C-MAAC: Cluster-based Multi-Actor Attention Critic for Reinforcement Learning

Hyuncheol Park [*], Taeyeong Lee [†], Yuseung Lee [‡], and Jongjun Park [§]

[1]Department of Computer Science, KAIST
[2]Department of Electrical Engineering, KAIST

Abstract

Multi-Actor-Attention-Critic (MAAC) [4] proposed an efficient form of an actor-critic algorithm for multi-agent environments by integrating it with the attention mechanism from [10]. In this paper, we test MAAC on two MPE environments *Simple Tag* and *Simple Spread*, then propose an extension of this method, *C-MAAC*, by introducing a new component *Cluster Critic* into the original critic pipeline. The experimental results show that our method fails to achieve superior results compared to the original MAAC, but we provide an analysis of the possible ways to improve the performance. Code is available at https://github.com/CS470RL/maac-MPE.

1 Introduction

Reinforcement learning has been a key area of interest in the field of AI in recent years. It has received a spotlight for its wide applicability to many fields, such as decision making, control theory, and simulation-based optimization. However, before a recent breakthrough by *DeepMind*, traditional reinforcement learning methods only focused on single-agent environments. When *DeepMind* [8] proposed a way of applying deep learning techniques in reinforcement learning, it opened a new horizon for researchers to use deep learning on not only single-agent, but also on multi-agent environments. Many subsequent kinds of research have followed, introducing new algorithms for multi-agent reinforcement learning (MARL), such as DDPG [5], MADDPG [6], and MAAC [4].

Challenges in Multi-Agent Environments A multi-agent environment is significantly different from an environment containing only a single agent, and thus can pose new challenges that haven't been encountered before. One example of such challenges is that the environment itself may change depending on the nature of the multiple agents. With a single agent, the environment around the agent can only change in response to the agent's action. But in the case of multi-agents, an action performed by one agent can change the environment from the perspective of another agent. Therefore, a novel approach is required to correctly extend the scope of reinforcement learning to multi-agent settings. Out of the many proposed MARL algorithms, we focused on MAAC [4], which proposes an efficient actor-critic algorithm by incorporating the attention mechanism into the critic module, thus guiding each agent to focus more on certain agents instead of considering every other agent equally. Moreover, in this work, we propose **C-MAAC**, an extension of MAAC derived by introducing an *agent clustering process* in order to promote the teaming behavior of the agents, when given either cooperative or competitive tasks.

2 Related Work

Multi-Agent Reinforcement Learning The emergence of MARL has led to new research in various areas, including *network packet routing* and *coordinating autonomous vehicles*. Early works on MARL mainly suggest simple approaches based on independent learning mechanisms. As a breakthrough, COMA (Counterfactual Multi-Agent Policy Gradients) [3] proposed the idea of sharing sensations, episodes, and learned policies. It uses a **centralized** (i.e. shared) critic to estimate the Q-function, and **decentralized** actors to optimize the agents' individual policies. Since then, many following methods has been suggested, including MAAC [4] which uses the attention mechanism to increase the efficiency of such actor-critic methods.

[*]hy2850@kaist.ac.kr
[†]ooasd990@kaist.ac.kr
[‡]phillip0701@kaist.ac.kr
[§]pjjkey@kaist.ac.kr

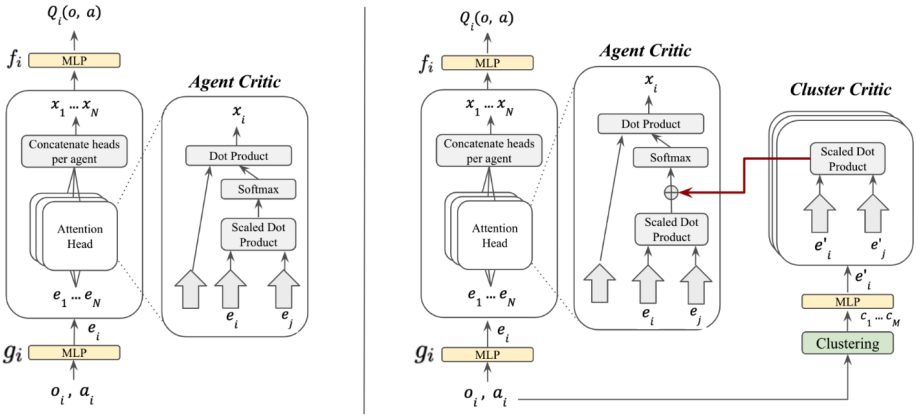

Figure 1: **Critic pipeline of the proposed method.** (Left) Original Attention-based critic architecture [4], (Right) Our proposed cluster-based critic architecture with *Cluster Critic*

