QattenLinear: A General Framework for Cooperative Multiagent Reinforcement Learning

Chao Li, Chennan Ma, Zhuohan Ye, Haibing Jin, Haoran Liu
School of Informatics Xiamen University, China
{23020221154092, 23020221154102, 23020221154138, 23020221154091, 23020221154101}@stu.xmu.edu.cn

Abstract

Multi-agent Reinforcement Learning (MARL) has been widely recognized as a superior approach by enabling a team of agents coordinate their behaviors while acting in a decentralized way. One representative class of work is value decomposition, which decomposes the global shared multi-agent Q-value Q_{tot} into individual Q-values Q^i. One of the key challenges in MARL is performance degradation of multi-agent systems in complex environments brought by a large number of agents. Moreover, most previous works have some assumptions about the relationship between Q_{tot} and Q^i, which lacking theoretical basis. To tackle the problem caused by the massive agents in MARL, We propose QattenLinear, an improved deep Q-value factorization network based on Qatten, which is based on a linear attention mechanism, allowing easy maximization of joint action values in off-policy learning and guaranteeing consistency between centralized and decentralized policies. We use the differentiable key-value memory model to estimate the coefficients and derive the relations from the agents to the global. Moreover, instead of using the native attention mechanism to calculate the attention weights, QattenLinear uses linear attention to calculate the attention weights in order to optimize the time complexity from square level to linear level. Extensive experiments on the widely used StarCraft II benchmark show that our method is superior to the state-of-the-art MARL methods in different scenarios.

Introduction

Reinforcement learning has been widely used in various fields, such as robots, games, recommendation systems, etc. (Wang et al. 2018) . As a branch of artificial intelligence, reinforcement learning is an important path to realize decision intelligence. It divides the world into two parts: agent and environment.(Sutton and Barto 2018) Agents interact with the environment by performing actions and get feedback from the environment. In reinforcement learning, feedback from the environment is reflected in the form of rewards, as shown in Figure 1.

Multi Agent System (MAS) refers to the existence of multiple agents that need to be controlled within a system. The cooperative multi-agent reinforcement learning problem has achieved lots of attention in the last decade, where a system of agents learns towards coordinated policies to optimize the accumulated global rewards. The change of environmental state is not determined by a single agent, but by the joint behavior of all agents. The idea of the early multi-agent reinforcement learning algorithm is very intuitive, that is, an independent single-agent reinforcement learning algorithm is used to control each agent in the multi-agent system, and each algorithm model is independent of each other. In this case, the actions of the other agents will have an impact on the environmental state, but do not improve the environmental instability brought by the training of the reinforcement learning algorithm. However, the joint state action search space will increase exponentially with the increase of the number of agents. Another choice is the decentralized approach that each agent learns its policy. Letting individual agents learn concurrently based on the global reward (aka. independent learners) (Tan 1993) is the simplest option. However, it is shown to be difficult in even simple two-agent, single-state stochastic coordination problems. One main reason is that the global reward signal brings the non-stationarity that agents cannot distinguish between the stochasticity of the environment and the exploitative behaviors of other co-learners (Lowe et al. 2017), and thus may

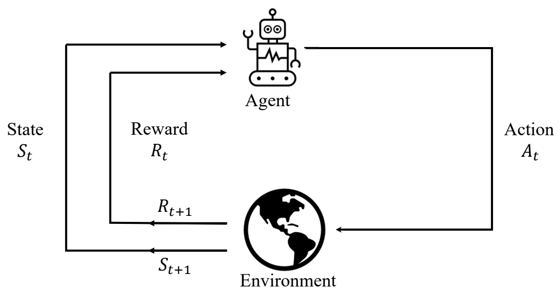

Figure 1: The interaction between the agent and the environment: the agent observes the state S_t and the corresponding reward value R_t at time t. Based on the state and reward information, the agent decides how to take appropriate actions. After the agent executes the action A_t, the environment will give the state S_{t+1} and reward R_{t+1} information of the $t+1$ time step based on the action taken by the agent.

mistakenly update their policies.

