

SPECTra: Scalable Multi-Agent Reinforcement Learning with Permutation-Free Networks

Hyunwoo Park¹, Baekryun Seong¹, Sang-Ki Ko¹

¹University of Seoul

March 18, 2025

Abstract

In cooperative multi-agent reinforcement learning (MARL), the permutation problem—where the state space grows exponentially with the number of agents—reduces sample efficiency. Additionally, many existing architectures struggle with scalability, relying on a fixed structure tied to a specific number of agents, limiting their applicability to environments with a variable number of entities. While approaches such as graph neural networks (GNNs) and self-attention mechanisms have progressed in addressing these challenges, they have significant limitations as dense GNNs and self-attention mechanisms incur high computational costs. To overcome these limitations, we propose a novel agent network and a non-linear mixing network that ensure permutation-equivariance and scalability, allowing them to generalize to environments with various numbers of agents. Our agent network significantly reduces computational complexity, and our scalable hypernetwork enables efficient weight generation for non-linear mixing. Additionally, we introduce curriculum learning to improve training efficiency. Experiments on SMACv2 and Google Research Football (GRF) demonstrate that our approach achieves superior learning performance compared to existing methods. By addressing both permutation-invariance and scalability in MARL, our work provides a more efficient and adaptable framework for cooperative MARL. Our code is available at <https://github.com/funny-rl/SPECTra>.

1 Introduction

Reinforcement learning (RL) has achieved remarkable success in various domains such as games [Mnih et al.(2013), Silver et al.(2016), Vinyals et al.(2019)], robotics [Zhao et al.(2020), Kober et al.(2013)], and natural language processing [Christiano et al.(2017), Rafailov et al.(2023), DeepSeek-AI et al.(2025)]. However, traditional RL aims to train a single agent, limiting its applicability to complex problems requiring interaction among multiple agents. To address this limitation, multi-agent reinforcement learning (MARL) has emerged as a framework that enables two or more agents to interact and learn to achieve a

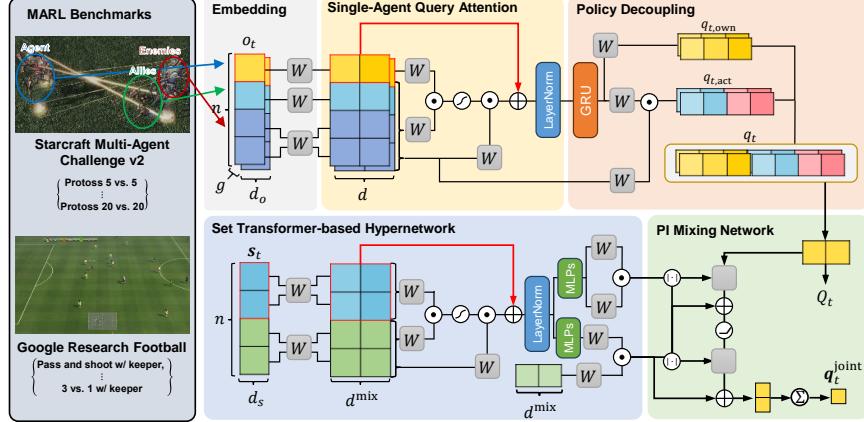


Figure 1: Overview of the proposed SPECTra framework. Gray boxes represent linear layers.

common goal. MARL offers significant potential to solve complex real-world challenges, such as dynamic resource allocation [Nguyen et al.(2018)], traffic control [Bazzan(2009)], and team sports [Lin et al.(2023)].

However, MARL faces significant challenges arising from the inherent complexity of multi-agent systems. One major hurdle is the scalability of the learning algorithms to varying numbers of agents. Traditional MARL algorithms using MLPs rely on fixed-size input representations [Rashid et al.(2018)], making them ill-suited for training with dynamic agent populations [Wang et al.(2020)]. In addition, the exponential growth rate of action space with the number of agents increases the exploration difficulty [Li et al.(2022), Wen et al.(2022)], forcing agents to take a long time to explore. Furthermore, entities an agent observes are implicitly ordered in sequence, leading to permutation problems where the state space grows exponentially with the number of agents. This reduces sample efficiency [Liu et al.(2019)].

Table 1: Comparison of the proposed framework with baselines in terms of two key properties.

Algorithm	Scalable	Permutation-Free
SPECTra-VDN	✓	✓
HPN-VDN	✗	✓
UPDeT-VDN	✓	✓
SPECTra-QMIX ⁺	✓	✓
HPN-QMIX	✗	✗
UPDeT-QMIX	✗	✗

To address these issues, we propose the **S**calable and **P**ermutation-free

Efficient Curriculum and Transferable learning framework based on Transformer architecture, called the SPECTra framework. The agent network and mixing network of SPECTra allow the networks to learn independently of the variable number of agents, even with a fixed structure. The implicit ordering of observation and action permutations is solved using permutation equivariance and permutation invariance. As a result, the agent network and mixing network improve sample efficiency compared to traditional methods and can learn from dynamic agent populations.

For the agent network to solve scalability and permutation problems, we propose single-agent query attention (SAQA), which reduces the time complexity to $O(nd)$. SAQA utilizes cross-attention by querying only information relevant to the observer agent. Unlike self-attention, which processes interactions between the observed entities in all directions, this approach focuses only on entities with which the observer interacts, effectively improving scalability [Sun et al.(2020)]. It also utilizes the policy decoupling [Hu et al.(2021b)] to compute action values independently of the scalability and permutation problems. We also design a set transformer-based hypernetwork (ST-HyperNet), which is a hypernetwork [Ha et al.(2016), Rashid et al.(2018)] that generates weight matrices and bias vectors of the mixing network to ensure the scalability and permutation-freeness of the mixing network while also maintaining the monotonicity of the network as in QMIX [Rashid et al.(2018)]. The agent network and mixing network can utilize their mathematical property to perform sample-efficient dynamic curriculum learning.

The main contributions of the paper are as follows:

- We propose the scalable and permutation-free MARL framework called SPECTra by employing Transformer-based architecture in agent networks and hypernetworks.
- We provide a theoretical analysis of the inference time of the proposed framework, especially the permutation freedom and computational efficiency, and a qualitative analysis of the sample complexity of SAQA.
- We demonstrate that SPECTra outperforms baselines such as UPDeT and HPN in MARL benchmarks such as SMACv2 and GRF, further showing that our framework is well-suited to curriculum learning and transfer learning settings.

2 Related Work

2.1 Multi-Agent Reinforcement Learning

Many cooperative MARL algorithms have used a centralized training decentralized execution (CTDE) [Lowe et al.(2017), Amato(2024)] framework to allow multiple agents to achieve a common goal. CTDE is a training method that calculates centralized value using each agent's extra signals during training, and each agent chooses an action independently using only its observations

during execution. VDN [Sunehag et al.(2017)] and QMIX [Rashid et al.(2018)] are representative value-based CTDE algorithms. VDN utilizes additive value decomposition to represent a linear joint-action value function as the sum of individual action values. Linear joint-action value function solves some cooperation problems, such as the lazy agent, but a linear mixing network can represent only a limited class of centralized action-value functions. QMIX significantly improves the representation capability by redesigning the linear joint-action value function of VDN into a non-linear mixing network satisfying Individual-Global-Max (IGM) [Son et al.(2019)] using hypernetworks. In addition, QMIX with normalized optimization [Hu et al.(2021a)] shows that QMIX can outperform QMIX’s variants [Wang et al.(2021), Yang et al.(2020), Rashid et al.(2020)] in SMAC [Samvelyan et al.(2019)] benchmark through regularized optimization. However, VDN and QMIX still can’t solve scalability and permutation problems.

2.2 Permutation-Invariant and Equivariant Modeling for Multi-Agent Systems

There have been many MARL frameworks that utilize permutation equivariance and invariance to address observation and action permutation problems [Hazra et al.(2024), McClellan et al.(2024), Li et al.(2021b)]. Among the frameworks, HPN [Hao et al.(2023)] has demonstrated an effective solution to the problems by independently embedding each observed entity using a permutation-equivariant hypernetwork. As a result, HPN has outperformed existing permutation-free architectures such as Deep Set [Zaheer et al.(2017)], self-attention [Vaswani et al.(2017)], and GNNs [Scarselli et al.(2009), Kipf and Welling(2017), Wu et al.(2021)].

However, HPN inherently applies a summation operation for achieving the permutation invariance on independent entity embeddings, preventing it from selectively focusing on specific entities to select the optimal action. Additionally, integrating a non-linear mixing network like QMIX cannot guarantee permutation invariance due to its reliance on simple MLPs, which are not inherently permutation-invariant. Furthermore, both the agent network and the mixing network face challenges in transfer learning and dynamic curriculum learning, as their architectures are dependent on the number of agents.