Actor-Critic Many reinforcement algorithms previously focused on using solely the value function to choose the next action of an agent (i.e. value-based method). However, since the value function is fixed after training, the model tends to produce deterministic results. To solve this problem, a new method that trains the policy function itself was introduced (i.e. policy-based method). The actor-critic algorithm utilizes both a value function and a policy function as a solution. The **actor** works as a policy to choose the next best action, while the **critic** rates how good the action was by calculating the action score (i.e. Q-value). This method has demonstrated huge success on many RL tasks and all the algorithms used for the experiments in our project are also from the actor-critic family.

Clustering-based Algorithms There were previous attempts to incorporate the concept of clustering in reinforcement learning. One example is [7], in which the authors divide the collected states into clusters based on the similarity in the state features. We also aim to utilize clustering in order to improve an existing RL algorithm.

3 Our Approach

3.1 Multi-Actor-Attention-Critic

Multi-Actor-Attention-Critic (MAAC) [4] applies the attention mechanism onto the actor-critic algorithm for MARL environments. This paper introduces a centralized critic that contains an attention mechanism for calculating **how closely each agent is related to each other**, at a given time step. As the attention values between agents are taken into account when training the policy of each agent, the authors argue that MAAC can lead to better efficiency and scalability for reinforcement learning in very complex multi-agent settings.

The critic module of MAAC is shown in the left figure of fig. 1. For each agent, its observation o_i and action a_i are sent to a centralized critic after each learning step. Then the critic returns the Q-value as

$$Q_i^{\psi}(o, a) = f_i(g_i(o_i, a_i), x_i), \tag{1}$$

where g_i is a single-layer MLP and f_i is a two-layer MLP used as an embedding function before and after the critic, respectively. x_i is the agent's attention value that indicates which agents it should focus more on in the upcoming learning steps. It is calculated as

$$x_i = \sum_{j \neq i} \alpha_j v_j = \sum_{j \neq i} \alpha_j h(V g_j(o_i, a_j)), \tag{2}$$

where V is a linear transformation matrix, h is a leaky ReLU function for bringing nonlinearity, and α_j is the attention weight showing how strongly agent i and agent j are correlated to each other. α_j is derived by using the query-key matching as follows:

$$\alpha_j \propto exp(e_j^T W_k^T W_q e_i) \tag{3}$$

in which $e_i = g_i(o_i, a_i)$ and W_q turns e_i into a *query* and W_k turns e_j into a *key* for the attention mechanism. The matched value is then scaled with respect to the dimensionality of the two matrices and sent through the Softmax function. In order to introduce clustering into this critic pipeline of MAAC, we have modified the above equations for calculating the attention between agents, and the results are explained in the following section.

3.2 Cluster-based MAAC (Our Method)

We focused on **clustering** as it tends to imitate and reflect one of the most basic social behaviors in real life, which is working as a team. In a predator-prey environment with tigers and deer, for example, it is natural for us to assume that the tigers would hunt together in groups, or the deer may also move in herds. Calculating the attention value for each individual agent in MAAC may be insufficient to fully simulate the teaming behavior of agents, since it could return higher rewards for individualistic behaviors. Therefore, we grouped the agents into multiple clusters to represent the **teams** or **group** of agents and also calculated the attention weights between two different clusters. We aimed to stimulate the agents to show more teamwork by adding a cluster-related term into the original MAAC critic module.

