shot generalization to environments with varying numbers of objects.

4.1 Object-Centric Notations

In this paper, we denote the set of environmental observations perceived by all N agents at time step t as $\mathbf{O}_t = \{\mathbf{O}_t^1, \mathbf{O}_t^2, \ldots, \mathbf{O}_t^N\}$, where $\mathbf{O}_t^k \in \mathbb{R}^{B \times E \times D_O}$ is the observation matrix of agent k. Here, B is the batch size, E is the number of observed objects, and D_O is the feature dimension of each object. The sequence of observations across T time steps for each agent is denoted as $\mathbf{O}^1, \mathbf{O}^2, \ldots, \mathbf{O}^N \in \mathbb{R}^{B \times T \times E \times D_O}$. We further denote $\mathbf{H}_t$ and $\mathbf{A}_t$ as the sets of GRU's outputs and attention module's outputs for all agents at time step t, respectively. Similarly, $\mathbf{M}_t \in \{0, 1\}^B$ represents a set of binary masks indicating whether each trajectory in the batch of experiences has been padded at time step t due to episode termination. These variables have the same agent-wise structure as $\mathbf{O}_t$. Specifically, $\mathbf{H}^1, \mathbf{H}^2, \ldots, \mathbf{H}^N \in \mathbb{R}^{B \times T \times E \times D_H}$ and $\mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^N \in \mathbb{R}^{B \times T \times E \times D_A}$, where D_H and D_A denote the embedding dimensions of the GRU and attention outputs, respectively. The corresponding mask sequences are denoted by $\mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^N \in \{0, 1\}^{B \times T}$, where 0 indicates a padded time step and 1 indicates a valid one.

4.2 ADAPT

As illustrated in Figure 1, the core architecture of the Q-network in **ADAPT** consists of three linear layers, a GRU module, and a multi-head self-attention mechanism. Under the DTDE paradigm, each agent k maintains a complete and independent Q-network Q^k for both learning and execution. At each time step t, the agent receives a local observation set $\mathbf{O}_t^k$ describing its perception of surrounding objects. These observations are first processed by the Linear-1 layer to obtain initial object-wise embeddings. The embeddings, together with the agent's hidden state $\mathbf{H}_{t-1}^k$ from the previous time step, are then passed into a GRU module to perform temporal aggregation over time.

The GRU outputs are subsequently processed by a multi-head self-attention module, enabling spatial aggregation across the observed objects. This allows the agent to assign differentiated attention weights to different objects, capturing spatial relevance and inter-object relationships. The resulting attention-enhanced representation $\mathbf{A}_t^k \in \mathbb{R}^{B \times E \times D_A}$ serves as the input to two parallel branches: Linear-2 and Linear-3. In the first branch, $\mathbf{A}_t^k$ is averaged along the object dimension E to produce a compact context vector, which is passed through Linear-2 to generate action scores $Q_{t,\text{action}\rightarrow\text{self}}^k$ for self-manipulation, e.g., movement actions. In the second branch, $\mathbf{A}_t^k$ is split along the same dimension to isolate object-specific representations, each of which is passed through Linear-3 to compute action scores $Q_{t,\text{action}\rightarrow\text{objects}}^k$ for interactions with individual objects, e.g.,