2.3 Dynamic Curriculum Learning

Curriculum learning has emerged as a powerful paradigm for accelerating MARL. Using a staged learning approach, curriculum learning begins with more straightforward tasks and progressively transitions to more complex challenges. In MARL, one standard method for defining task difficulty involves varying the number of agents. Increasing the number of agents inherently introduces greater environmental dynamics and stochasticity, making coordination and collaboration among agents significantly more challenging [Zhou et al.(2019)]. For instance, DymaCL [Wang et al.(2020)], a GNN-based curriculum learning framework, applied curriculum learning to simple agent networks. Their findings demonstrate that

even simple algorithms such as IQL [Tampuu et al.(2015)] and VDN can achieve more efficient training by leveraging curriculum learning by gradually increasing agents instead of training from scratch. Recent advancements in automatic curriculum learning [Wang et al.(2023), Long et al.(2020), Chen et al.(2021)] have introduced frameworks that automatically generate and adapt curricula by dynamically assigning the number of agents. These developments underscore the potential of designing models capable of adapting to varying agent counts, motivating further exploration into architectures that remain invariant to the number of agents.

A notable related effort is the UPDeT [Hu et al.(2021b)] framework, which, while not explicitly incorporating curriculum learning, offers an important example of a universal MARL pipeline. UPDeT employs transformer-based self-attention mechanisms to handle observations and action configurations of dynamic sizes using a unified architecture. This design partially enables zero-shot learning across agent configurations. However, UPDeT’s reliance on dense self-attention connections imposes an $O(n^2d)$ computational complexity, and MLPs-based hypernetwork limits UPDeT’s scalability of non-linear mixing networks. Moreover, self-attention computations significantly increase sample complexity due to dense connections, which diminishes the benefits of scalability [Sun et al.(2020)].

3 Background

3.1 Problem Formulation

We formulate the MARL problem as a decentralized partially observable Markov decision process (Dec-POMDP) [Bernstein et al.(2002), Oliehoek and Amato(2016)] with a 8-tuple $M = \langle S, A, U, P, r, O, n, \gamma \rangle$. Let S be the global state of the environment, $A = \{a_1, \dots, a_n\}$ be the set of n entities, and U is the action space. Since an agent is an independent decision-making entity that interacts with the environment, we denote the set of agents by A_{agent} and the number of agents by m . Namely, $|A_{\text{agent}}| = m$, where $n > m$.

We denote the *entity-wise observation* from entity a_1 to entity a_2 by $o_{1 \rightarrow 2} \in O$. Then, the local observation of entity a_i is denoted by a set of observations $o_i = \{o_{i \rightarrow j} \mid j \in [n]\}$. An agent a_i selects an action $u \in U$ at each time step t , and actions selected by m agents form a joint action $\mathbf{u} \in \mathbf{U} \equiv U^m$. We suppose that each agent a_i has its own action space U_i , which is independent of the permutation of agents. We also let $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \rightarrow [0, 1]$ denote the state transition probability from s to s' given the joint action \mathbf{u} . The immediate reward function $r(s, \mathbf{u}) : S \times \mathbf{U} \rightarrow R$, while $\gamma \in [0, 1)$ is a discount factor. At time step t , an agent a_i observes a joint observation $o_{i,t}$ and takes an action $u_t \in U_i$ using the policy $\pi_\theta(u_t | O_t)$ parameterized by θ . Now we can define the joint action value at time step t following the policy π_θ as follows:

$$Q^{\pi_\theta}(s_t, \mathbf{u}_t) = \mathbb{E}[R_t | s_t, \mathbf{u}_t],$$

where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the discounted return.

3.2 Permutation Invariance / Permutation Equivariance

In MARL, the order of agents can change, and it leads to an explosion in the number of possible input representations due to permutations. In the worst case, a naive approach would require considering all $n!$ possible permutations. Moreover, the output complexity also increases as the joint action space scales as U^m .

Let us first formally define the PI and PE attributes from the perspective of MARL as follows:

Definition 3.1 (permutation). *Let S_n be the set of all bijective functions (permutations) from the set $\{1, 2, \dots, n\}$ to itself.*

Definition 3.2 (permutation equivariance). *A function $f : X^n \rightarrow Y^n$ is permutation equivariant iff for any permutation $\sigma \in S_n$ and an input matrix $X \in \mathbb{R}^{n \times d}$, $f(P_\sigma X) = P_\sigma f(X)$, where $P_\sigma \in \mathbb{R}^{n \times n}$ is the permutation matrix corresponding to σ , which acts on X by permitting its rows.*

Definition 3.3 (permutation invariance). *A function $f : X^n \rightarrow Y^n$ is permutation invariant iff for any permutation $\sigma \in S_n$ and an input matrix $X \in \mathbb{R}^{n \times d}$, $f(P_\sigma X) = f(X)$, where $P_\sigma \in \mathbb{R}^{n \times n}$ is the permutation matrix corresponding to σ .*

To ensure that the overall framework remains permutation-free, it is not sufficient for the model architecture alone to be permutation-invariant or permutation-equivariant. The structure of observations and states must also meet specific conditions as follows:

1. First, each agent has a fixed index for encoding its own observation, meaning that the way an agent perceives itself does not change regardless of how other agents are ordered.
2. Second, when choosing an action, an agent can either take an action that directly affects itself (e.g., move) or an action that interacts with another entity (e.g., attack).
3. Lastly, since observations are represented as decomposable vectors, entities that are not directly observable due to partial observability should be masked, ensuring that missing information does not introduce unintended dependencies.

By maintaining these structural constraints, the entire learning process remains permutation-free, leading to more stable and efficient training.

4 Main Contributions

This section describes the agent network and mixing network of SPECTra in detail. First, we explain the important elements of the agent network: SAQA and action value estimation via policy decoupling. Then, prove that the agent network is scalable and permutation-equivariant. We also use a Set Transformer [Lee et al.(2019)]-based hypernetwork called the ST-HyperNet to show that non-linear mixing networks can estimate joint-action values in a permutation-invariant way and prove that the mixing network is scalable and permutation-invariant.

4.1 Lightweight Attention Mechanism: Single-Agent Query Attention (SAQA)

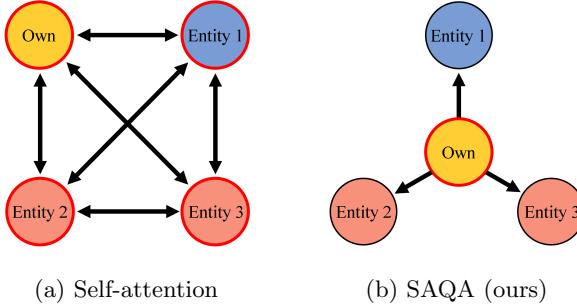


Figure 2: Comparison of connection density between self-attention and single-agent query attention.

To address the high computational cost of the self-attention mechanism, there have been alternative attention mechanisms proposed to reduce its complexity from $O(n^2d)$ to $O(nd^2)$, such as Efficient Attention [Shen et al.(2021)] and Hydra-Attention [Bolya et al.(2022)]. However, these algorithms have limited applicability because they require the hidden dimension d to be smaller than the number n of agents to be efficient. These limitations highlight the need for more tailored approaches to efficient attention mechanisms in MARL, ensuring computational scalability while preserving critical agent-centric information.

On the other hand, we propose a lightweight attention mechanism called *single-agent query attention (SAQA)*, which performs attention operations with a single agent information as a query and contextualizes the agent’s embedding with more relevant observations using the attention mechanism. In this way, we can achieve the computational cost of $O(nd)$ instead of $O(n^2d)$. Figure 2 Shows the difference in structural complexity between self-attention and SAQA.

Given the set of observations $o_i = \{o_{i \rightarrow j} \mid j \in A_i^{\text{obs}}\}$ from agent a_i , we first embed the observations via three separate linear layers E_{own} , E_{ally} , and E_{enemy}

follows:

$$\begin{aligned} e_i = & \{E_{\text{own}}(o_{i \rightarrow i})\} \cup \{E_{\text{ally}}(o_{i \rightarrow j}) \mid j \in A_{\text{ally}}\} \\ & \cup \{E_{\text{enemy}}(o_{i \rightarrow j}) \mid j \in A_{\text{enemy}}\} \subset \mathbb{R}^{d_h}. \end{aligned}$$

Note that $E_*(o_{i \rightarrow j}) \in \mathbb{R}^{d_h}$ is a vector that encodes the observation of entity a_j from agent a_i and defined as $o_{i \rightarrow j} \mathbf{W}_*$, where $\mathbf{W}_* \in \mathbb{R}^{d_o \times d_h}$ for $*$ $\in \{\text{own, ally, enemy}\}$. Note that ‘*’ is determined by the type of entity a_j from a_i ’s perspective (e.g., $*$ = own if $i = j$). Let us denote $E_*(o_{i \rightarrow j})$ by $e_{i \rightarrow j}$ for simplicity.