Cluster Critic. The critic module of our proposed method, C-MAAC, is shown in the right figure of fig. 1. The Q-value is derived in a similar process as in eq. (1), but uses a modified attention value x_i^* instead of the original x_i:

$$Q_i^{\psi}(o, a) = f_i(g_i(o_i, a_i), x_i^*). \tag{4}$$

x_i^* is calculated similarly to eq. (2) as in MAAC, however we changed the equation for α_j^* so that it not only considers the attention between two *agents*, but also the attention between two *clusters*. For this, we added an additional component in the critic pipeline, ***Cluster Critic***, as shown in fig. 1.

$$x_i^* = \sum_{j \neq i} \alpha_j^* v_j = \sum_{j \neq i} \alpha_j^* h(V g_j(o_i, a_j)) \tag{5}$$

Finally, the value of α_j^* can be expressed as the following equation eq. (6), in which P_i and P_j are modified results of the query-key matching shown in eq. (3) so that α_j^* now influenced by both the agent attention and cluster attention values.

$$\alpha_j^* \propto exp(P_j^T P_i). \tag{6}$$

Here, $P_i = (1 - r) \times W_q e_i + r \times W_q' e_i'$, where r is a preset ratio. $W_q e_i$ is the scaled dot-product between **agents** as in eq. (3), and $W_q' e_i'$ is the scaled dot-product between **clusters**. The two values are combined by the preset ratio r before being passed into the softmax function in the critic, as shown by the red line in fig. 1. After the two values are combined, the remaining pipeline for calculating the Q-value is the same as the original MAAC, but with our new critic we expect the Q-value to imply the cluster relationships as well as the relation between individual agents.

Clustering methods C-MAAC groups the agents into clusters based on the initial environment, and the cluster is maintained throughout the learning steps. We set two criteria in terms of the specific method of clustering the agents:

- **Clustering algorithm:** K-means vs. Random
- **Cluster size:** Equal vs. Varying

The K-means algorithm clusters the agents based on their positions (i.e. (x, y) coordinate), whereas Random means that the agents are randomly grouped. If all clusters have equal size, they must contain the same number of agents, but if the sizes are varying then each cluster may have a different number of agents.
To test the effect of the different choices for clustering methods, we compared the Mean Episode Rewards with the following clustering methods. The experimental results are presented in the **Results** section.

No.	Clst. algorithm	Clst. size
1	K-means	Varying
2	K-means	Equal
3	Random	Equal

Table 1: **Different agent clustering methods.**

Constrained K-means clustering For fixing cluster size, we utilize **Constrained K-means algorithm** [1] Given a database D of m points in R^n, minimum cluster membership values $\tau_h \geq 0$, $h = 1, ..., k$ and cluster centers $C^{1,t}, C^{2,t}, ..., C^{k,t}$ at iteration t, compute $C^{i,t+1}$ in the 2 steps:

- **Cluster Assignment.** Let $T_{i,h}^t$ be a solution to the following linear program with $C^{h,t}$ fixed:

$$\underset{T}{minimize} \sum_{i=1}^{m} \sum_{h=1}^{k} T_{i,h} \cdot (\frac{1}{2} \left\| x^i - C^{h,t} \right\|_2^2)$$

$$\sum_{i=1}^{m} T_{i,h} \geq \tau_h,\ h = 1, ..., k$$

$$subject\ to \sum_{h=1}^{k} T_{i,h} = 1,\ i = 1, ..., m$$

$$T_{i,h} \geq 0,\ i = 1, ..., m,\ h = 1, ..., k.$$

- **Cluster Update.** Update $C^{h,t}$ as follows:

$$C^{h,t+1} = \begin{cases} \frac{\sum_{i=1}^{m} T_{i,h}^{t} x^{i}}{\sum_{i=1}^{m} T_{i,h}^{t}} & if \ \sum_{i=1}^{m} T_{i,h}^{t} > 0 \\ C^{h,t} & otherwise. \end{cases}$$

Combining Agent and Cluster Attention In summary, the attention information between clusters is added with the attention information between agents (See eq. (6)). A detailed process is as follows: for the agents in the same cluster, their observation o_i and action a_i are concatenated into a single matrix to be used as a input for the *Cluster Critic*. Using the attention calculation process from [10], the **scaled dot-products** (not the final attention value) between two clusters are calculated using the query-key matching method. These scaled dot-product result from *Cluster Critic* is sent to the *Agent Critic* and is added to the scaled dot-product of agents (i.e. red line in fig. 1). This process is added in order to enforce the agents within the same cluster to work as a team.