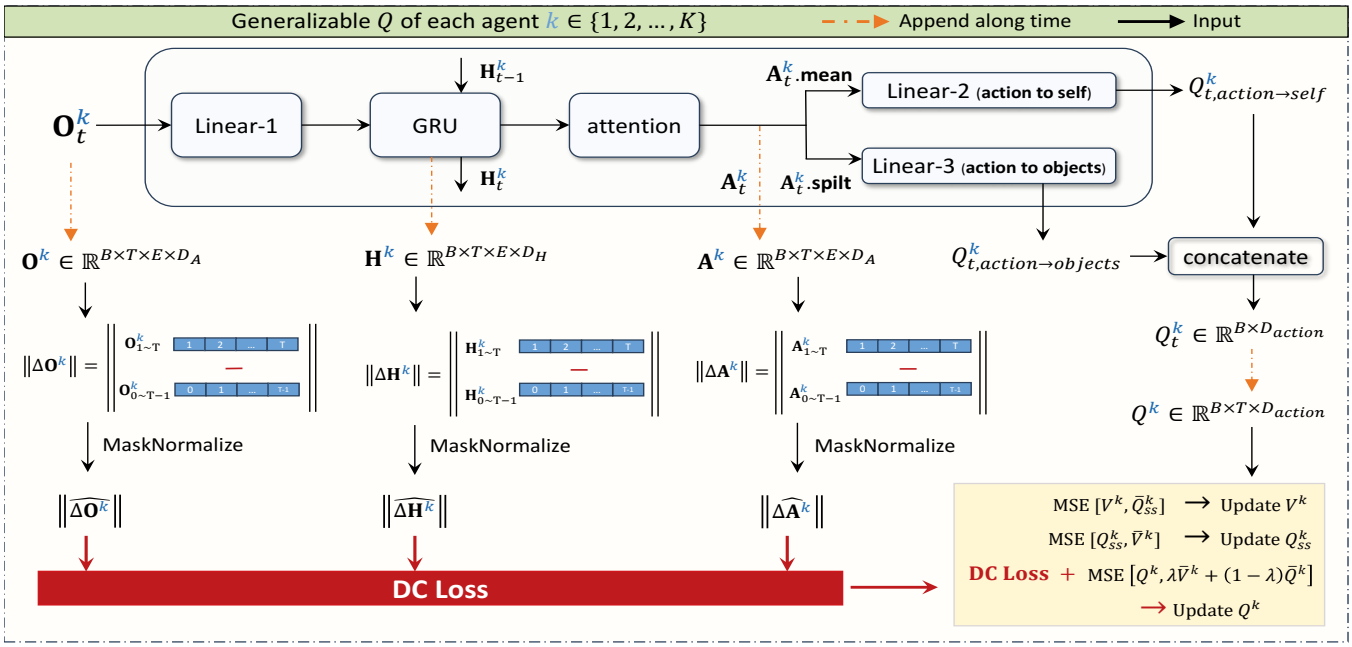


Figure 1: An illustration of the proposed **ADAPT** architecture for Q-learning in each agent k.

attacking target. These two components are then concatenated to form the final Q-value vector $Q_t^k \in \mathbb{R}^{B\times D_{\text{action}}}$, which evaluates the full action space composed of both self-directed and object-directed behaviors.

4.3 Dynamic Consistency Loss

The architecture of **ADAPT** enables each agent to perform dual-level aggregation: temporal aggregation via the GRU and spatial aggregation via the self-attention mechanism. This design allows the Q-network to integrate both historical context and current environmental information, resulting in more accurate Q-value estimation. To further enhance representation quality and consistency, we introduce a *dynamic consistency loss (DC Loss)*. At each time step t, the agent retains its raw object-centric observations, GRU outputs, and attention outputs, denoted as $\mathbf{O}^k$, $\mathbf{H}^k$, and $\mathbf{A}^k$, respectively. The system then computes the changes in these quantities between consecutive time steps, capturing how the agent's perception and internal representations evolve over time.

The DC Loss is designed to align these temporal changes: it penalizes mismatches between the evolution of raw observations and the corresponding changes in the GRU and attention outputs. By enforcing spatio-temporal consistency in the learned representations, this loss encourages the network to maintain coherent and meaningful internal dynamics across time. Together with the MSE-based updates for V^k, Q_{ss}^k, and Q^k, the DC Loss jointly guides the training of the **ADAPT** architecture, resulting in improved generalization and policy robustness. We provide the detailed formulation of the DC Loss below. We begin by introducing the **Huber loss function** (Huber 1964), which is used in the formulation of the dynamic consistency loss. It is defined as:

$$\text{Huber}(x, y) = \begin{cases} \frac{1}{2}(x-y)^2 & \text{if } |x-y| \le \delta \\ \delta\left(|x-y| - \frac{\delta}{2}\right) & \text{otherwise} \end{cases} \tag{1}$$

where δ is a threshold parameter, typically set to 1. The Huber loss behaves like mean squared error (MSE) for small differences ($|x-y| \le \delta$) and like mean absolute error (MAE) for large differences. This combination ensures smooth optimization for inliers while limiting the influence of outliers, enhancing training stability. We now define the **DC Loss**.