To mitigate this issue, decentralized policies can be learned in the centralized training with decentralized execution (CTDE) paradigm (Oliehoek, Spaan, and Vlassis 2008). A typical method in the CTED framework is Value Decomposition, which determines the role of each agent in the joint reward and then somehow isolate its share out of it. So that each agent can get the reward in line with the current task, so as to alleviate the problem of inconsistent task objectives of each agent in the multi-agent system. Value Decomposition Network (VDN) (Sunehag et al. 2017) represents Q_{tot} as a sum of individual Q-values that condition only on individual observations and actions. QMIX(Rashid et al. 2018) employs a network that estimates joint action-values as a non-linear combination of per-agent values. QTRAN (Son et al. 2019) is proposed to guarantee optimal decentralization inheriting the additive assumption while avoiding representation limitations introduced by VDN and QMIX. Qatten (Yang et al. 2020) is a variant of QMIX, which supplements global information through a multi-head attention structure that condition on local observations. However, Qatten cannot achieve a linear complexity for the increase in the number of agents, so the effect will be weakened when the number of agents increases.

In this paper, We further improved the limitations of Qatten, that is, in view of the limitations that it is difficult to deal with multiple agents, I used the attention mechanism to deal with it linearly, and finally applied it to the hybrid network to achieve a far-reaching Higher than the original Qatten effect method, this method is called QattenLinear.

Related Work

Single-agent reinforcement Learning Reinforcement learning (RL) is a type of machine learning where an agent learns to take actions in an environment to maximize some notion of cumulative reward. For decrete action space, in 1988, Sutton et al. proposed Temporal-Difference Learning (TD) and applied it to the MDP model (Sutton and Barto 2018). In 2015, the DeepMind team combined the RL with the DL for the first time, and proposed the DQN (Mnih et al. 2015), which was published in the journal Nature, in which the neural network was used to represent the role of the Q table in Q-Learning (Watkins and Dayan 1992), which solved the problem of "Dimensional disaster", the trained game AI can outperform humans at Atrai games. However, Some researchers (Van Hasselt, Guez, and Silver 2016) found the problem of over-estimation in DQN, and proposed the Double-DQN algorithm to successfully solve the phenomenon of over-estimation. Prioritize replaybuffer (Schaul et al. 2015) significantly improved the convergence of DQN algorithm by assigning priorities to buffer samples. Later, Dueling DQN (Wang et al. 2016) was proposed, which improved the DQN algorithm. The algorithm divides the value of Q into a value function and an advantage function, which speeds up the convergence of the algorithm.

Multi-agent reinforcement learning Multi-agent reinforcement learning (MARL) is a subfield of reinforcement learning that involves multiple agents learning to interact with each other and with their environment in order to accomplish a common goal.In 1998, Kaelbling et al. proposed Partically Observable MDP (POMDP), a Markov process based on local observations, which provided a modeling basis for multi-intelligent reinforcement learning(Oliehoek, Spaan, and Vlassis 2008).Kraemer proposed the CTDE framework , which uses centralized training for distributed execution to train decentralized policies, and Foerster proposed the COMA algorithm (Lowe et al. 2017), which reduces the problem of reputation assignment among agents by means of a counterfactual mechanism. Later, Schroeder pointed out the limitation of the COMA method in evaluating the global Q-values from the joint action state space, and the algorithm becomes inefficient in more complex cases (Schroeder de Witt et al. 2019).Sunehag P proposed the VDN method (Sunehag et al. 2017), which decomposes the global Q-values into Q-values at the individual level, using a simple summation approach. Kyunghwan Son proposed the QTRAN method (Son et al. 2019), which uses global information to train the joint network of intelligences, and then further trains the individual network through the joint network of intelligences. Qatten(Yang et al. 2020) mainly proposes a practical multi-head attention based q-value hybrid network (Qatten) to approximate the global q-value. Qatten leverages key-value memory operations to explicitly measure the importance of each individual to the global system, thereby transforming individual Q_i into Q_{tot} in a multi-head attention structure. QattenLinear combines the multi-head linear attention mechanism to modify Qatten, and the attention mechanism is processed linearly, and finally applied to the hybrid network, so as to achieve a method that is much higher than the original Qatten effect.