Why use Scaled Dot-Products? We initially aimed to modify the **final attention values** of the agents by combining with the attention values between clusters. However, the attention value of each agent didn't contain clear information about other agents since they have the dimension of *[batch size, hidden dimension]*. Thus it was difficult to accurately understand the meaning of the clustering when each agent did not have clear information on other agents.
Instead, the **scaled dot-product**, which is an intermediate product in the attention calculation process, has the dimension of *[batch size, 1, other agents]*. So it contains the information about other agents in a more clear manner. We could properly modify such scaled dot-product between two agents while considering the attention information between the clusters in this case.

4 Experiments

4.1 Setup

OpenAI GYM Gym [2] is a collection of environments and APIs for reinforcement learning, provided by *OpenAI* to facilitate research on RL. It provides a wide variety of environments that one can choose from and run experiments easily. For this work, we focused on using the *Multi Particle Environments* (MPE) to simulate and test our C-MAAC algorithm. Its simplicity and ease of control seems to be the best for evaluating our novel approach, without having to worry about unexpected side-effects from tweaking the environment settings by too much. For the following experiments, we used the *PettingZoo* library [9], an extension of OpenAI Gym specialized for multi-agent environments.

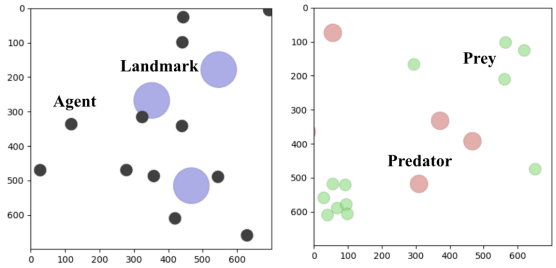

Figure 2: **MPE Environments.** (Left) Simple Spread *(cooperative)*, (Right) Simple Tag *(competitive)*.

Environment We used the two MPE environments shown in fig. 2: **Simple Tag** and **Simple Spread**. Simple Tag [1] is a *competitive* environment, similar to a predator-prey relationship. There are a good agents (*preys*) and adversary agents (*predators*). The preys move faster and receive a negative reward for being hit by an predator, and the reward is given by -10 for each collision. The predators are slower and are rewarded for hitting a prey agent. The reward is given by +10 for each collision. We conducted experiments using Simple Tag with 12 prey agents and 6 predator agents.
Simple Spread [2] is a *cooperative* environment composed of agents and landmarks. Agents must learn to cover all the landmarks while avoiding collisions with other agents. More specifically, all agents are *globally* rewarded based on how far the closest agent is to each landmark. It is calculated as the sum of the minimum distances. Locally, the agents are penalized if they collide with other agents. We conducted the experiments using Simple Spread with 12 agents and 12 landmarks. We conducted each experiments for 50K episodes and 25 steps per episodes for both of the experiments. We ran the model on provided GPU for CS470.

[1] www.pettingzoo.ml/mpe/simple_tag
[2] www.pettingzoo.ml/mpe/simple_spread

Figure 3: **Mean Episode Rewards on a *competitive* environment.** (Left) Reward sum of **predator** agents, (Middle) Reward sum of **prey** agents, (Right) Reward sum of all agents. The specific experiment settings are shown in table 2.

4.2 Baselines

As a baseline, we used the basic MADDPG [6] and MAAC models. The results of the two models in comparison with our C-MAAC are presented in the following section. We use MADDPG and MAAC to verify the effectiveness of the attention mechanism in critics and compare the performance of our method in competitive and cooperative environment.

4.3 Results

Comparison on *Competitive* Environment As shown in fig. 3, there are oscillations in the Mean Episode Rewards for all three experiment settings: only agent attention, agent-cluster mixed attention (main setting for our C-MAAC model) and only cluster attention.

The first notable feature is that the overall tendency of the graph oscillates in an underdamped manner. Since the environment is competitive between the predator agents and the prey agents, they are competing against each other during the learning. Thus, when the prey agents learn how to avoid the predator agents, the amount of negative rewards of the prey agents becomes smaller. Then, the predators learn how to catch the preys and the amount of rewards of the predators becomes increased. This leads to the tendency of oscillation of the total reward.

Second, for the first 5K episodes, the reward values of all models are similar, which implies that attention makes difference in the rewards at least 5K episodes after the learning begins.

Lastly, the reward in the experiment with only the *agent attention* (in fig. 3) shows the highest reward at the end of the episode. However, the model with only clusters and the model with the clusters and the agents shows higher rewards in the middle of the learning. This represents that the period of oscillation becomes diversed by using clustering mechanism.