Definition 1. For each agent k, given its object-centric observation sequence $\mathbf{O}^k$, GRU outputs $\mathbf{H}^k$, attention outputs $\mathbf{A}^k$, and mask sequence $\mathbf{M}^k$, the DC Loss is defined as:

$$\mathcal{L}_{\text{DC}} = \xi \cdot \left[\frac{1}{\sum \mathbf{M}_{\text{valid}}^k} \sum \text{Huber}\left(\|\Delta\hat{\mathbf{O}}^k\|, \|\Delta\hat{\mathbf{H}}^k\|\right) + \frac{1}{\sum \mathbf{M}_{\text{valid}}^k} \sum \text{Huber}\left(\|\Delta\hat{\mathbf{O}}^k\|, \|\Delta\hat{\mathbf{A}}^k\|\right) \right] \tag{2}$$

where ξ is a scaling factor controlling the influence of DC Loss. The required components are computed as follows:

$$\Delta x_t = x_{t+1} - x_t, \quad x \in \{\mathbf{O}^k, \mathbf{H}^k, \mathbf{A}^k\} \tag{3}$$

$$\mathbf{M}_{\text{valid},t}^k = \mathbf{M}_t^k \cdot \mathbf{M}_{t+1}^k \tag{4}$$

$$\|\hat{\Delta} \cdot \| = \text{MaskedNormalize}\left(\|\Delta \cdot \|, \mathbf{M}_{\text{valid}}^k\right) \tag{5}$$

Here, $\|\Delta \cdot \|$ denotes the $L2$ norm computed along the feature dimension—D_O, D_H, or D_A for observations, GRU outputs, and attention outputs respectively—and the result is averaged over the object dimension E. The DC loss encourages the internal representations to evolve consistently with

the raw observations, thus enforcing spatio-temporal coherence across the network.

4.4 Action Selection for Zero-Shot Generalization

In our framework, each agent k possesses a fully independent policy. However, when directly generalizing to new scenarios with a different number of agents, a challenge arises: which pre-trained policies from the source scenario should be assigned to the agents in the target scenario?

A straightforward solution is the *voting-based method*. This method leverages all pre-trained Q-networks from the source scenario to guide each agent's action selection in the target scenario. Specifically, for every agent in the target scenario, all source-trained Q-networks independently evaluate the agent's current observation and propose an action. If a majority of networks agree on a particular action, that action is selected. If no consensus is reached, an action is randomly chosen from among the proposed candidates. However, certain scenarios may arise in which the majority of pre-trained networks propose suboptimal action advice, thereby overshadowing the correct strategies suggested by the minority.

To evaluate the most appropriate action proposed by pre-trained policies at every timestep, we introduce a novel metric termed **FINE** (Foresight INdex for multi-agEnt). FINE quantifies the proportion of long-term return that remains unrealized, serving as a measure of a policy's foresight. By selecting the action with the highest FINE value, we can efficiently identify the most promising decision. We formally define FINE as follows.

Definition 2. At time step t for policy k, given an agent's local observation $\mathbf{O}_t^k$, the FINE metric is defined as:

$$\mathcal{F}_t^k = \frac{\max_{u_t^k} Q_t^k(\mathbf{O}_t^k, u_t^k)}{\max_{u_0^k} Q_0^k(\mathbf{O}_0^k, u_0^k)} \tag{6}$$

where Q_t^k denotes the action-value function of agent k at time t, and Q_0^k represents the initial value estimation at the beginning of the episode.