Let n_h be the number of attention heads in each layer. Then, $n_h \times d_k = d_h$, where d_k is the hidden dimension of each attention head. We compute the query, key, and value vector for k -th attention head by applying a linear transformation from d_h to d_k as follows:

$$Q_{i \rightarrow j}^{(k)}, K_{i \rightarrow j}^{(k)}, V_{i \rightarrow j}^{(k)} = e_{i \rightarrow j} \mathbf{W}_Q^{(k)}, e_{i \rightarrow j} \mathbf{W}_K^{(k)}, e_{i \rightarrow j} \mathbf{W}_V^{(k)}.$$

Now, we compute the attention-weighted vector for agent i :

$$e_{i,\text{att}}^{(k)} = \text{softmax} \left(\frac{Q_{i \rightarrow i}^{(k)} (\mathbf{K}_i^{(k)})^\top}{\sqrt{d_k}} \right) \mathbf{V}_i^{(k)} \in \mathbb{R}^{d_k}.$$

We concatenate the vectors from d_h heads as follows:

$$e_{i,\text{att}} = \begin{bmatrix} e_{i,\text{att}}^{(1)} & \dots & e_{i,\text{att}}^{(d_h)} \end{bmatrix}^\top \in \mathbb{R}^{d_h}$$

Lastly, we apply layer normalization after combining the attention-weighted vector with the input query embedding via a residual connection:

$$e'_i = \text{LayerNorm}(e_{i \rightarrow i} + e_{i,\text{att}}) \in \mathbb{R}^{d_h}.$$

Remark that the proposed SAQA can be performed with a time complexity of $O(nd)$. We can prove that the entire attention computation is permutation-invariant.

Proposition 4.1. *The single-agent query attention of the SPECTra framework is a permutation-invariant function.*

4.2 Permutation-Free Policy Decoupling

Remark that the actions against the other entities should be determined in a permutation-equivariant manner, as the input order should not affect the optimal decision-making of the current agent. We can design the policy network of each agent as a permutation-equivariant one by utilizing the permutation-invariant contextualized embedding e'_i from SAQA and the individual entity observations that are also free from the permutation problem.

We improve the policy decoupling proposed by UPDeT [Hu et al.(2021b)] to output action values permutation-equivariant using linear layers and inner products. Our policy decoupling uses the attention-based embedding of the current agent and the embedding of each observation-entity to compute action values for entities and optimize the policy at an action-group level.

To compute action values based on the action-observation history of agent a_i , we employ the GRU cell as follows:

$$h_{i,t} = \text{GRUCell}(e'_i, h_{i,t-1}) \in \mathbb{R}^d.$$

From the hidden state $h_{i,t}$, we compute two vectors: one ($Q_{i,t,\text{own}}$) for computing the action value for the actions on the agent itself (a_i) and the other ($Q_{i,t,\text{act}}$) for the actions on other agents (including enemies and allies) as follows:

$$\begin{aligned} q_{i,t,\text{own}} &= h_{i,t} \mathbf{W}_{\text{own}}^Q \in \mathbb{R}^{|U_{\text{own}}|} \\ Q_{i,t,\text{act}}^{(k)} &= e_i \mathbf{W}_{\text{act}}^{(k),Q} \in \mathbb{R}^{d_k} \text{ for } 1 \leq k \leq d_h. \end{aligned}$$

Now we apply a linear transformation to n embedded observations as follows:

$$\mathbf{K}_{i,t,\text{act}} = [e_{i \rightarrow i}^\top; e_{i \rightarrow 1}^\top; \dots; e_{i \rightarrow i-1}^\top; e_{i \rightarrow i+1}^\top; \dots; e_{i \rightarrow n}^\top] \mathbf{W}_{\text{act}}^K \in \mathbb{R}^{n \times d_k}.$$

Now we are ready to compute the q -values for the actions that can be performed on the other agents by conducting the dot product operations between a Q -vector and K -matrix:

$$q_{i,t,\text{act}} = \sum_{k=1}^{d_h} \frac{Q_{i,t,\text{act}}^{(k)} \left(\mathbf{K}_{i,t,\text{act}}^{(k)} \right)^\top}{d_h \sqrt{d_k}} \in \mathbb{R}^n.$$

Note that the q -vector is the result of the scaled inner product of $Q_{i,t,\text{act}}^{(k)}$ and $\mathbf{K}_{i,t,\text{act}}^{(k)}$. The K -matrix represents the latent information of individual observations of all agents. By applying the inner product, the observer agent can estimate the value of taking action on the other agents based on its observations.

Finally, we have the action value of a_i at t as follows:

$$\begin{aligned} q_{i,t} &= [q_{i,t,\text{own}}; q_{i,t,\text{act}}]^\top \in \mathbb{R}^{|U|} \\ q_{i,t}^{\text{mask}} &= \mathbf{Mask}_{i,t,\text{act}} \odot q_{i,t} + (1 - \mathbf{Mask}_{i,t,\text{act}}) \odot (-\infty), \end{aligned}$$

where $\mathbf{Mask}_{i,t,\text{act}} \in \{0, 1\}^{|U|}$.

We can use the inner product to output $q_{i,t,\text{act}}$ that is permutation-equivariant to the permutations of allies and enemies, and $\mathbf{W}_{\text{own}}^Q$ to make permutation-independent $q_{i,t,\text{own}}$ permutation-invariant.

Proposition 4.2. *The policy decoupling of SPECTra framework is a permutation-equivariant function.*

Proof. Since proposition 4.1, e'_i is permutation invariant. Thus, we only need to inspect $M_{\text{obs},t}^i$. Expression e_i are permutation equivariant. As other variables are not affected by observation permutation, the proposed policy decoupling method is permutation equivariant. \square

4.3 Set Transformer-based Hypernetwork

Hypernetworks [Ha et al.(2016)] refer to a class of neural architectures where one neural network generates the weights of another network. To ensure the permutation-equivariance of the mixing network, we employ the Set Transformer [Lee et al.(2019)] that consists of the permutation-equivariant encoder and the permutation-invariant decoder.

First we compute the query $\mathbf{Q}_{i,t}^{W,(k)}$ and key $\mathbf{K}_{i,t}^{W,(k)}$ values for the agent a_i by applying a linear transformation to the state representation of a_i . Then, ST-HyperNet computes the weight matrix and the bias vector of the mixing network as follows:

$$W_t^{\text{out}} = \sum_{k=1}^{d_h} \left| \frac{\mathbf{Q}_{i,t}^{W,(k)} (\mathbf{K}_{i,t}^{W,(k)})^\top}{d_h \sqrt{d^{\text{mix}}/d_h}} \right| \in \mathbb{R}_+^{m \times m}$$

$$b_t^{\text{out}} = \sum_{k=1}^{d_h} \frac{Q_{i,t}^{b,(k)} (\mathbf{K}_{i,t}^{b,(k)})^\top}{d_h \sqrt{d^{\text{mix}}}} \in \mathbb{R}^m$$

As in QMIX [Rashid et al.(2018)], absolute value operations are applied to the weights W_t^{out} of the mixing network to ensure monotonic value decomposition when computing the joint value function. Additionally, inspired by the decoder of the Set Transformer, we generate the permutation-equivariant bias vector b_t^{out} using a random seed vector.

Proposition 4.3. *The ST-HyperNet of the SPECTra framework is a permutation-equivariant function.*

4.4 Permutation-Invariant Mixing Network

Since $W_t^{\text{out}}, b_t^{\text{out}}$ obtained using ST-HyperNet are permutation equivariant, we can compute a permutation-invariant joint-action value using the ST-HyperNet as follows:

$$W_t^1, b_t^1 = \text{ST-HyperNet}^1(S_t)$$

$$W_t^2, b_t^2 = \text{ST-HyperNet}^2(S_t)$$

$$Q_t = [q_t^1 \quad \dots \quad q_t^m]^\top \in \mathbb{R}^m$$

$$H_t^{\text{mix}} = \text{ReLU}(Q_t W_t^1 + b_t^1) \in \mathbb{R}^m$$

$$q_t^{\text{joint}} = H_t^{\text{mix}} W_t^2 + b_t^2 \in \mathbb{R}$$

Simply speaking, since the weights and biases generated by the two ST-HyperNets depend on the order of the agents provided as input, the sequence of q -values follows the same order as the weights and biases. This ensures that the computed value of H_t^{mix} remains permutation-equivariant. Finally, the operations such as matrix multiplication and vector addition between H_t^{mix} and

the weight matrix and bias vector, which are given in the same permutation, are always guaranteed to be permutation-invariant.

Proposition 4.4. *The mixing network of SPECTra framework is a permutation-invariant function.*

5 Experiments

We evaluate our proposed SPECTra agent network and SPECTra mixing network on SMACv2 and GRF. We first evaluate our contribution to SMACv2, a successor to SMAC, a benchmark developed to improve the limitations of SMAC’s limited stochasticity by using probabilistic unit sampling and initial state distribution. SMACv2 introduces probabilistic unit sampling and diverse initial state distributions, incorporating random team compositions, starting positions, and unit types. This increased stochasticity highlights the necessity of addressing the permutation problem, reinforcing our motivation.

