ACTION SEMANTICS NETWORK: CONSIDERING THE EFFECTS OF ACTIONS IN MULTIAGENT SYSTEMS

Weixun Wang[1] *, Tianpei Yang[1] *, Yong Liu[2] , Jianye Hao[1] ✉, Xiaotian Hao[1], Yujing Hu[3], Yingfeng Chen[3], Changjie Fan[3], Yang Gao[2],

[1]Tianjin University, {wxwang, tpyang, jianye.hao, xiaotianhao}@tju.edu.cn

[2]Nanjing University, lucasliunju@gmail.com, gaoy@nju.edu.cn

[3]NetEase Fuxi AI Lab, {huyujing, chenyingfeng1, fanchangjie}@corp.netease.com

ABSTRACT

In multiagent systems (MASs), each agent makes individual decisions but all of them contribute globally to the system evolution. Learning in MASs is difficult since the selection of actions must take place in the presence of other co-learning agents. Moreover, the environmental stochasticity and uncertainties increase exponentially with the number of agents. A number of previous works borrow various multiagent coordination mechanisms into deep multiagent learning architecture to facilitate multiagent coordination. However, none of them explicitly consider action semantics between agents. In this paper, we propose a novel network architecture, named Action Semantics Network (ASN), that explicitly represents such action semantics between agents. ASN characterizes different actions' influence on other agents using neural networks based on the action semantics between agents. ASN can be easily combined with existing deep reinforcement learning (DRL) algorithms to boost their performance. Experimental results on StarCraft II and Neural MMO show ASN significantly improves the performance of state-of-the-art DRL approaches compared with a number of network architectures.

1 INTRODUCTION

Deep reinforcement learning (DRL) (Sutton & Barto, 2018) has achieved a lot of success at finding optimal policies to address complex control tasks(Mnih et al., 2015; Silver et al., 2017; Lillicrap et al., 2015). However, there also exist a lot of challenges in multiagent systems (MASs) since agents' behaviors are influenced by each other and the environment exhibits more stochasticity and uncertainties(Claus & Boutilier, 1998; Hu et al., 1998; Bu et al., 2008; Rashid et al., 2018; Sen & Weiss, 1999).

Recently, a number of deep multiagent reinforcement learning (MARL) approaches have been proposed to address complex problems in MASs. One major class of works incorporates various multiagent coordination mechanisms into deep multiagent learning architecture(Foerster et al., 2018b;a; Yang et al., 2018; Palmer et al., 2018). Lowe et al. (2017) proposed a centralised actor-critic architecture to address the partial observability in MASs. They also incorporate the idea of joint action learner (JAL)(Littman, 1994) to facilitate multiagent coordination. Later, Foerster et al. (2018b) proposed Counterfactual Multi-Agent Policy Gradients (COMA) which is motivated from the difference reward mechanism(Wolpert & Tumer, 2002) to address the challenges of multi-agent credit assignment. Recently, Yang et al. (2018) proposed applying mean-field theory(Stanley, 1971) to solve large-scale multiagent learning problems. Palmer et al. (2018) extended the idea of leniency(Potter & De Jong, 1994; Panait et al., 2008) to deep MARL and proposed the retroactive temperature decay schedule to address stochastic rewards problems. Another class of works focus on specific network structure design by putting constraints between individual agents' Q values and the global one to facilitate multiagent coordination(Sunehag et al., 2018; Rashid et al., 2018; Sukhbaatar et al., 2016). Sunehag et al. (2018) designed a value-decomposition network (VDN) to learn an optimal linear

* Equal contribution. ✉Corresponding author.

value decomposition from the team reward signal based on the assumption that the joint action-value function for the system can be additively decomposed into value functions across agents. Later, Rashid et al. (2018) assumed that the Q-values of individual agents and the global one are also monotonic, and proposed QMIX by employing a network that estimates joint action-values as a complex non-linear combination of per-agent values that condition only on local observations. However, none of the existing works consider from the perspective that an agent's different actions may have different impacts on other agents, which is a nature property in MASs. In multiagent settings, each agent's action set can be naturally divided into two types of actions: one type contains actions that affect environmental information or its private properties and the other type of actions directly influence other agents. Intuitively, the Q-values of actions of different types should be learned over different input information. Thus this indicates that we can leverage the action semantics information to improve an agent's policy network design toward more efficient multiagent learning.