Interpretation. Based on the Bellman equation unrolled to step t, the initial Q-value can be expressed as:

$$\max Q_0^k = \sum_{m=0}^{t-1} \gamma^m r_m + \gamma^t \max Q_t^k \tag{7}$$

By rearranging, the FINE metric can be rewritten as:

$$\mathcal{F}_t^k = \frac{\max Q_t^k}{\max Q_0^k} = \frac{1}{\gamma^t}\left(1 - \frac{\sum_{m=0}^{t-1} \gamma^m r_m}{\max Q_0^k}\right) \tag{8}$$

This shows that maximizing $\mathcal{F}_t^k$ is equivalent to minimizing the proportion of accumulated past rewards in the overall return. Therefore, FINE quantifies how much value remains to be realized in the future. A high value of $\mathcal{F}_t^k$ indicates that the policy's current decision leads to substantial delayed rewards. In contrast, a low $\mathcal{F}_t^k$ implies that most of the return has already been received, which may reflect short-sighted behavior. In the zero-shot generalization, we use FINE as a selection criterion among multiple trained policies. At each time step, we compute the FINE score for each policy and choose the one with the highest value as the advising policy for action selection. This allows us to use policies that exhibit stronger foresight in the current situation.

In addition to the ADAPT and its action selection methods designed for fully decentralized settings, we propose a variant of ADAPT, referred to as **ADAPT-share**, designed for scenarios where limited communication is permitted during training. In this method, each agent still maintains a complete and independent Q-network. However, unlike the CTDE paradigm, ADAPT-share does not rely on access to global state or observations during training. Instead, agents communicate only their gradient information.

Specifically, at each training step, after computing gradients locally based on their own observations, agents either (i) share their gradients directly with one another in a peer-to-peer manner, or (ii) send them to a lightweight central server, which simply aggregates, e.g., by summation, and broadcasts the combined gradient back to all agents. Each agent then updates its own network parameters using the aggregated gradient. As a result, all agents maintain identical Q-network parameters throughout training. This unified model can then be directly deployed to any agent in a new task, enabling straightforward and effective generalization without the need for agent-specific policy assignments.

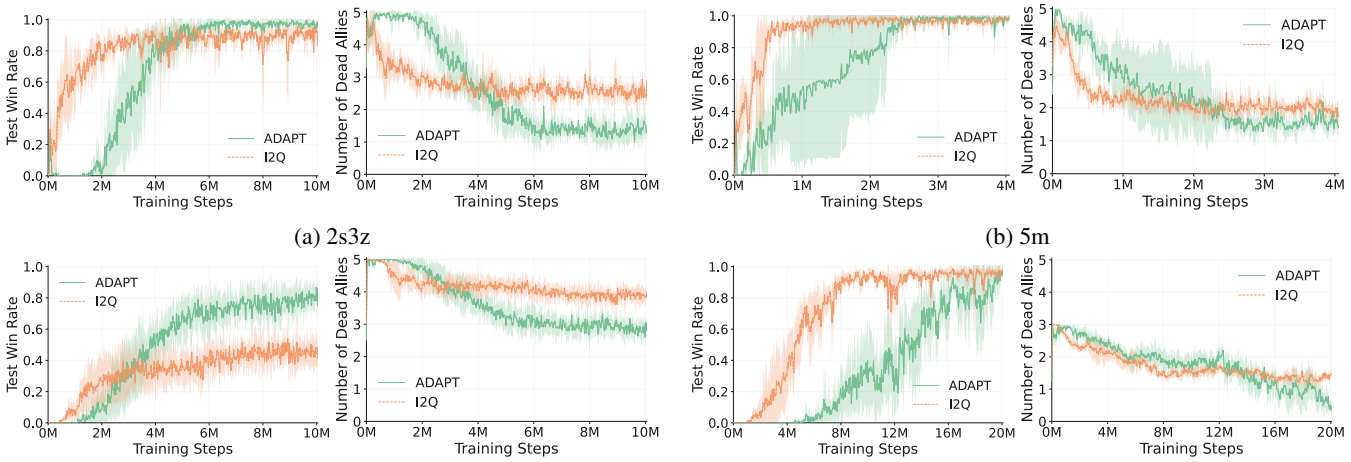
ADAPT-share offers two advantages over CTDE-based methods. First, it avoids the need to exchange raw observations, substantially reducing communication overhead and mitigating the scalability issues caused by the exponential growth of the observation space with the number of agents. Second, by not requiring access to private observations, it mitigates the privacy concern in real-world deployments where transmitting local observations may not be permitted.