Background

DEC-POMDP A fully cooperative multi-agent task can be described as a Dec-POMDP consisting of a tuple $G =< S, U, P, r, Z, O, n, \gamma >$. $s \in S$ describes the true state of the environment. At each time step, each agent $a \in A \equiv 1, \cdots, n$ chooses an action $u^a \in U$, forming a joint action $u \in U \equiv U^n$. This causes a transition on the environment according to the state transition function $P(s_0|s, u) : S \times U \times S \rightarrow [0, 1]$. All agents share the same reward function $r(s, u) : S \times U \rightarrow R$ and $\gamma \in [0, 1)$ is a discount factor.

Attention mechanism The attention mechanism is derived from the bionics of the human visual attention mechanism. A method of data processing that was born. When calculating the specific attention matrix A, we usually need three important components, which are: 1.$Query(Q)$ 2.$Key(K)$3.$Value(V)$,$Q, K, V \in R^{N\times d}$. Q, K, V are calculated as follows:

$$Q = xW_Q, \quad K = xW_K, \quad V = xW_V$$

After getting the variables of Q, K, and V, multiply Q and K to calculate the weight coefficient:

$$w_i = softmax\left(\frac{Q_i K^T}{\sqrt{d_k}}\right)$$

$$\text{where, } softmax(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e_{z_k}} (j = 1, \cdots, K)$$

Finally, after matrix multiplication of the attention coefficient vector w_i and the V feature, the new features of the self-attention network are obtained:

$$s_i^{new} = w_i V$$

.

Linear attention mechanism For the self-attention mechanism in the original Transformer, when it is assumed that the initial processing obtains $Q, K, V \in R^{Nd}$. The time complexity of solving the attention matrix A is $O(N^2)$, and then the weighted average is obtained Its complexity is $O(N^{2d})$, and the general situation is that the length N of the text we input is much larger than the feature dimension d, so the complexity can be simplified to $O(N^2)$, as shown in (a) of Figure 2.

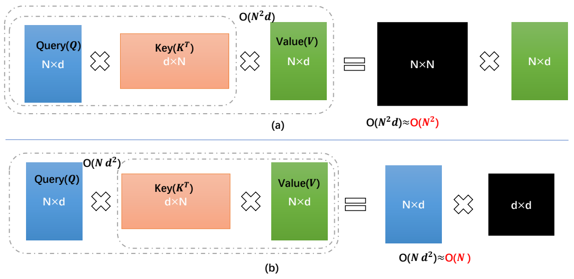

Figure 2: Figure (a) represents the original attention mechanism, and figure (b) represents the self-attention mechanism for linearization. Usually, the amount of data N we process is far greater than d, that is, $N \gg d$. Therefore, when the final calculation is complicated, d can be ignored relative to N. So the native self-attention mechanism is $O(N^2)$ time complexity, while the linearized self-attention mechanism is $O(N)$.

We can expand the exponential form according to Taylor's formula:

$$e^{\boldsymbol{q}_i^T \boldsymbol{k}_j} \approx 1 + \boldsymbol{q}_i^T \boldsymbol{k}_j$$

Then the calculation of the attention mechanism can be written as follows:

$$D(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})_i = \frac{\sum_{j=1}^{N} \left(1 + \left(\frac{\boldsymbol{q}_i}{\|\boldsymbol{q}_i\|_2}\right)^T \left(\frac{\boldsymbol{k}_j}{\|\boldsymbol{k}_j\|_2}\right)\right) \boldsymbol{v}_j}{\sum_{j=1}^{N} \left(1 + \left(\frac{\boldsymbol{q}_i}{\|\boldsymbol{q}_i\|_2}\right)^T \left(\frac{\boldsymbol{k}_j}{\|\boldsymbol{k}_j\|_2}\right)\right)},$$

and simplified as:

$$D(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})_i = \frac{\sum_{j=1}^{N} \boldsymbol{v}_j + \left(\frac{\boldsymbol{q}_i}{\|\boldsymbol{q}_i\|_2}\right)^T \sum_{j=1}^{N} \left(\frac{\boldsymbol{k}_j}{\|\boldsymbol{k}_j\|_2}\right) \boldsymbol{v}_j^T}{N + \left(\frac{\boldsymbol{q}_i}{\|\boldsymbol{q}_i\|_2}\right)^T \sum_{j=1}^{N} \left(\frac{\boldsymbol{k}_j}{\|\boldsymbol{k}_j\|_2}\right)}$$