To ensure that SMACv2’s training difficulty increases with the number of agents, we turn off the SMACv2 action mask option to reduce the frequency of masked actions. Additionally, to account for the decentralized execution of CTDE, we set the probability of an agent observing an enemy within its sight range to $p = 1.0$. Under these settings, SMACv2 significantly increases training difficulty as the number of agents grows. To smooth the variability of the experimental results and clarify the trend, we applied an exponentially weighted moving average (EWMA) to the experimental results, with a smoothing factor of 0.99. Hyperparameters follow the HPN paper [Hao et al.(2023)], and detailed hyperparameter settings are described in Appendix B.

5.1 Comparison with Baselines

To evaluate the performance of our proposed algorithm, we first compare the performance of HPN, UPDeT, and SPECTra based on VDN and QMIX algorithms. The results of the experiments in the 10 vs 10 scenarios of SMACv2 are provided in Figure 3.

The results exhibit that SPECTra-QMIX⁺ and SPECTra-VDN show strong performance compared to the same QMIX and VDN-based models. SPECTra-QMIX and SPECTra-VDN train much faster than UPDeT-QMIX and UPDeT-VDN utilizing self-attention due to the low computational cost of SAQA. Moreover, as argued in Section 4.3, the difference in computation time between QMIX and VDN-based algorithms is not significant, which proves that the amount of computation in the agent network has a significant impact on training time than the amount of computation in the mixing network. The comparison of inference time for each model as the number of agents increases is described in Appendix A.1.

It is important to note that SPECTra-QMIX⁺ achieved superior performance with significantly fewer parameters than the other baselines. Unlike the baselines, where the number of parameters in the network increases with the number of

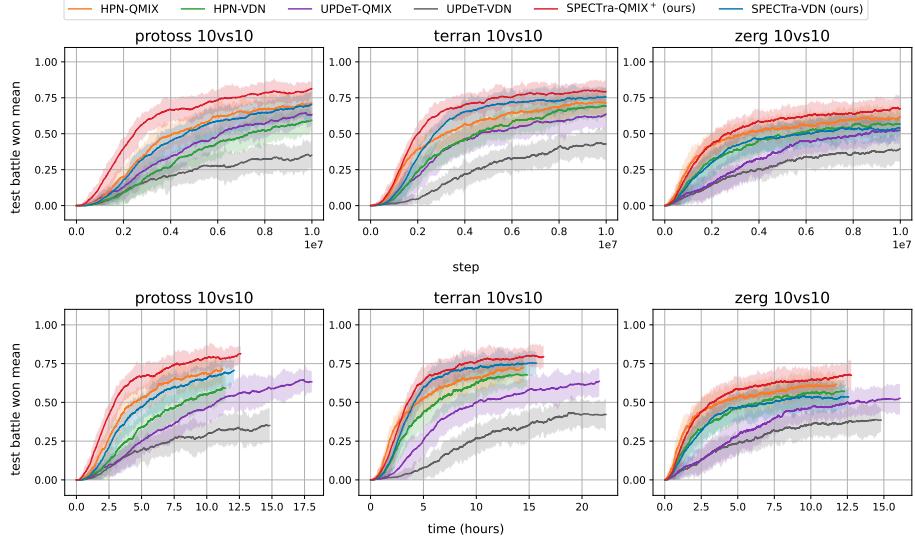


Figure 3: Comparison of SPECTra-QMIX⁺ and SPECTra-VDN against baselines in SMACv2.

agents, SPECTra-QMIX⁺ and SPECTra-VDN do not increase the number of parameters in the network as the number of agents to control increases. The number of parameters in SPECTra-QMIX⁺ is about 50% of UPDeT-QMIX and 33% of HPN-QMIX, and this difference increases further as the number of agents increases. SPECTra-VDN also has about 48% parameters than HPN-VDN in the above environment, but it performs competitively or is superior to HPN-VDN in most environments. UPDeT-VDN has fewer parameters than SPECTra-VDN, but its scalability is limited due to the sharp rise in computational cost. The parameters for each model can be found in Appendix C. Meanwhile, despite having more parameters than SPECTra-QMIX⁺, UPDeT-QMIX does not achieve better performance. The full experimental results can be found in the Appendix A.2.

5.2 Curriculum Learning

As shown in Table 1, SPECTra-QMIX⁺ is highly scalable, allowing it to adapt to larger environments without requiring additional parameters. This scalability makes it particularly well-suited for curriculum learning in settings with dynamically changing agent populations. While previous models like UPDeT have significantly improved transferability at the agent network level, they have not been extended to non-linear mixing networks.

Specifically, 30% of the total training occurs in a 5 vs 5 environment before transitioning to the target evaluation environment for the remaining 70%. We denote the variant of our model for curriculum learning by SPECTra-QMIX^{+CL}.

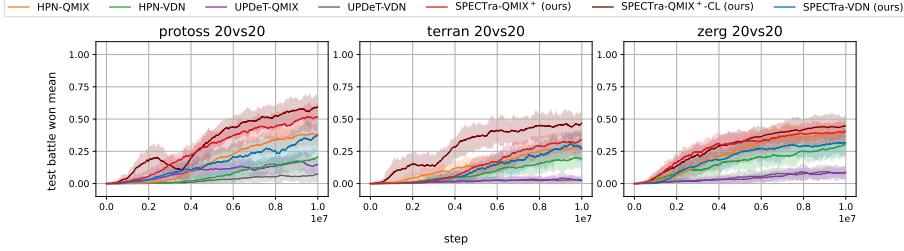


Figure 4: Comparison SPECTra-QMIX⁺ and SPECTra-VDN with curriculum learning against baselines in SMACV2.

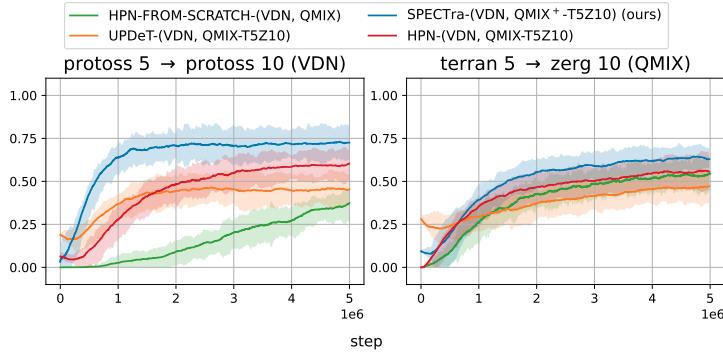


Figure 5: Comparison SPECTra-QMIX⁺ and SPECTra-VDN with transfer learning against baselines in protoss 5 → protoss 10 and terran 5 → zerg 10.

As shown in Figure 4, SPECTra-QMIX^{+CL} achieves superior performance compared to the standard SPECTra-QMIX⁺, demonstrating that even a simple curriculum learning strategy can enhance transferability.

Moreover, to evaluate the relative transfer ability of SPECTra-VDN and SPECTra-QMIX⁺, we perform transfer learning from 5 vs 5 to 10 vs 10 to see how well SPECTra, HPN, and UPDeT generalize policies. Moreover, to evaluate the relative transfer ability of SPECTra-VDN and SPECTra-QMIX⁺, we perform transfer learning from 5 vs 5 to 10 vs 10 to see how well SPECTra, HPN, and UPDeT generalize policies. To ensure that they understand games and not just unit-based repeated actions, we perform homology-based transfer learning and heterology-based transfer learning. The results in Figure 5 show that SPECTra-VDN and SPECTra-QMIX⁺ perform well on homologous and heterologous transfer learning. This means that SPECTra-VDN and SPECTra-QMIX⁺ can fully utilize the pre-trained models, which justifies the experimental results in Figure 4.

In the SMACv2 20 vs 20 environments, SPECTra-QMIX⁺ performs well in each environment despite having only 24% parameters of HPN-QMIX. On the other hand, UPDeT-QMIX using self-attention shows a sharp drop in learning

performance as the number of entities increases, despite having 3.4 times more parameters than SPECTra-QMIX⁺. This suggests that self-attention-based computation cannot cope with the exponential growth of complexity as the number of entities increases.

5.3 Google Research Football

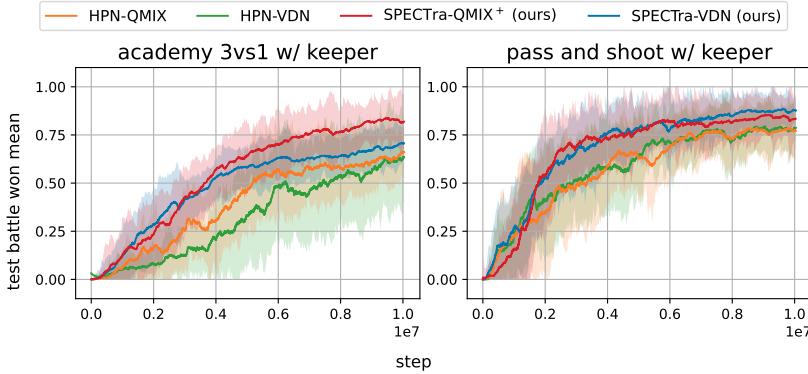


Figure 6: Comparison of SPECTra-QMIX⁺ and SPECTra-VDN against baselines in Google Research Football.