5 Experiments

In this section, we evaluate ADAPT on challenging micromanagement tasks in the StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al. 2019) and the Multi Particle Environments (MPE) (Lowe et al. 2017). We compare ADAPT with I2Q (Jiang and Lu 2022) across multiple scenarios to demonstrate its superior performance when trained from scratch. Furthermore, we assess the zero-shot generalization capabilities of both ADAPT and ADAPT-share in various scenarios, comparing them against UPDeT (Hu et al. 2021) and OPT (Liu et al. 2024).

5.1 SMAC

To evaluate the effectiveness of the ADAPT series in complex cooperative environments using neural network-based implementations, we select SMAC as one of the primary training and testing platforms. We first train the models on a single map and then directly test their zero-shot generalization performance on other maps without any further training. Specifically, we use the 2s3z map (controlling 2 Stalkers and 3 Zealots against enemies with the same composition), the 5m map (controlling 5 Marines against 5 enemy Marines), the 5m_vs_6m map (controlling 5 Marines against 6 enemy Marines) and the 3s_vs_4z map (controlling 3 Stalk-



(a) 2s3z (b) 5m

(c) 5m_vs_6m (d) 3s_vs_4z

Figure 2: Performance of ADAPT vs. I2Q in maps 2s3z (10M steps), 5m (4M steps), 5m_vs_6m (10M steps), and 3s_vs_4z (20M steps) of SMAC. Evaluation metrics are the test win rate and number of dead allies.

Map	Method	Dead allies	Test win rate (%)
2s3z	I2Q	2.89 ± 0.40	83.13 ± 10.00
	ADAPT	$\mathbf{1.33} \pm 0.15$	$\mathbf{96.88} \pm 3.95$
5m	I2Q	1.87 ± 0.11	$\mathbf{99.38} \pm 1.25$
	ADAPT	$\mathbf{1.54} \pm 0.31$	98.13 ± 1.53
5m_vs_6m	I2Q	3.88 ± 0.26	40.00 ± 9.76
	ADAPT	$\mathbf{2.69} \pm 0.12$	$\mathbf{86.25} \pm 3.75$
3s_vs_4z	I2Q	1.49 ± 0.13	95.63 ± 6.12
	ADAPT	$\mathbf{0.48} \pm 0.28$	$\mathbf{96.25} \pm 4.59$

Table 1: Performance comparisons between I2Q and ADAPT when trained from scratch on **SMAC**. Both methods are trained on the map 2s3z (10M steps), 5m (4M steps), 5m_vs_6m (10M steps), and 3s_vs_4z (20M steps) with convergence observed. The number of dead allies and the test win rate are reported as the mean $\pm$ standard deviation across 5 independent runs with different random seeds.

ers against 4 enemy Zealots) to demonstrate ADAPT's superior training-from-scratch capabilities. We then evaluate the zero-shot generalization performance of ADAPT, UPDeT, and OPT in two generalization settings: both methods are trained on the 3m map and tested on the 5m map; trained on the 5m map and tested on the 7m map. We compare their performance on 5m and 7m to assess their generalization capabilities across increasing task complexity.

As shown in Table 1 and Figure 2, we train both I2Q and ADAPT to convergence on the 2s3z, 5m, 5m_vs_6m, and 3s_vs_4z maps to compare their upper-bound performance. The results show that ADAPT leads to significantly fewer allied casualties than I2Q, while achieving comparable or even higher test win rates. This suggests that ADAPT enables agents to learn more effective strategies that not only eliminate enemies efficiently but also minimize team losses.