The above equation can be written in vectorized form as:

$$D(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \frac{\sum_j \boldsymbol{V}_{i,j} + \left(\frac{\boldsymbol{Q}}{\|\boldsymbol{Q}\|_2}\right) \left(\left(\frac{\boldsymbol{K}}{\|\boldsymbol{K}\|_2}\right)^T \boldsymbol{V}\right)}{N + \left(\frac{\boldsymbol{Q}}{\|\boldsymbol{Q}\|_2}\right) \sum_j \left(\frac{\boldsymbol{K}}{\|\boldsymbol{K}\|_2}\right)^T_{i,j}}.$$

By changing the calculation order as above, we can reduce the computational complexity to $0(N)$. As shown in (b) of figure 2.

Method

In this section, we propose the QattenLinear, an improved deep Qvalue decomposition network based on Qatten(Yang et al. 2020). According to the proof of the general decomposition form in Qatten, the design of the decomposition network of QattenLinear is designed according to the decomposition form of Eq.(1). Figure 1 illustrates the overall architecture, which consists of agents'recurrent Qvalue networks representing each agent's individual value function Q^i (τ^i,a^i) and the refined attention based value-mixing network to model the agent-level individual impacts while transforming individual Qvalues Q^i s to Q_{tot}. Besides Q^i s, the global information (including s and u^i) is also fed into the attentionbased mixing network.

For each h, in order to implement the inner weighted sum operation, we use the differentiable key-value memory model (Graves, Wayne, and Danihelka 2014)to estimate the coefficients and derive the relations from the individuals to the global. This is different from MAAC(Iqbal and Sha 2019), which uses the self-attention to learn the critic for each agent by selectively paying attention to information from other agents. Instead of performing self-attention among each pair of the agents, we use this mechanism to help the whole system to model each individual agent's impact at a per-agent level. Moreover, different from Qatten, which uses the native attention mechanism to calculate the attention weights, QattenLinear uses linear attention to calculate the attention weights in order to optimize the time complexity from square level to linear level. Specifically, we pass the global state's embedding vector e_s (s) and the individual features' embedding vector e_i (u^i) into the Linear Attention Module to generate the similarity value.

The linear attention module in QattenLinear uses the $elu(\cdot)$ function for each pair of Query, Key and Value vectors. For the above linear attention mechanism, the $elu(\cdot)$ function is selected instead of the above feature mapping function.

For the outer sum over h, we use multiple attention heads to implement the approximations of different orders of partial derivatives. By summing up the head Q-values Q^h from different heads, we get

$$Q_{tot} \approx c(s) + \sum_{h=1}^{H} Q^h, where\ \ Q^h = \sum_{i=1}^{N} \lambda_{i,h} \left(Q^i\right)$$

H is the number of attention heads. Lastly, the first term $c(s)$ could be learned by a neural network with the global state s as the input.