To ensure the general performance of our algorithm, we conduct comparative experiments not only in the SMACv2 environment but also in the GRF environment. Unlike SMACv2, GRF controls all the agents of the home team except the goalkeeper and receive sparse team rewards only based on goal scoring. Moreover, addressing the permutation problem is crucial in GRF, as the vast action space leads to an enormous state space. To analyze the performance of SPECTra-QMIX⁺ and SPECTra-VDN in GRF, we use HPN as the baseline, as HPN has demonstrated strong performance in this environment.

Unlike SMACv2, GRF has a fixed action space, which requires different methods to solve the permutation problem. Since HPN does not currently provide an available implementation for GRF, we reproduce the algorithm as faithfully as possible based on the details available [Hao et al.(2023)]. How we implemented algorithm in the GRF environment is described in detail in the Appendix D. To make SPECTra-QMIX⁺ and SPECTra-VDN applicable to GRF, we do not use policy decoupling and apply mean pooling across agents for pass actions, which is problematic for the permutation problem, respectively. The experimental results in Figure 6 demonstrate that both SPECTra-QMIX⁺ and SPECTra-VDN achieve strong learning performance in GRF and further prove that SPECTra algorithms generally exhibit robust and effective performance in various MARL environments. The full experimental results can be found in the Appendix A.3.

5.4 Ablation Study

We conduct ablation studies to evaluate the performance of each proposed component individually in SMACv2 environment. The full experimental results can be found in the Appendix A.4.

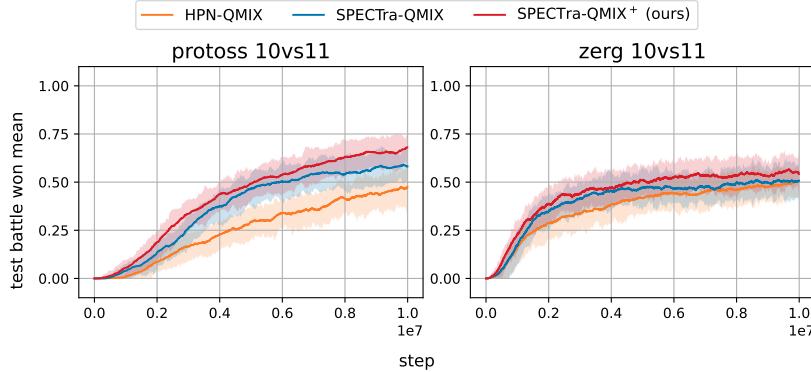


Figure 7: Experiments for the ablations of SAQA and ST-HyperNet modules.

5.4.1 Main Ablations

First, we analyze the effectiveness of the proposed agent networks and mixing networks in terms of performance. To evaluate the effectiveness of each module, we conduct an ablation study in a 10 vs 11 scenario across all races in SMACv2 (Protoss, Terran, and Zerg) using SPECTra-QMIX⁺, HPN-QMIX, and SPECTra-QMIX, where SPECTra-QMIX is a model that changes the mixer used in HPN-QMIX to ST-HyperNet. The experimental results are presented in Figure 13.

By comparing the experimental results of SPECTra-QMIX and HPN-QMIX, we confirm that our agent networks outperform the agent networks in the original HPN model. This supports our claim that performing attention on each entity’s observation embeddings is more effective than mean pooling. Furthermore, the comparison between SPECTra-QMIX⁺ and SPECTra-QMIX demonstrates that the permutation-invariance property of our non-linear mixing network significantly contributes to improving model performance.

5.4.2 SAQA vs Self-Attention

Recall that we propose SAQA to address the high computational cost and sample complexity of conventional self-attention. Here, we examine whether SAQA achieves superior performance compared to self-attention in terms of computational efficiency and performance. To evaluate the architectural differences between these two modules, we conduct experiments in a 5 vs 5 scenario across all races using SPECTra-VDN and SA-SPECTra-VDN, where SA-SPECTra-VDN is a model derived from SPECTra-VDN by replacing SAQA with a self-attention layer. Figure 8 shows the experimental results.

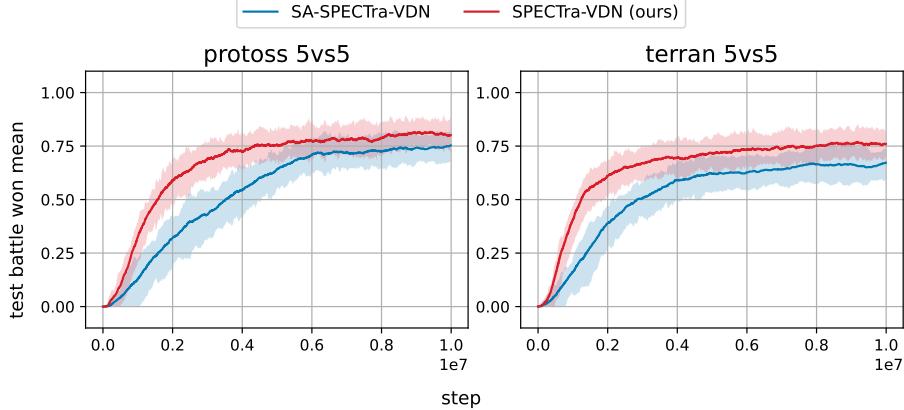


Figure 8: Comparison of SAQA and self-attention layers with mean pooling.

Note that we use all observations along with the current agent information as queries, and the output embeddings of the self-attention are aggregated using a mean pooling operation. The experimental results demonstrate that self-attention unnecessarily increases sample complexity, whereas SAQA achieves higher performance with significantly lower computational cost.

We have conducted a qualitative analysis to examine how SAQA and self-attention layers interpret input observations differently. Figure 9 compares the attention scores of SAQA and self-attention when an observation from SMACv2 is input into SPECTra and UPDeT. To analyze the response to entities that were not observed and thus processed as zero vectors, we trained the models without applying masks to any attention. While SPECTra selected the attack action, UPDeT chose the move action. In the given state, the optimal action of the agent is to attack and eliminate Enemy 4 as quickly as possible since the enemy entity has significantly lower HP compared to Enemy 1. SAQA utilizes residual connections, allowing it to pay less attention to the query’s own information. This enables each head to focus on different aspects of the four observed objects. This suggests that all heads clearly understand the subject of the observation, indicating that the model can learn a unidirectional relationship centered on the observer. In contrast, UPDeT employs attention by using each entity’s vector as the basis for selecting an action corresponding to that entity. The most crucial information for selecting an action should be the relationship between the observer agent and the given entity. To compute the action value for Enemy 4, the model should assign a high attention score to the relationship between the agent and Enemy 4. However, UPDeT tends to focus more on Ally 3 than on Enemy 4. This suggests that UPDeT struggles to prioritize the complex relationships among entities.

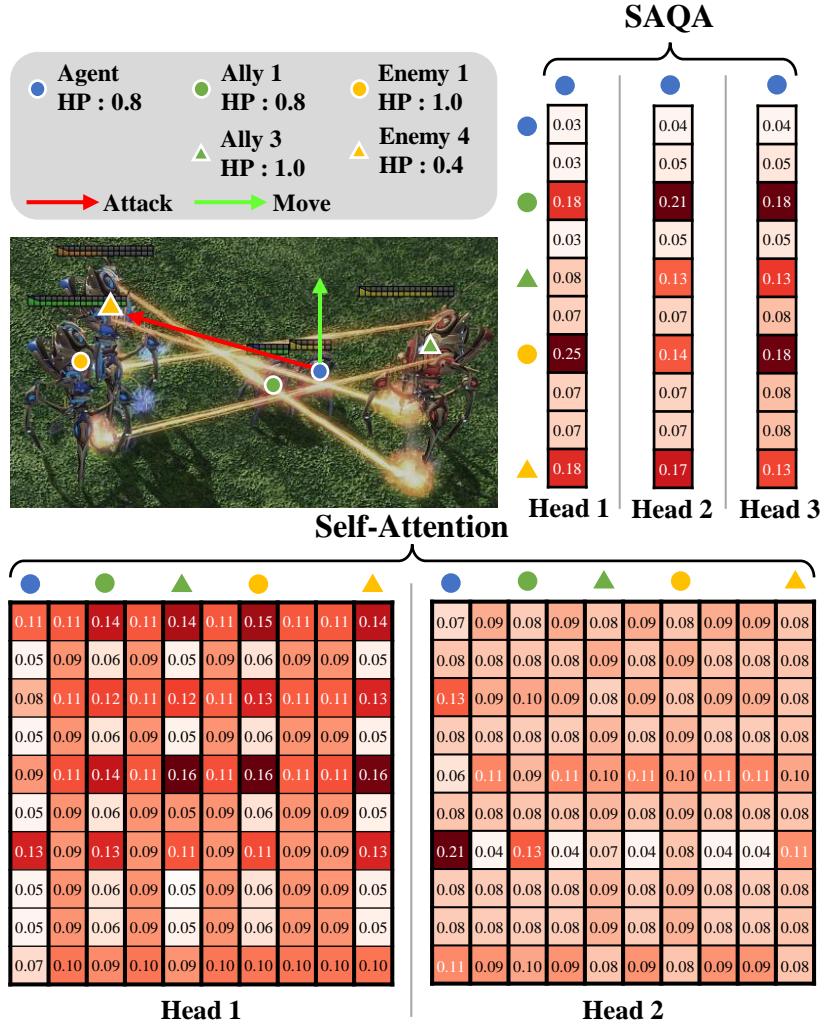


Figure 9: Example of attention maps computed from SAQA and self-attention layers (UPDeT) in SMACv2 environment.