To this end, we propose a novel network structure, named Action Semantics Network (ASN) to characterize such action semantics for more efficient multiagent coordination. The main contributions of this paper can be summarized as follows. 1) To the best of our knowledge, we are the first to explicitly consider action semantics and design a novel network to extract it to facilitate learning in MASs. 2) ASN can be easily combined with any existing RL models to boost its learning performance. 3) Experimental results* on StarCraft II and Neural MMO(Suarez et al., 2019) show our ASN leads to better performance compared with state-of-the-art approaches in terms of both convergence speed and final performance.

2 BACKGROUND

Stochastic games (SGs)(Littman, 1994) are a natural multiagent extension of Markov decision processes (MDPs), which models the dynamic interactions among multiple agents. Considering the fact that agents may not have access to the complete environmental information, we follow previous work's settings and model the multiagent learning problems as partially observable stochastic games (POSGs)(Hansen et al., 2004).

A *partially observable stochastic game* (POSG) is defined as a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}^1, \cdots, \mathcal{A}^n, \mathcal{T}, \mathcal{R}^1, \cdots, \mathcal{R}^n, \mathcal{O}^1, \cdots, \mathcal{O}^n \rangle$, where $\mathcal{N}$ is the set of agents; $\mathcal{S}$ is the set of states; $\mathcal{A}^i$ is the set of actions available to agent i (the joint action space $\mathcal{A} = \mathcal{A}^1 \times \mathcal{A}^2 \times \cdots \times \mathcal{A}^n$); $\mathcal{T}$ is the transition function that defines transition probabilities between global states: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$; $\mathcal{R}^i$ is the reward function for agent i: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and $\mathcal{O}^i$ is the set of observations for agent i.

Note that a state $s \in \mathcal{S}$ describes the possible configurations of all agents, while each agent i draws a private observation o^i correlated with the state: $\mathcal{S} \mapsto \mathcal{O}^i$, e.g., an agent's observation includes the agent's private information and the relative distance between itself and other agents. Formally, an observation of agent i at step t can be constructed as follows: $o_t^i = \{o_t^{i,env}, m_t^i, o_t^{i,1}, \cdots, o_t^{i,i-1}, o_t^{i,i+1}, \cdots, o_t^{i,n}\}$, where $o_t^{i,env}$ is the observed environmental information, m_t^i is the private property of agent i (e.g., in robotics, m_t^i includes agent i's location, the battery power and the healthy status of each component) and the rest are the observations of agent i on other agents (e.g., in robotics, $o_t^{i,i-1}$ includes the relative location, the exterior of agent $i-1$ that agent i observes). An policy π_i: $\mathcal{O}^i \times \mathcal{A}^i \to [0; 1]$ specifies the probability distribution over the action space of agent i. The goal of agent i is to learn a policy π_i that maximizes the expected return with a discount factor γ: $J = \mathbb{E}_{\pi_i}\left[\sum_{t=0}^{\infty} \gamma^t r_t^i\right]$.

3 THE ACTION SEMANTICS NETWORK ARCHITECTURE

3.1 MOTIVATION

In MASs, multiple agents interact with the environment simultaneously which increases the environmental stochasticity and uncertainties, making it difficult to learn a consistent globally optimal policy for each agent. A number of deep multiagent Reinforcement Learning(MARL) approaches

*More details can be found at https://sites.google.com/view/arn-multiagent, the source code will be released in the future

have been proposed to address such complex problems in MASs by either incorporating various multiagent coordination mechanisms into deep multiagent learning architecture(Foerster et al., 2018b;a; Yang et al., 2018; Palmer et al., 2018) or designing specialized network structures to facilitate multiagent learning(Sunehag et al., 2018; Rashid et al., 2018; Sukhbaatar et al., 2016). However, none of them explicitly consider action semantics, which we believe is a critical factor in multiagent settings. Specifically, each agent's action set can be naturally classified into two types: one type contains actions that affect environmental information or its private properties and the other type of actions directly influence other agents. Therefore, if an agent's action is to attack (or communicate with) one of other agents, the value of performing this action should be explicitly dependent on the agent's perception of its environments and the agent to be attacked (or communicated with), and any additional information is irrelevant and may add noise. We refer to the property that different actions may have different impacts on other agents and should be evaluated differently as action semantics between agents.