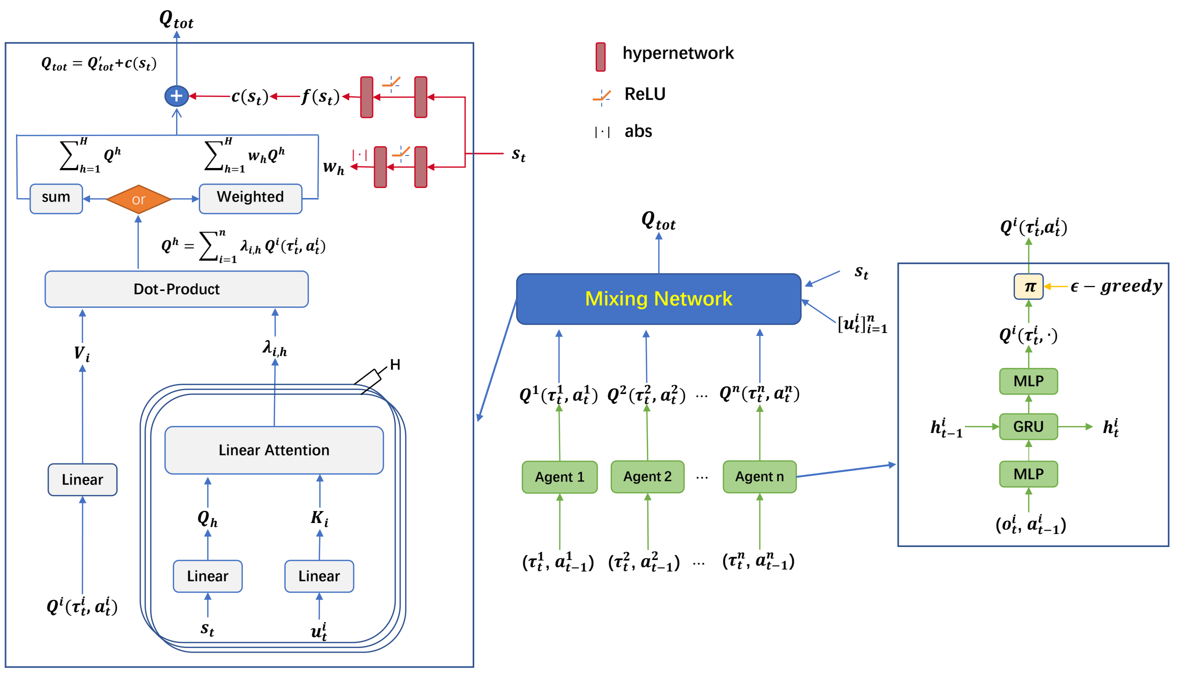

Figure 3: The overall architecture of QattenLinear. The right is agent i's recurrent deep Q-network, which receives the action-observation history record τ^i (last hidden states h^i_{t-1}, current local observations o^i_t and last action a^i_{t-1}). The left is the mixing network of QattenLinear, which mixes $Q^i(\tau^i_t, a^i_t)$ together with s_t and u^i_t. In general, s is global state and u^i is the agent i's individual features like its position.

QattenLinear naturally holds the monotonicity and satisfies the IGM principle

$$\frac{\partial Q_{tot}}{\partial Q^i} \geq 0, \forall i \in \{1, 2, \cdots, N\}$$

Thus, QattenLinear allows tractable maximization of the joint action-value in off-policy learning and guarantees consistency between the centralized and decentralized policies. In previous descriptions, we directly add the Q-value from different heads. However, to relax the weight boundary limitation, we could assign weights w_h for the Q-values from different heads. To guarantee monotonicity, we retrieve these head Q-value weights with an absolute activation function from a two-layer feed-work network f^N, which adjust head Q-values based on global state s.

$$Q_{tot} \approx c(s) + \sum_{h=1}^{H} w_h \sum_{i=1}^{N} \lambda_{i,h} Q^i$$

Experiment

In this section, we evaluate QattenLinear in the StarCraft II decentralized micromanagement tasks(Figure) and use StarCraft Multi-Agent Challenge (SMAC) environment as our testbed, which has become a common-used benchmark for evaluating state-of-the-art MARL approaches. In the SMAC environment, each agent has a field of view, and can only observe teammate information, enemy information, and map information within the field of vision. In this paper, the field of view is set to 9. As shown in the figure 4,

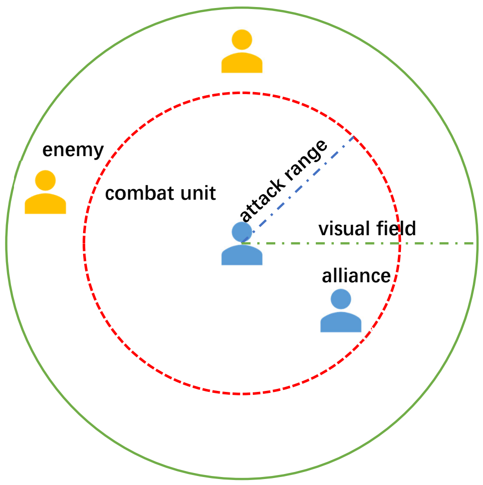

Figure 4: The attack range and field of vision display of the agent

When the teammates are out of the field of view, the agent cannot determine whether the teammates are alive, which is also a big challenge for the agent's behavior selection. The agent can observe the following attributes of enemy units within the field of view: distance, relative x, relative y,