6 Conclusions

In this work, we have proposed a MARL framework called the SPECTra that leverages Transformer-based neural architecture to address the scalability and permutation problems. By utilizing SAQA, we have mitigated the high computational cost of self-attention layers while maintaining the ability to learn complex interactions between many entities. Our experimental results confirm that SAQA accelerates the training process and achieves superior performance compared to baseline approaches. Additionally, we have successfully addressed the scalability

and permutation problems in non-linear mixing networks by introducing the Set Transformer-inspired hypernetwork called the ST-HyperNet.

Furthermore, the SPECTra framework has demonstrated strong applicability and generality by the results of transfer learning and curriculum learning. Our study presents an efficient way to adopt the well-known attention mechanism in highly complex MARL scenarios, and it holds the potential to be extended to real-world scale environments that require the capability to model the interactions among a significant number of agents.

References

- [Amato(2024)] Christopher Amato. 2024. An Introduction to Centralized Training for Decentralized Execution in Cooperative Multi-Agent Reinforcement Learning. *CoRR* abs/2409.03052 (2024).
- [Bazzan(2009)] Ana L. C. Bazzan. 2009. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Auton. Agents Multi Agent Syst.* 18, 3 (2009), 342–375.
- [Bernstein et al.(2002)] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research* 27, 4 (2002), 819–840.
- [Bolya et al.(2022)] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, and Judy Hoffman. 2022. Hydra Attention: Efficient Attention with Many Heads. In *Computer Vision - ECCV 2022 Workshops - Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part VII (Lecture Notes in Computer Science, Vol. 13807)*, Leonid Karlinsky, Tomer Michaeli, and Ko Nishino (Eds.). Springer, 35–49.
- [Chen et al.(2021)] Jiayu Chen, Yuanxin Zhang, Yuanfan Xu, Huimin Ma, Huazhong Yang, Jiaming Song, Yu Wang, and Yi Wu. 2021. Variational Automatic Curriculum Learning for Sparse-Reward Cooperative Multi-Agent Problems. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 9681–9693.
- [Christiano et al.(2017)] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 4299–4307.

[DeepSeek-AI et al.(2025)] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghai Lu, Shangyan Zhou, Shanhua Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanja Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL]

[Ha et al.(2016)] David Ha, Andrew M. Dai, and Quoc V. Le. 2016. HyperNetworks. *CoRR* abs/1609.09106 (2016).

[Hao et al.(2023)] Jianye Hao, Xiaotian Hao, Hangyu Mao, Weixun Wang, Yaodong Yang, Dong Li, Yan Zheng, and Zhen Wang. 2023. Boosting Multiagent Reinforcement Learning via Permutation Invariant and Permutation Equivariant Networks. In *The Eleventh International Conference*

on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.
OpenReview.net.

- [Hazra et al.(2024)] Somnath Hazra, Pallab Dasgupta, and Soumyajit Dey. 2024. Addressing Permutation Challenges in Multi-Agent Reinforcement Learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS '24)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2303–2305.
- [Hu et al.(2021a)] Jian Hu, Haibin Wu, Seth Austin Harding, Siyang Jiang, and Shih-wei Liao. 2021a. RIIT: Rethinking the Importance of Implementation Tricks in Multi-Agent Reinforcement Learning. *CoRR* abs/2102.03479 (2021).
- [Hu et al.(2021b)] Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. 2021b. UPDeT: Universal Multi-agent RL via Policy Decoupling with Transformers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- [Kipf and Welling(2017)] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.
- [Kober et al.(2013)] Jens Kober, J. Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *Int. J. Robotics Res.* 32, 11 (2013), 1238–1274.
- [Lee et al.(2019)] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3744–3753.
- [Li et al.(2021a)] Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. 2021a. Celebrating Diversity in Shared Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 3991–4002.
- [Li et al.(2022)] Pengyi Li, Hongyao Tang, Tianpei Yang, Xiaotian Hao, Tong Sang, Yan Zheng, Jianye Hao, Matthew E. Taylor, Wenyuan Tao, and Zhen Wang. 2022. PMIC: Improving Multi-Agent Reinforcement Learning with Progressive Mutual Information Collaboration. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland*,

USA (Proceedings of Machine Learning Research, Vol. 162), Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 12979–12997.

[Li et al.(2021b)] Yan Li, Lingxiao Wang, Jiachen Yang, Ethan Wang, Zhaoran Wang, Tuo Zhao, and Hongyuan Zha. 2021b. Permutation Invariant Policy Optimization for Mean-Field Multi-Agent Reinforcement Learning: A Principled Approach. arXiv:2105.08268 [cs.LG]

[Lin et al.(2023)] Fanqi Lin, Shiyu Huang, Tim Pearce, Wenze Chen, and Wei-Wei Tu. 2023. TiZero: Mastering Multi-Agent Football with Curriculum Learning and Self-Play.

[Liu et al.(2019)] Iou-Jen Liu, Raymond A. Yeh, and Alexander G. Schwing. 2019. PIC: Permutation Invariant Critic for Multi-Agent Deep Reinforcement Learning. In *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings (Proceedings of Machine Learning Research, Vol. 100)*, Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (Eds.). PMLR, 590–602.

[Long et al.(2020)] Qian Long, Zihan Zhou, Abhinav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. 2020. Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[Lowe et al.(2017)] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *CoRR* abs/1706.02275 (2017).

[McClellan et al.(2024)] Joshua McClellan, Naveed Haghani, John Winder, Furong Huang, and Pratap Tokek. 2024. Boosting Sample Efficiency and Generalization in Multi-agent Reinforcement Learning via Equivariance. *CoRR* abs/2410.02581 (2024).

[Mnih et al.(2013)] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013).

[Nguyen et al.(2018)] Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. 2018. Credit Assignment For Collective Multiagent RL With Global Rewards. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 8113–8124.

- [Oliehoek and Amato(2016)] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer.
- [Rafailov et al.(2023)] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).
- [Rashid et al.(2020)] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).
- [Rashid et al.(2018)] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. arXiv e-prints, page. *arXiv preprint arXiv:1803.11485* (2018).
- [Samvelyan et al.(2019)] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, 2186–2188.
- [Scarselli et al.(2009)] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Trans. Neural Networks* 20, 1 (2009), 61–80.
- [Shen et al.(2021)] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient Attention: Attention with Linear Complexities. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*. IEEE, 3530–3538.
- [Silver et al.(2016)] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis

- Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nat.* 529, 7587 (2016), 484–489.
- [Son et al.(2019)] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. *CoRR* abs/1905.05408 (2019).
- [Sun et al.(2020)] Chuangchuang Sun, Macheng Shen, and Jonathan P. How. 2020. Scaling Up Multiagent Reinforcement Learning for Robotic Systems: Learn an Adaptive Sparse Communication Graph. *CoRR* abs/2003.01040 (2020).
- [Sunehag et al.(2017)] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2017. Value-Decomposition Networks For Cooperative Multi-Agent Learning. *CoRR* abs/1706.05296 (2017).
- [Tampuu et al.(2015)] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2015. Multiagent Cooperation and Competition with Deep Reinforcement Learning. *CoRR* abs/1511.08779 (2015).
- [Vaswani et al.(2017)] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008.
- [Vinyals et al.(2019)] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çağlar Gülcöhre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nat.* 575, 7782 (2019), 350–354.
- [Wang et al.(2021)] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

- [Wang et al.(2023)] Rundong Wang, Longtao Zheng, Wei Qiu, Bowei He, Bo An, Zinovi Rabinovich, Yujing Hu, Yingfeng Chen, Tangjie Lv, and Changjie Fan. 2023. Towards Skilled Population Curriculum for Multi-Agent Reinforcement Learning. *CoRR* abs/2302.03429 (2023).
- [Wang et al.(2020)] Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. 2020. From Few to More: Large-Scale Dynamic Multiagent Curriculum Learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 7293–7300.
- [Wen et al.(2022)] Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. 2022. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem. arXiv:2205.14953 [cs.MA]
- [Wu et al.(2021)] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24.
- [Yang et al.(2020)] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. 2020. Qatten: A General Framework for Cooperative Multiagent Reinforcement Learning. arXiv:2002.03939 [cs.MA]
- [Zaheer et al.(2017)] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. 2017. Deep Sets. *CoRR* abs/1703.06114 (2017).
- [Zhao et al.(2020)] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. 2020. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020, Canberra, Australia, December 1-4, 2020*. IEEE, 737–744.
- [Zhou et al.(2019)] Ming Zhou, Yong Chen, Ying Wen, Yaodong Yang, Yufeng Su, Weinan Zhang, Dell Zhang, and Jun Wang. 2019. Factorized Q-learning for large-scale multi-agent systems. In *Proceedings of the First International Conference on Distributed Artificial Intelligence, DAI 2019, Beijing, China, October 13-15, 2019*. ACM, 7:1–7:7.