As we analyzed, the value estimation of actions of different semantics can be improved by relying on only those relevant information. However, previous works use all information to estimate the Q values of all actions, lack of targeted evaluations. To this end, we propose a new network architecture called Action Semantics Network (ASN) that explicitly considers action semantics between agents to improve the estimation accuracy on different actions. Instead of mixing all agent's information together and then inputting into the network, ASN separates the agent's information according to the action semantics. In this way, ASN can provides a more accurate estimation of each action's value and significantly improve the performance of existing DRL algorithms. Besides, ASN is quite general and can be incorporated into any existing deep MARL frameworks. In the next section, we will describe the ASN structure in detail.

3.2 ASN

Considering the semantic difference of different actions, we classify an agent's action set $\mathcal{A}^i$ of agent i into two subsets: $\mathcal{A}^i_{in}$, $\mathcal{A}^i_{out}$. $\mathcal{A}^i_{in}$ contains actions that affect the environmental information or its private properties and do not influence other agents directly, e.g., moving to different destinations would only affect its own location information. $\mathcal{A}^i_{out}$ corresponds to those actions that directly influence some of other agents, e.g., attack agent j in competitive settings, communicate with agent j in cooperative settings.

Following the above classification, the proposed network architecture, ASN, explicitly considers different influence of an agent's actions on other agents by separating the information stream through different isolated networks (shown in Figure 1). Considering an agent i and $n-1$ agents in its neighborhood, ASN decouples agent i's network as follows. The left part shown in Figure 1 contains a network EN^i which is used to generate the state embedding $\boldsymbol{e}^i$ and a network $E2A^i$ (embedding to action) which generates the values of all action in $\mathcal{A}^i_{in}$ as output. The right part is used to estimate the values of those actions in $\mathcal{A}^i_{out}$ related with each influenced agent, composed of $n-1$ subnetworks ($EN^{i,j}, j \in \mathcal{N}, j \neq i$) which are responsible for determining the state embeddings related with each influenced agent. The input of EN^i is the agent i's observation o^i_t, and each of other $n-1$ subnetworks takes one neighbor agent j's information $o^{i,j}_t$ (which denotes the observation of agent i on its neighbor agent j and is a part of o^i_t) as the input.

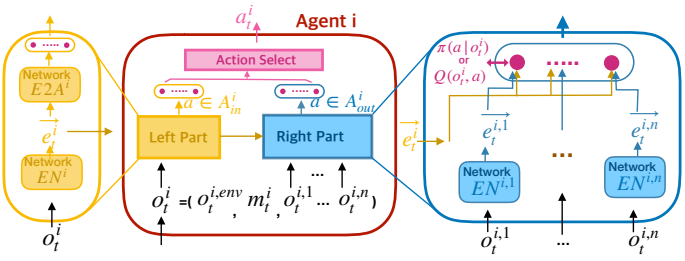


Figure 1: Action Semantics Network Architecture

At each step t, the output of EN^i is the embedding of agent i, denoted as $\boldsymbol{e}^i_t$. The evaluation of executing each action $a^i_t \in \mathcal{A}^i_{in}$ is $Q(o^i_t, a^i_t) = fa(\boldsymbol{e}^i_t, a^i_t)$, where $fa(\boldsymbol{e}^i_t, a^i_t)$ is one of the output of the $E2A^i$ network corresponding to a^i_t. The output of $EN^{i,j}$ is the embedding of agent i on agent j, denoted as $\boldsymbol{e}^{\boldsymbol{i,j}}_{\boldsymbol{t}}$. To evaluate the performance of executing an action $a^{i,j}_t \in \mathcal{A}^i_{out}$ on any of another agent j, ASN combines the output of two subnetworks using a pairwise interaction function

$\mathcal{M}$ (e.g., inner product):

$$Q(o_t^i, a_t^{i,j}) = \mathcal{M}(\boldsymbol{e}_t^i, \boldsymbol{e}_t^{i,j}) \tag{1}$$

then agent i selects the action : $a_t^i = \arg\max_{a_t^i \in A^i} \{Q(o_t^i, a_t^i)\}$.