Next, we compare the zero-shot generalization performance of ADAPT using two action selection strategies, FINE and voting-based, with UPDeT, OPT-CTDE, and OPT-DTDE in two transfer settings: from 3m to 5m and from 5m to 7m. Note that OPT is originally designed under the CTDE paradigm. For fair comparison, we also test OPT-DTDE by modifying the centralized components of OPT to comply with the DTDE paradigm. As shown in Table 2, in both transfer settings, ADAPT combined with the FINE metric achieves better generalization performance than that with the voting-based method, demonstrating the effectiveness of FINE in identifying the policy with greater foresight. Furthermore, when comparing ADAPT+FINE to UPDeT and OPT series, we observe that ADAPT+FINE consistently outperforms both baselines across tasks of increasing complexity, underscoring its superior adaptability and robustness in unseen scenarios.

5.2 MPE

To further validate the generalization effectiveness of ADAPT across a wider range of scenarios, we conducted experiments in the simple-tag environment of MPE.

In simple-tag, there are three types of objects: prey, predators, and blocks. The predators aim to catch the prey, while the prey strive to evade the predators. The environment is a square bounded by [-1,1] in both dimensions. Both prey and predators have a discrete action space: stay, move up, move down, move left, and move right. In our experiments, prey follow a simple rule-based policy: if a prey remains within the spatial boundaries, it selects an action randomly from the action space; if it moves beyond the boundary, it selects the action that leads it back inside the boundary as quickly as possible. We set the number of blocks to zero, so only the predators (treated as agents) are controlled by RL algorithms to catch the prey. Additionally, the winning condition is defined as all prey being caught at least once.

At each timestep, agent k selects a discrete action u_t^k from its action space based on its observation $\mathbf{O}_t^k$, which includes

Map	Phase	ADAPT+FINE (%)	ADAPT+vote (%)	UPDeT (%)	OPT-CTDE (%)	OPT-DTDE (%)
3m	Training	98.75±1.53	98.75±1.53	-	99.13±0.55	99.33±0.35
5m	Zero-shot generalization	**47.16**±3.49	43.13±3.65	39.00	15.11±1.72	13.20±1.21
5m	Training	98.13±1.53	98.13±1.53	98.00	98.79±0.32	94.12±1.21
7m	Zero-shot generalization	**85.76**±3.09	73.44±10.36	63.00	80.11±1.40	20.20±1.99

Table 2: Performance comparisons of zero-shot generalization capabilities among ADAPT with two action selection strategies, UPDeT, OPT-CTDE and OPT-DTDE in **SMAC**, measured by test win rate. The upper section presents results where all methods are trained on the 3m map and evaluated via zero-shot generalization on 5m, while the lower section shows models trained on 5m and tested on 7m. The UPDeT results are sourced from the original paper (Hu et al. 2021). Although the final performance on 3m is not explicitly reported, the original paper indicates that training on 3m had converged prior to performing generalization.

Map	Phase	ADAPT+FINE (%)	ADAPT+vote (%)	ADAPT-share (%)	UPDeT (%)	OPT-CTDE (%)
3prey-3predator	Training	98.40±1.96	98.40±1.96	96.00±4.00	**100.00**±0.00	98.27±0.09
4prey-8predator	Zero-shot generalization	98.83±0.94	97.16±3.32	93.63±3.19	**99.25**±0.83	95.00±1.41
10prey-20predator		**91.83**±3.88	87.33±8.73	78.63±7.56	85.25±11.21	86.50±3.34
8prey-8predator		**87.67**±5.44	76.50±13.95	87.00±9.53	84.75±6.80	78.00±7.63
20prey-20predator		**79.17**±6.02	76.17±5.91	63.50±17.90	63.75±12.37	73.50±8.55
6prey-4predator		81.33±2.95	72.67±18.12	**88.25**±7.85	81.00±2.37	76.67±3.77
12prey-10predator		**73.50**±7.48	53.17±12.28	72.38±22.03	65.63±9.52	68.67±5.89
2prey-4predator	Training	83.20±5.88	83.20±5.88	**100.00**±0.00	92.80±8.45	99.44±0.08
4prey-8predator	Zero-shot generalization	97.70±1.21	98.40±1.50	**98.83**±1.65	91.20±5.04	95.10±0.54
10prey-20predator		91.00±5.22	84.40±10.47	92.83±5.31	72.00±7.69	**98.53**±1.80
8prey-8predator		78.40±9.32	**83.60**±7.42	81.33±22.52	45.80±11.03	70.57±5.70
20prey-20predator		66.60±12.72	62.40±17.91	61.00±28.15	42.00±13.56	**83.07**±7.03
6prey-4predator		75.60±10.93	**84.90**±8.03	72.83±30.68	33.60±8.36	32.10±8.47
12prey-10predator		**62.50**±14.99	62.30±13.82	60.50±30.43	29.40±10.89	59.77±12.09