A Additional Experiments

A.1 Comparison of Inference Time

Figure 10 compares the inference time of each algorithm with 1,000 samples and shows that SAQA requires significantly less computation time than self-attention, supporting the results in Figure 3. It is faster than UPDeT and outperforms HPN, and we confirm that this trend is consistent as the number of agents increases.

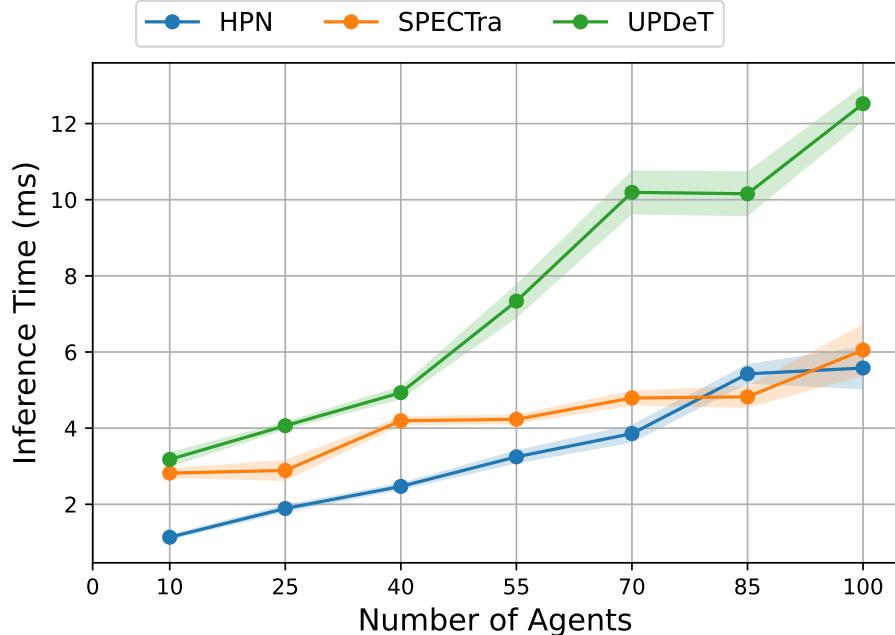


Figure 10: Comparison of inference time between the proposed SPECTra and baseline methods.

A.2 SMACv2 Experiments

Figure 11 compares SPECTra and other baseline algorithms in SMACv2 environment. SPECTRA-QMIX⁺ shows outstanding performance in overall experiments.

A.3 GRF Experiments

Figure 12 shows the comparison between SPECTra and HPN. Each of them is combined to QMIX or VDN. Used scenarios in the figure are *academy 3 vs*

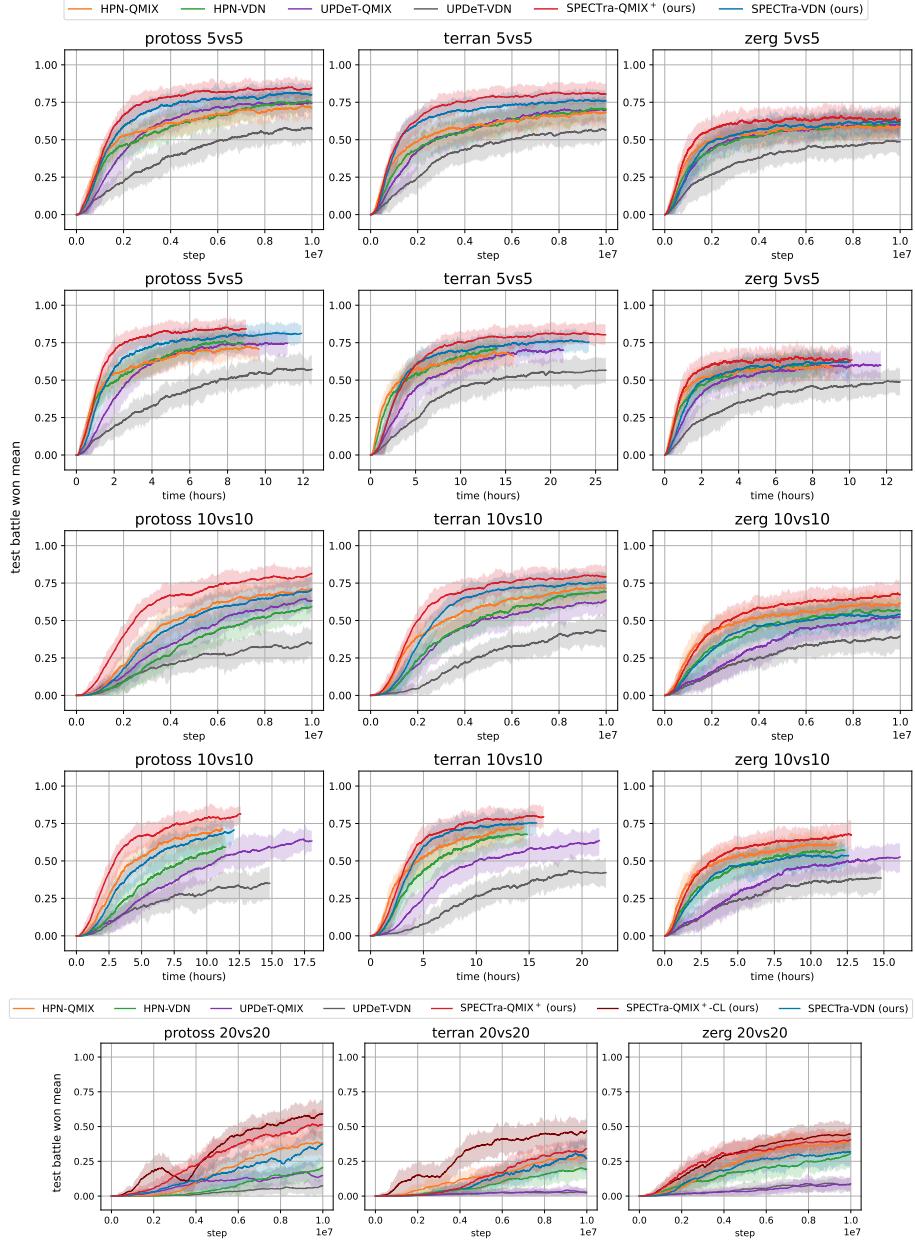


Figure 11: Overall comparison over SMACv2 challenges 5 vs 5, 10 vs 10, 20 vs 20.

1 with keeper, academy pass and shoot with keeper, and academy run pass and shoot with keeper, respectively.

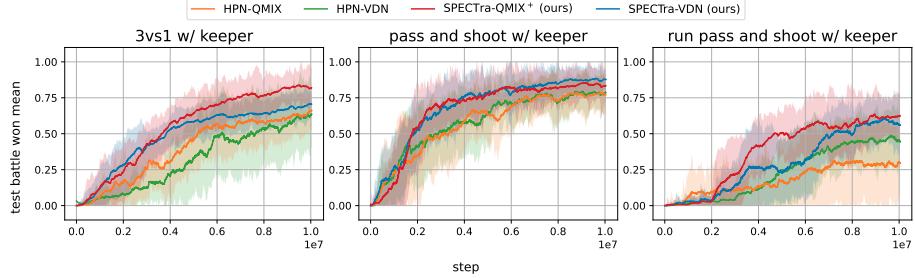


Figure 12: Comparison of SPECTra-QMIX⁺ and SPECTra-VDN at the 3 maps against baselines in Google Research Football.

A.4 Ablation Experiments

To understand how each module of the proposed algorithm contributes to performance, we conducted two ablation studies. Figure 13 confirms that the proposed agent network and mixing network contribute to performance improvement. Additionally, Figure 14 shows that SAQA is more effective than embedding generation using conventional self-attention and directly impacts performance.

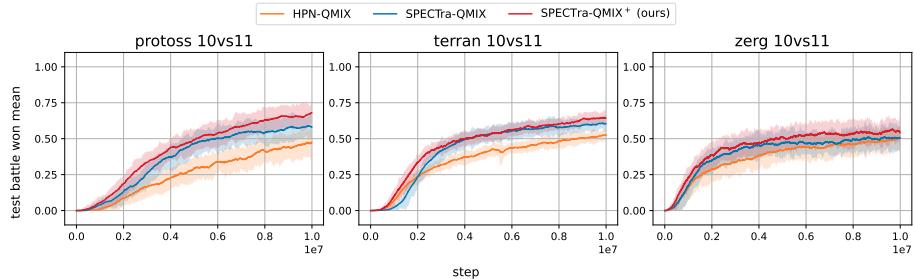


Figure 13: Ablation experiments for SAQA and ST-HyperNet modules.