Comparison on *Cooperative* Environment Intuitively, the reward from the cooperative task is expected to increase gradually, as agents help each other to achieve the goal. However, the result oscillates, not gradually increasing (fig. 4). In competitive environment, it was reasonable to think that the reward oscillates, as good and adversary agents fight over the limited resource. But, it was hard to explain why the same pattern appeared in the cooperative environment too.

In fig. 4, mean episode rewards for the case of no clustering were the highest. The rewards of the clustering case started with low rewards, but increased rapidly after 30k episodes. This suggests that clustering technique may have changed the behavior of the agents even slightly.

When we compared our C-MAAC with the baseline algorithm, MADDPG, in cooperative environment (fig. 6), MADDPG actually showed much higher rewards than any of the MAAC-derived algorithms. Also, it was surprising to see that MADDPG converges very fast compared to MAAC.

Perhaps MADDPG just performs better in cooperative environment.

Comparison on different clustering methods Out of 3 clustering methods, the baseline (clustering without any agent number limits) seems to perform the best (fig. 5). It is clear that the position-based clustering using K-means is better than randomly assigning agents into clusters. Also, constrained K-means stood between vanilla K-means and random clustering method, definitely showing that the vanilla K-means approach (not constraining the number of agents in each cluster) is better.

5 Conclusion

In this paper, we proposed a novel extension of the MAAC algorithm by introducing the concept of agent clustering and added a *Cluster Critic* module

No.	Environment	Algorithm	Critic Type
1	Simple Tag	MADDPG	Only Agent
2	Simple Tag	C-MAAC	Only Agent
3	Simple Tag	C-MAAC	Agent 0.5 + Clst. 0.5
4	Simple Tag	C-MAAC	Only Clst.

Table 2: **Comparisons on a *competitive* environment.** All experiments were ran on the Simple Tag MPE environment from PettingZoo for 50,000 episodes. We set 12 *prey* agents and 6 *predator* agents for Simple Tag.

No.	Environment	Algorithm	Critic Type
1	Simple Spread	C-MAAC	Only Agent
2	Simple Spread	C-MAAC	Agent 0.5 + Clst. 0.5
3	Simple Spread	C-MAAC	Only Clst.

Table 3: **Comparisons on a *cooperative* environment.** All experiments were ran on the Simple Spread MPE environment from PettingZoo for 50,000 episodes. We set 12 agents and 12 landmarks of Simple Spread.

Figure 4: **Mean Episode Rewards on a *cooperative* environment.** The specific experiment settings are shown in table 3.

into the original critic pipeline. We expected that our model, *C-MAAC*, would show better performance than MAAC in MARL environments with a large number of agents, and conducted experiments in two types of MPE environments to compare the performance. But as explained in the **Results** section, our model failed to get better reward graphs than the baseline models, and we have analyzed the possible causes of such shortcomings and suggested the future works we should conduct in order to improve the performance of our cluster-based actor-critic method.

5.1 Limitations and Future Works

Real-time Rendering As all experiments were ran on a remote server through SSH connection, we failed to render the environments and watch the agents move in real-time. Instead, we captured the environment for every 50 episodes (each experiment was ran for total 50,000 episodes) and tried to discover some tendencies in the way that the agents move in different learning settings, and the results are shown in **Appendix**. However, using the real-time rendering would have given us a better insight of how the agents actually behave throughout the learning process The graphs of Mean Episode Rewards are insufficient to understand the specific emergent behaviors of agents in different settings.

Lack of testing environment and time The tests were ran on two MPE environments : Simple Tag and Simple Spread. However, for a more generalized and robust analysis for our model, it should be tested on multiple types of environments with more various scenarios.

Due to limited time and resources, we had to set only a small number of agents, at most 18. We suggest the possibility that clustering may be more advantageous in environment with a larger number of agents. With a small number of agents, there would only be a few clusters which makes the attention less effective. We assume that C-MAAC may use its attention mechanism more effectively for a larger number of clusters, which should be done as a future work.

Also, we trained our model up to 50k episodes, but the value of *Mean Episode Rewards* did not doesn't seem to converge. If we train the model much further to the point where the rewards converge, there could be some difference.