Table 3: Performance comparisons of zero-shot generalization capabilities of ADAPT+FINE, ADAPT+vote, ADAPT-share, UPDeT, and OPT-CTDE in terms of test win rate across various predator-prey scenarios in **MPE**. All methods are trained to converge. The test win rates are reported as mean ± standard deviation.

the positions and velocities of itself and other observable objects, along with an indicator of whether the object has been captured. The reward function at each timestep is defined as follows: if the overall game-winning condition is met, all agents receive a reward of +20 at each timestep until the episode ends; additionally, if agent k captures a prey it has not caught before, it receives a +10 reward.

As shown in Table 3, we train ADAPT, ADAPT-share, UPDeT, and OPT-CTDE from scratch for up to 3.5 million timesteps on two source maps and evaluate their zero-shot generalization on a variety of unseen target maps. The target maps are selected based on two criteria: the quantitative relationship between prey and predators, and the total number of objects. Based on these two criteria, we select six representative maps for the zero-shot evaluation of policies trained on each source map. First, ADAPT+FINE outperforms ADAPT+vote on 9 out of 12 zero-shot maps, highlighting the effectiveness of the proposed FINE metric in revealing the generalization potential of ADAPT. Furthermore, the ADAPT series achieve state-of-the-art performance compared to UPDeT and OPT-CTDE across most generalization tasks. Notably, despite a 9.6% performance gap with UPDeT and a 16.24% gap with OPT-CTDE during training on the 2prey-4predator map, ADAPT+FINE still outperforms UPDeT across all corresponding transfer tasks and exceeds OPT-CTDE on 4 out of 6 zero-shot maps.

6 Conclusion

In this paper, we introduce ADAPT, the first DTDE-MARL architecture explicitly designed for structural generalization. ADAPT incorporates a self-supervised objective, termed dynamic consistency (DC) loss, which enables agents to derive local task-level context independently. To further improve and leverage ADAPT's generalization capabilities, we propose FINE (Foresight INdex for multi-agEnt), an action advising metric that accounts for policy-wise Q-value overestimation and enables cross-policy comparison of long-term action impact, thereby highlighting the policy that exhibits better foresight for more effective zero-shot generalization. Moreover, ADAPT is capable of handling arbitrary and varying numbers of objects and follows a DTDE paradigm, effectively mitigating issues related to data privacy, communication overhead, and computational cost. Empirical results demonstrate that ADAPT outperforms state-of-the-art baselines in both training-from-scratch and zero-shot generalization settings across a diverse suite of tasks.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No.62173258, T2495254), and by the State Key Laboratory of General Artificial Intelligence. The authors also thank Dr. Yuhan Zhao, algorithm engineer Juewen Hu, Ph.D. candidate Wentao Zhang, Ph.D. students Yi Ding and Jingyuan Huang for their valuable discussions during this work, and the AAAI-26 Student Scholar Volunteer Program for their support.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*.

Chu, T.; Wang, J.; Codecà, L.; and Li, Z. 2019. Multi-Agent Deep Reinforcement Learning for Large-scale Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3): 1086–1095.

de Witt, C. S.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P. H.; Sun, M.; and Whiteson, S. 2020. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? *arXiv preprint arXiv:2011.09533*.

El-Tantawy, S.; Abdulhai, B.; and Abdelgawad, H. 2013. Multiagent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers (MARLIN-ATSC): Methodology and Large-scale Application on Downtown Toronto. *IEEE Transactions on Intelligent Transportation Systems*, 14(3): 1140–1150.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual Multi-Agent Policy Gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1): 2974–2982.