B Hyperparameter

During experiments, the implementations of baseline methods are consistent with their official repositories. We list the detailed hyperparameter settings we used in our SMACv2, GRF, and Ablation study in the paper in the Table 2 ~ 5 below.

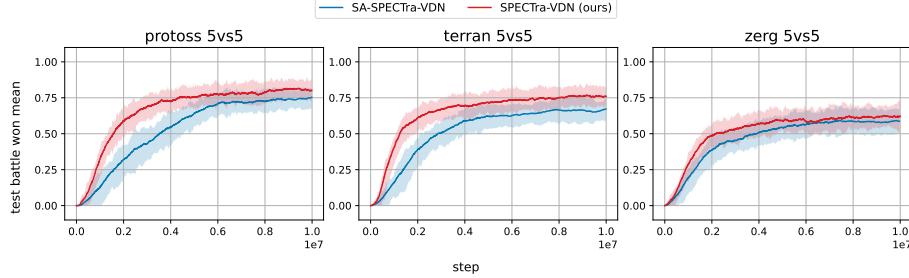


Figure 14: Comparison of SAQA and self-attention layers with mean pooling in 3 races.

B.1 Hyperparameters in SMACv2 Experiments

In SMACv2, the batch_size is set differently depending on the total number of entities to manage memory stably. Specifically, 128 for 5 vs 5 environments, 32 for 10vs10, and 10 for 20vs20. In addition, SPECTra-QMIX⁺ uses the attention scores of ST-HyperNet as the weight and bias of the mixing network, and these values change dynamically with the number of agents. Therefore, the number of agents dynamically determines the mixing_embed_dim.

Hyperparameter	SPECTra-QMIX ⁺	HPN-QMIX	UPDeT-QMIX	SPECTra-VDN	HPN-VDN	UPDeT-VDN
mac	ss_mac	hpns_mac	updet_mac	ss_mac	hpns_mac	updet_mac
agent	ss_rnn	hpns_rnn	updet_agent	ss_rnn	hpns_rnn	updet_agent
learner	nq_learner	nq_learner	nq_learner	nq_learner	nq_learner	nq_learner
mixer	ss_mixer	qmix	qmix	vdn	vdn	vdn
hidden_size	64	64	32	64	64	32
n_head	4	-	3	4	-	3
transformer_depth	1	-	2	1	-	2
mixing_n_head	4	-	-	-	-	-
mixing_embed_dim	number of agents	32	32	-	-	-
hypernet_embed	32	64	64	-	-	-
batch_size	128	128	128	128	128	128

Table 2: Hyperparameter Settings of SMACv2 Experiment Algorithms.

B.2 Hyperparameters in GRF Experiments

GRF is one of the most challenging environments for MARL benchmarks due to its score-based sparse rewards and long-term episodes. If we use only the score-based reward function of the environment, the over-exploration will significantly degrade training speed and performance. So, it is necessary to customize and override the reward function to prevent over-exploration. Also, since the action space differs depending on the environment, each model should use an optimized action sampling method for solving the permutation problem according to the environment. Since implementing this part of the HPN [Hao et al.(2023)] paper is not publicly available in code, we designed a GRF training based on the GitHub project published by CDS [Li et al.(2021a)]. We implemented a model that follows the architecture described in the HPN [Hao et al.(2023)] paper. We

also tried accelerating training by using additional action masks and reward functions. All implementations are 100% publicly available on our GitHub project. Below is the implementation of the hyperparameter table we used to train the model: ??.

Hyperparameter	SPECTra-QMIX ⁺	HPN-QMIX	SPECTra-VDN	HPN-VDN
mac	ss_mac	hpns_mac	ss_mac	hpns_mac
agent	ss_rnn	hpns_rnn	ss_rnn	hpns_rnn
learner	nq_learner	nq_learner	nq_learner	nq_learner
mixer	ss_mixer	qmix	vdn	vdn
hidden_size	64	64	64	64
n_head	4	-	4	-
transformer_depth	1	-	1	-
mixing_n_head	4	-	-	-
mixing_embed_dim	-	32	-	-
hypernet_embed	32	64	-	-
batch_size	32	32	32	32

Table 3: Hyperparameter Settings of Google Research Football Experiment.

B.3 Hyperparameters in Ablation Studies

We have conducted ablation studies to evaluate the performance of each proposed component individually. As discussed in Section 5.4, each ablation study demonstrates that the proposed components contribute to the performance of our model in a positive manner. Unless otherwise mentioned, the hyperparameters used in the experiments follow the values provided in Appendix B.1. The hyperparameter settings for the ablation studies are presented in Tables ?? to 5.

Hyperparameter	SPECTra-QMIX ⁺	SPECTra-QMIX	HPN-QMIX
mixer	ss_mixer	qmix	qmix
n_head	4	4	-
transformer_depth	1	1	-
mixing_n_head	1	-	-
mixing_embed_dim	-	32	-
hypernet_embed	32	64	-
batch_size	32	32	32

Table 4: Hyperparameter Settings of Ablation 5.4.1 Experiment Algorithms.

C Model Parameters

In this section, we compare the number of parameters in the models used in our experiments.

Hyperparameter	SPECTra-VDN	SA-SPECTra-VDN
mixer	vdn	vdn
n_head	4	4
transformer_depth	1	1
batch_size	128	128
use_SAQA	True	False

Table 5: Hyperparameter Settings of Ablation 5.4.2 Experiment Algorithms.

C.1 Model Parameters in SMACv2 Experiments

In the Protoss scenario, ‘‘shield’’ features are added to the feature information, resulting in different feature dimensions compared to scenarios of other races, which leads to a different number of parameters. However, since Terran and Zerg share the same feature dimensions, their architectures have the same number of parameters. As shown in Table ??, both SPECTra-QMIX⁺ and SPECTra-VDN maintain the same number of parameters regardless of the number of agents and enemies, making them scalable. In contrast, HPN-QMIX and HPN-VDN are not scalable, as the architecture of HPN is dependent on the number of agents. For UPDeT, while UPDeT-VDN remains scalable as we simply add Q -values in VDN, UPDeT-QMIX is not scalable as its mixing network is still dependent on the number of agents and its number of parameters also varies with the number of entities.

Race	Map	SPECTra-QMIX ⁺	HPN-QMIX	UPDeT-QMIX	SPECTra-VDN	HPN-VDN	UPDeT-VDN
Protoss	5 vs 5	72.90	144.488	80.903	48.326	106.823	43.238
	10 vs 10	72.90	189.768	125.863	48.326	107.143	43.238
	20 vs 20	72.90	309.128	244.583	48.326	107.783	43.238
Terran	5 vs 5	72.582	134.056	78.951	48.134	98.311	43.206
	10 vs 10	72.582	177.416	121.991	48.134	98.631	43.206
	20 vs 20	72.582	292.936	236.871	48.134	99.271	43.206
Zerg	5 vs 5	72.582	134.056	78.951	48.134	98.311	43.206
	10 vs 10	72.582	177.416	121.991	48.134	98.631	43.206
	20 vs 20	72.582	292.936	236.871	48.134	99.271	43.206

Table 6: Number of parameters each algorithm has according to the SMACv2 maps (in thousands, k).

C.2 Model Parameters in GRF Experiments

Due to differences in action space, GRF and SMACv2 require separately designed layers for estimating action values. To ensure effective training, we adapt the models as described in Appendix D. Table ?? below presents the number of parameters for the models implemented in GRF.

Scenario	SPECTra-QMIX ⁺	HPN-QMIX	SPECTra-VDN	HPN-VDN
3 vs 1 w/ keeper	72.844	86.901	56.787	73.364
pass and shoot w/ keeper	72.844	84.053	56.787	73.364
run pass and shoot w/ keeper	72.844	84.053	56.787	73.364

Table 7: Number of parameters each algorithm has according to the GRF scenarios (in thousands, k).

D Model Adaptations for Google Research Football

This section describes how the experimental algorithms were implemented in the GRF benchmark. Unlike SMACv2, where an action is performed by identifying a specific entity, GRF does not allow agents to control which ally to pass the ball to directly. The GRF has a fixed number of actions, so we must change how the agent network outputs actions to address scalability and permutation issues. First, to implement HPN-QMIX and HPN-VDN, we need to consider the action permutations. GRF has 19 actions, including permutation-invariant actions such as moving, sliding, and shooting, and permutation-equivariant actions such as long pass, high pass, and short pass. We apply the max pooling operation to all friendly-related action values to obtain the final q-values for the three pass actions. To implement SPECTra-QMIX⁺ and SPECTra-VDN, we remove the policy decoupling modules and apply the mean pooling operation for the three pass actions to obtain their final q-values. The above implementation is 100% public.

E Computation Resource

Experiments were conducted on the Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz 24 core processor with 768GB RAM and NVIDIA A10 D6 24GB.