Lack of theoretical background on combining attention information To combine the attention information from the clusters into that from the agents, our team simply added cluster attention logits to agent attention logits with certain ratio. This was done under the intuition that simple addition would suffice to incorporate the effect of clustering (fig. 1). Thus, our attention combining method lacks any mathematical background to support the validity.

6 Appendix

Training Graphs

Rendered Environments OpenAI Gym provides native method to render the actual movements of the agents using built-in 'render' method. However, MAAC algorithm uses parallel environments, making it difficult to play a continuous rendered screen

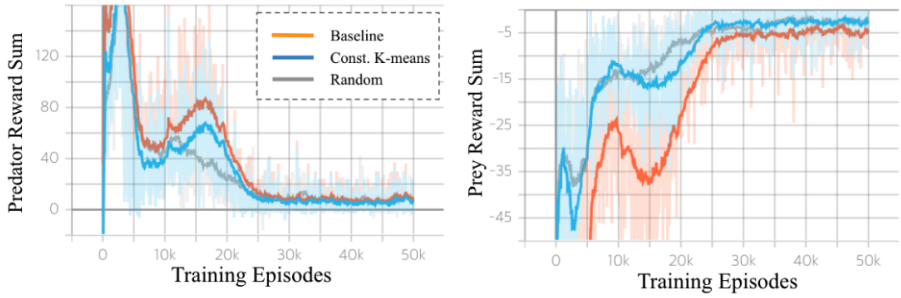

Figure 5: **Mean Episode Rewards with different clustering methods.** (Left) Reward sum of all *predator* agents, (Right) Reward sum of all *prey* agents. The specific experiment settings are shown in table 1.

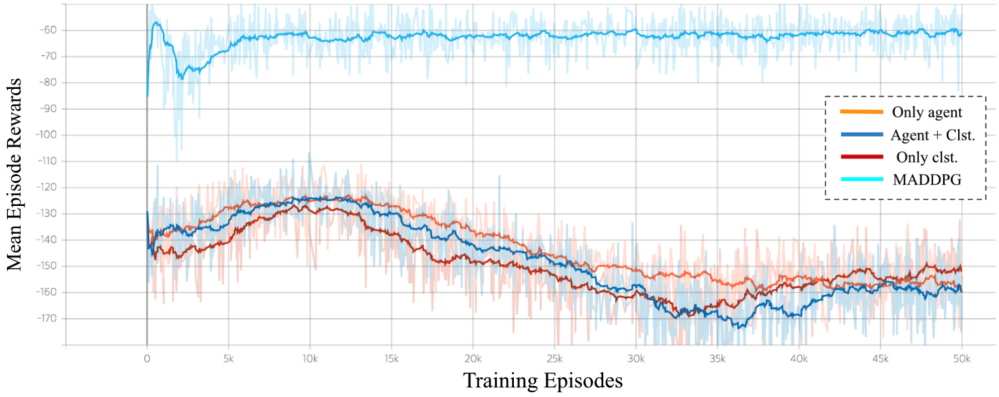

Figure 6: **Mean Episode Rewards on a *cooperative* environment.** The specific experiment settings are shown in table 3.

with the 'render' function. Instead, we captured the scene from one of the environments for every 50 epochs to get the rough picture whether the clusters are in action or not.

Acknowledgements

For our implementation, we used the official MAAC code as a base code, provided by Shariq Iqbal [3]. We would like to thank him for his novel approach towards MARL. All four team members have contributed to the architecture of C-MAAC and the experiments.

References

[1] K.P. Bennett, P.S. Bradley, and A. Demiriz. 2000. Constrained k-means clustering. Technical Report MSR-TR-2000-65.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym.

[3] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual multi-agent policy gradients.

[4] Shariq Iqbal and Fei Sha. 2018. Actor-attention-critic for multi-agent reinforcement learning.

[5] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning.

[6] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments.

[7] Xiao Ma, Shen-Yi Zhao, and Wu-Jun Li. 2019. Clustered reinforcement learning.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning.

[9] J. K. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis Santos, Rodrigo Perez, Caroline Horsch, Clemens Dieffendahl, Niall L. Williams, Yashas Lokesh, and Praveen Ravi. 2020. Pettingzoo: Gym for multi-agent reinforcement learning.

[3] github.com/shariqiqbal2810/MAAC

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.