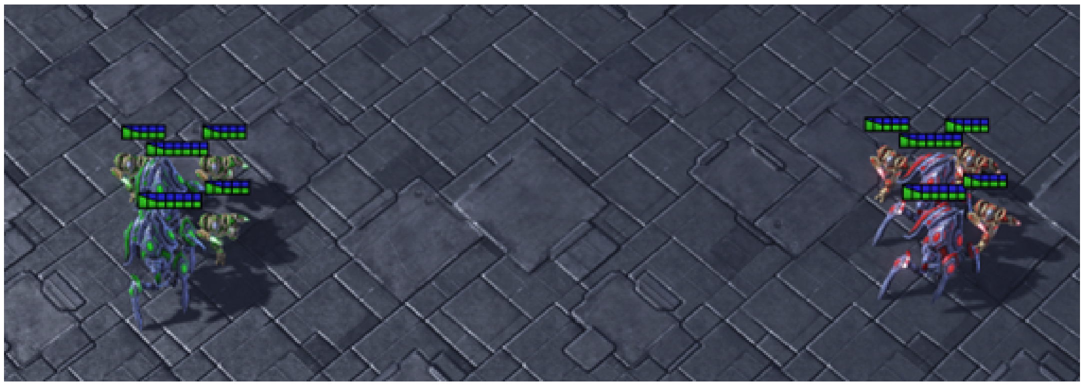

Figure 5: Map 2s3z battle diagram, the smaller unit is Zealots, the number is 3; the larger unit is Stalkers, the number is two. The display on the unit is the blood volume and shield value, the blue is the shield value, and the green is the blood volume value. During the battle, both sides must first destroy the shield to cause damage.

shield, health, unit type. If in a map environment, the combat units are of the same type, then the characteristic of the unit type will be ignored. If the unit has a shield, only by breaking the existing shield of the unit can it cause damage to the enemy. When the unit is not damaged, the shield can be regenerated. In addition to the field of view, the agent also has its own attack range, and can only attack the enemy within its own attack range. The attack range used in this paper is 6. The action space of the agent is discrete, and the actions that can be taken are: move (north, south, east, west), attack, treat (only applicable to the medical transport aircraft Medivac), and do not operate these actions. The agent can only execute the attack command when the enemy is observed within the field of vision and the enemy is within the attack range. When the blood volume of the agent reaches 0, it will fall down, unable to perform any actions, and cannot be detected by friendly forces.

Regarding the setting of the reward value of the agent, at a certain time step, if the trained game agent attacks the enemy unit and the blood volume decreases, the reward value equivalent to the damage caused to the enemy unit can be obtained. When the agent attacks, if it kills the enemy agent, it can get an extra 10 points of reward. If the agent kills all of the enemy units, the game is won and an additional 200 bonus points are awarded. In addition, this paper normalizes all rewards so that the maximum reward value is limited to 20.

In this paper, representative maps such as 2s3z, 8m_vs_9m and 1c3s5z are used for experiments, as shown in Table 1. In order to prevent contingency in experimental training, 5 independent experiments were conducted for each algorithm trained in each map, and finally the median of the 5 results was taken as the final result to avoid outliers during the training process. Training and evaluation schedules such as the testing episode number and training hyper-parameters are kept the same as QMIX in SMAC. The final experimental results are in figure 6.

We can see that our method QattenLinear is better than Qatten on the 2s3z map, and it is comparable to the original Qmix algorithm, and the convergence speed is slightly better than the Qmix algorithm. Of course, for the 2s3z map, there is no big gap because the map itself is very simple, and most methods can produce good results. On the 1s3s5z map, the QattenLinear algorithm is better than the QMIX and Qatten algorithms after 800,000 steps, indicating that the linear attention mechanism can improve the performance of the algorithm. There is an obvious gap on the 8mVS9m map. Our method QattenLinear is much improved than the original Qatten and Qmix, so we can conclude that our method is indeed aimed at multiple agents, and our method has been improved.