Hou, Y.; Zhao, J.; Zhang, R.; Cheng, X.; and Yang, L. 2023. UAV Swarm Cooperative Target Search: A Multi-Agent Reinforcement Learning Approach. *IEEE Transactions on Intelligent Vehicles*, 9(1): 568–578.

Hu, S.; Zhu, F.; Chang, X.; and Liang, X. 2021. UPDeT: Universal Multi-agent RL via Policy Decoupling with Transformers. In *The Ninth International Conference on Learning Representations*.

Huber, P. J. 1964. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1): 73–101.

Iqbal, S.; De Witt, C. A. S.; Peng, B.; Boehmer, W.; Whiteson, S.; and Sha, F. 2021. Randomized Entity-wise Factorization for Multi-Agent Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, 4596–4606.

Jiang, J.; and Lu, Z. 2022. I2Q: A Fully Decentralized Q-Learning Algorithm. In *Advances in Neural Information Processing Systems*, volume 35, 20469–20481.

Koh, W.; Oh, W.; Kim, S.; Shin, S.; Kim, H.; Jang, J.; Lee, J.; and Yun, S.-Y. 2025. FlickerFusion: Intra-trajectory Domain Generalizing Multi-Agent Reinforcement Learning. In *The Thirteenth International Conference on Learning Representations*.

Lei, C.; Wu, S.; Yang, Y.; Xue, J.; and Zhang, Q. 2024. Joint Trajectory and Communication Optimization for Heterogeneous Vehicles in Maritime SAR: Multi-Agent Reinforcement Learning. *IEEE Transactions on Vehicular Technology*, 73(9): 12328–12344.

Liu, S.; Song, J.; Zhou, Y.; Yu, N.; Chen, K.; Feng, Z.; and Song, M. 2024. Interaction Pattern Disentangling for Multi-Agent Reinforcement Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12): 8157–8172.

Lowe, R.; WU, Y.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, volume 30, 1–12.

Qin, R.; Chen, F.; Wang, T.; Yuan, L.; Wu, X.; Kang, Y.; Zhang, Z.; Zhang, C.; and Yu, Y. 2024. Multi-Agent Policy Transfer via Task Relationship Modeling. *Science China Information Sciences*, 67(8): 182101.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2020. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Journal of Machine Learning Research*, 21(178): 1–51.

Samvelyan, M.; Rashid, T.; Witt, C. S. D.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.-M.; Torr, P. H. S.; Foerster, J. N.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge. *arXiv preprint arXiv:1902.04043*.

Shi, R.; Yu, X.; Wang, Y.; Tian, Y.; Liu, Z.; Wu, W.; Zhang, X.-P.; and Veloso, M. M. 2025. Symmetry-Informed MARL: A Decentralized and Cooperative UAV Swarm Control Approach for Communication Coverage. *IEEE Transactions on Mobile Computing*, 24(9): 8039–8056.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2017. Value-decomposition Networks for Cooperative Multi-Agent Learning. *arXiv preprint arXiv:1706.05296*.

Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; and Vicente, R. 2017. Multiagent Cooperation and Competition with Deep Reinforcement Learning. *PLoS ONE*, 12(4): e0172395.

Venturini, F.; Mason, F.; Pase, F.; Chiariotti, F.; Testolin, A.; Zanella, A.; and Zorzi, M. 2021. Distributed Reinforcement Learning for Flexible and Efficient UAV Swarm Control. *IEEE Transactions on Cognitive Communications and Networking*, 7(3): 955–969.

Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; and WU, Y. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*, volume 35, 24611–24624.

Yuan, L.; Zhang, Z.; Li, L.; Guan, C.; and Yu, Y. 2023. A Survey of Progress on Cooperative Multi-Agent Reinforcement Learning in Open Environment. *arXiv preprint arXiv:2312.01058*.

Zhou, W. J.; Subagdja, B.; Tan, A.-H.; and Ong, D. W.-S. 2021. Hierarchical Control of Multi-Agent Reinforcement Learning Team in Real-time Strategy (RTS) Games. *Expert Systems with Applications*, 186: 115707.