Similarly, if the policy is directly optimized through policy based RL methods, the probability of choosing each action is proportional to the output of each subnetwork: $\pi(a_t^i|o_t^i) \propto \exp(fa(\boldsymbol{e}_t^i, a_t^i)), \pi(a_t^{i,j}|o_t^i) \propto \exp(\mathcal{M}(\boldsymbol{e}_t^i, \boldsymbol{e}_t^{i,j}))$. Then agent i selects an action following π^i:

$$\pi(a_t^i|o_t^i) = \frac{\exp(fa(\boldsymbol{e}_t^i, a_t^i))}{Z^{\pi_i}(o_t^i)}, \pi(a_t^{i,j}|o_t^i) = \frac{\exp(\mathcal{M}(\boldsymbol{e}_t^i, \boldsymbol{e}_t^{i,j}))}{Z^{\pi_i}(o_t^i)} \tag{2}$$

where $Z^{\pi_i}(o_t^i)$ is the partition function that normalizes the distribution. Here we only consider the case that an action $a^{i,j}$ directly influences one particular agent j. In general, there may exist multiple actions directly influencing one particular agent which is detailed in Section 3.3(Multi-action ASN).

3.3 ASN-MARL

Next we describe how ASN can be incorporated into existing deep MARL, which can be classified into two paradigms: Independent Learner (IL) (Mnih et al., 2015; Schulman et al., 2017) and Joint Action Learner (JAL) (Lowe et al., 2017; Rashid et al., 2018; Foerster et al., 2018b). IL applies a single-agent learning algorithm to a multiagent domain to treat other agents as part of the environment. In contrast, JALs observe the actions of other agents, and optimize the policy for each joint action. Following the above two paradigms, we propose two classes of ASN-based MARL: ASN-IL and ASN-JAL. For ASN-IL, we focus on the case of combing ASN with PPO(Schulman et al., 2017), a popular single-agent policy-based RL. The way ASN combines with other single-agent RL is similar. In contrast, ASN-JAL describes existing deep MARL approaches combined with ASN, e.g., QMIX(Rashid et al., 2018) and VDN(Sunehag et al., 2018).

ASN-PPO In PPO, each agent i is equipped with a policy network parameterized by θ^i. ASN-PPO replaces the vanilla policy network architecture with ASN and optimizes the policy following PPO.

Generally, policy gradient methods optimize the expected return $J(\theta_i) = \mathbb{E}_{\pi_{\theta^i}}\left[\sum_{t=0}^{\infty} \gamma^t r_t^i\right]$ using the policy gradient theorem: $\nabla_{\theta^i} J(\theta^i) = \mathbb{E}_t\left[\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|o_t^i) A_t(o_t^i, a_t^i)\right]$, where A_t is the advantage function. PPO uses constraints and advantage estimation to reformulate the optimization problem as:

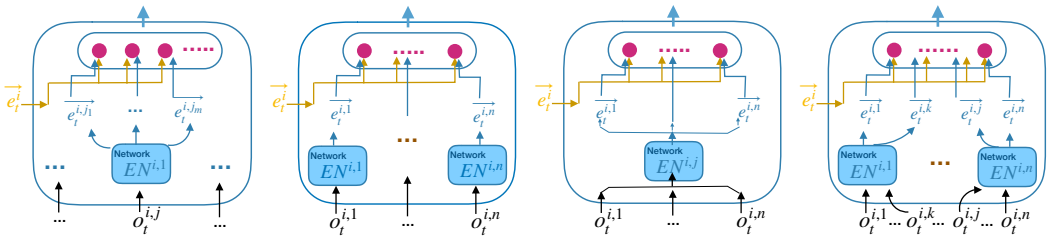
$$\max_{\theta^i} \mathbb{E}_t\left[\frac{\pi_{\theta^i}(a_t^i|o_t^i)}{\pi_{\theta^i_{old}}(a_t^i|o_t^i)} A_t(o_t^i, a_t^i)\right] \tag{3}$$

where θ^i_{old} is the policy parameters before the update. Here, let $r_t(\theta^i)$ denote the probability ratio $\frac{\pi_{\theta^i}(a_t^i|o_t^i)}{\pi_{\theta^i_{old}}(a_t^i|o_t^i)}$, then in ASN-PPO, $r_t(\theta^i)$ can be rewritten as follows by substituting Equation 2:

$$r_t(\theta^i) = \begin{cases} \frac{\exp(fa(\boldsymbol{e}_t^i, a_t^i; \theta^i))}{\exp(fa(\boldsymbol{e}_t^i, a_t^i; \theta^i_{old}))} \frac{Z^{\pi_i}(o_t^i; \theta^i_{old})}{Z^{\pi_i}(o_t^i; \theta^i)} & \text{if } a_t^i \in \mathcal{A}_{in}^i \\ \frac{\exp(\mathcal{M}(\boldsymbol{e}_t^i, \boldsymbol{e}_t^{i,j}; \theta^i))}{\exp(\mathcal{M}(\boldsymbol{e}_t^i, \boldsymbol{e}_t^{i,j}; \theta^i_{old}))} \frac{Z^{\pi_i}(o_t^i; \theta^i_{old})}{Z^{\pi_i}(o_t^i; \theta^i)} & \text{if } a_t^i \in \mathcal{A}_{out}^i \end{cases} \tag{4}$$

Lastly, ASN-PPO maximizes the objective (Equation 3) following PPO during each iteration.

ASN-QMIX The way ASN combines with deep MARL algorithms is similar and we take QMIX(Rashid et al., 2018) as an example which is detailed as follows. Figure 2 illustrates the ASN-QMIX network structure, where for each agent i, ASN-QMIX replaces the vanilla Q-network architecture with ASN. At each step t, the individual Q-function $Q(o_t^i, a_t^i)$ is first calculated following Section 3.2 and then is input into the mixing network. The mixing network mixes the output of all agents' networks monotonically and produces the joint action-value function $Q_{tot}(s_t, a_t)$. The weights of the mixing network are restricted to be non-negative and produced by separate hypernetworks (shown in Figure 2), each of which takes state s_t as input and generates the weights of one layer of the mixing network. Finally, ASN-QMIX is trained to minimize the loss: $L(\theta) = \sum_{i=1}^{b}\left[(y_t^{tot} - Q_{tot}(s, \boldsymbol{a}; \theta))^2\right]$, where b is the batch size of transitions, $y_t^{tot} = r_t + \gamma \max_{\boldsymbol{a'}} Q_{tot}(s', \boldsymbol{a'}; \theta^-)$, and θ^- is the parameters of the target network as in DQN(Mnih et al., 2015).



(a) Multi-action ASN (b) Basic ASN (c) Homogeneous ASN (d) Mixed ASN

Figure 3: Different variants of ASN.

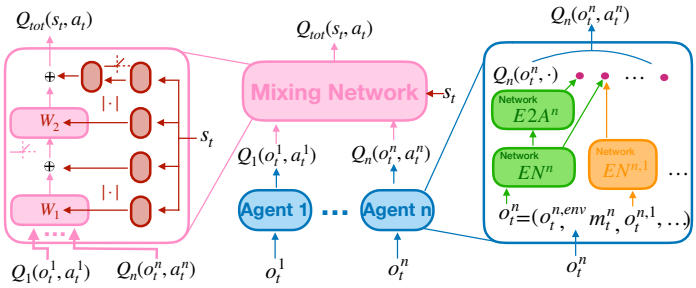


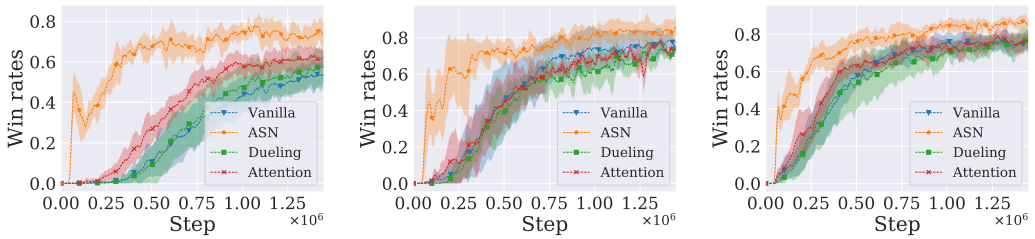
Figure 2: QMIX-ASN

Multi-action ASN The general case in MASs is that an agent may have different influence on other agents through choosing different actions, e.g., a soldier can select different weapons to attack enemies and cause different damages. Motivated by this, we extend the basic ASN to a generalized version, named Multi-action ASN (shown in Figure 3 (a)), that takes $o^{i,j}$ as input, and produces a number of embeddings $\boldsymbol{e}^{\boldsymbol{i},\boldsymbol{j_1}}, \cdots, \boldsymbol{e}^{\boldsymbol{i},\boldsymbol{j_m}}$, where m is the number of actions that directly influences agent j. After that, multi-action ASN calculate the Q-function of each action, which uses a pairwise interaction function $\mathcal{M}$ to combine the two embeddings $\boldsymbol{e}^{\boldsymbol{i},\boldsymbol{j_k},k\in[1,m]}$ and $\boldsymbol{e}^{\boldsymbol{i}}$ following Equation (1).