Result

In this paper, we study the problems caused by massive agents in Deep-marl and find that previous works did not explicitly consider the impact of excessive number of agents on the whole system when converting individual Q^i to Q_{tot}. Inspired by this, we propose an improved deep Q-value decomposition network based on Qatten called QattenLinear, which allows tractable maximization of the joint action-value in off-policy learning and guarantees consistency between the centralized and decentralized policies. To better derive the relations from the agents to the global, we use the differentiable key-value memory model to estimate the coefficients. Since Qatten using the native attention mechanism to calculate the attention weights, which has high time complexity, we use linear attention to calculate the attention weights in order to optimize the time complexity from square level to linear level. With the extensive experiments, we show that 1) In the final phase, our proposed QattenLinear has better effectiveness than baseline methods on 2s3z and 8mVS9m maps. For 1c3s5z map, our proposed QattenLinear still yields competitive results comparing with baseline methods. 2) In the training phase, our proposed QattenLinear has faster convergence speed than baseline methods on all three maps.

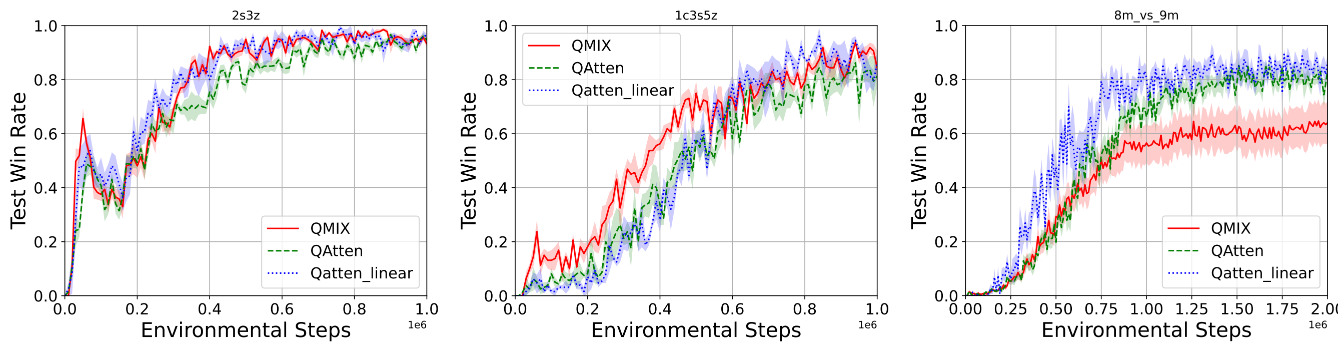

Figure 6: The mean test win rate of the algorithm on the 2s3z, 1c3s5z, 8m_vs_9m maps in the StarCraft II game, where the red curve represents the performance of the qmix algorithm, the blue curve represents the Qatten algorithm based on the attention mechanism, and the green dotted line represents the performance of the QattenLinear algorithm based on linear attention mechanism.

Table 1: Maps in SMAC environment

Name	Ally Units	Enemy Units	Type
3m	3 Marines	3 Marines	Homogeneous
8m	8 Marines	8 Marines	Homogeneous
2s3z	2 Stalkers, 3Zealots	2 Stalkers, 3 Zealots	Heterogeneous

Reference

Wang, J. P.; Shi, Y. K.; Zhang, W. S.; Thomas, I.; and Duan, S. H. 2018. Multitask policy adversarial learning for human-level control with large state spaces. *IEEE Transactions on Industrial Informatics*, 15(4): 2395–2404

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press

Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 330–337

Lowe, R.; Wu, Y. I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30

Oliehoek, F. A.; Spaan, M. T.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32: 289–353

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

Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, 4295–4304. PMLR

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*

Son, K.; Kim, D.; Kang, W. J.; Hostallero, D. E.; and Yi, Y. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, 5887–5896. PMLR

Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; and Zhang, C. 2020. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*

Yang, Y.; Hao, J.; Liao, B.; Shao, K.; Chen, G.; Liu, W.; and Tang, H. 2020. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*, 8(3): 279–292

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*

Schroeder de Witt, C.; Foerster, J.; Farquhar, G.; Torr, P.; Boehmer, W.; and Whiteson, S. 2019. Multi-agent common knowledge reinforcement learning. *Advances in Neural Information Processing Systems*, 32

Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*

Iqbal, S.; and Sha, F. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *International conference on machine learning*, 2961–2970. PMLR