Parameter Sharing Parameter Sharing (PS) mechanism is widely used in MARL. If agents are homogeneous, their policies can be trained more efficiently using PS which greatly reduces the training complexity(Gupta et al., 2017). Recent work(Rashid et al., 2018) also incorporates PS on heterogeneous agents by adding extra information to identify each type. Following previous work, here we incorporate PS into ASN. The right part of ASN (Figure 1) contains a number of subnetworks, each of which takes $o^{i,j}$ as input (Figure 3 (b)). In this way, the number of subnetworks is equal to the number of agents that an action $a_t^{i,j} \in \mathcal{A}^i_{out}$ has a direct impact on any of another agent j. The training of basic ASN is inefficient with the increase of the number of subnetworks. If the other agents that agent i can directly influence are homogeneous, the subnetwork parameters can be shared across those agents. Thus, in a homogeneous MAS, all influencing agents can share one subnetwork (shown in Figure 3 (c)); in a MAS that contains several types of agents, each type of agents can share one subnetwork (Mixed ASN in Figure 3 (d)). Note that the basic ASN (Figure 3 (b)) can be seen as the simplest case that designs a subnetwork for each influencing agent without PS.

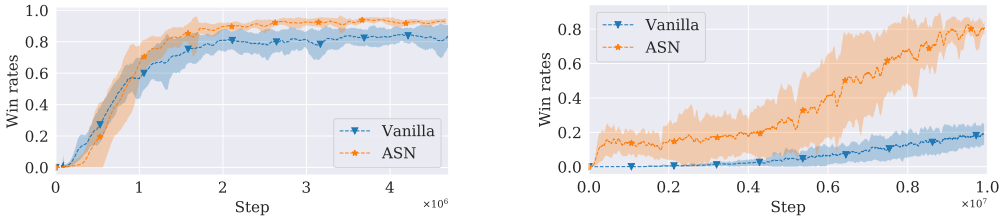
4 Simulations

We evaluate the performance of ASN compared with different network structures including the vanilla network (i.e., aggregate all information and input into one network), the dueling network (Wang et al., 2016) and the attention network (i.e., the vanilla network adds an additional hidden layer to compute the weights of the input and then generate an element-wise product to input into the next layer) under various DRL approaches. Our test domains include StarCraft II (Samvelyan et al., 2019) and Massively Multiplayer Online Role-Playing Games (Neural MMO)(Suarez et al., 2019). The details of neural network structures and parameter settings are in the supplementary material.



(a) IQL (b) QMIX (c) VDN

Figure 4: Win rates of various methods on the StarCraft II 8m map.



(a) 2s3z (b) 15m

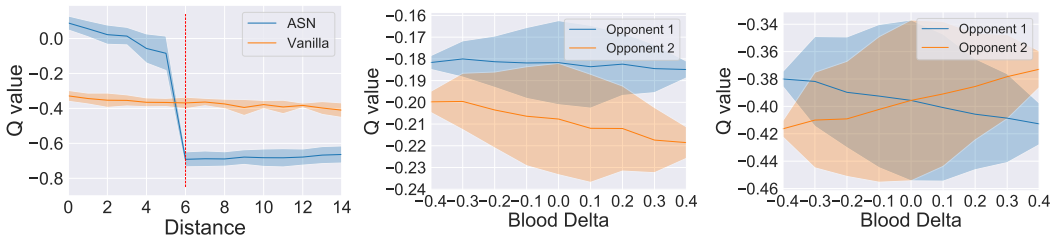
Figure 5: Win rates of ASN-QMIX and vanilla-QMIX on different SC II maps.

4.1 StarCraft II

StarCraft II is a real-time strategy game with one or more humans competing against each other or against a built-in game AI. At each step, each agent observes the local game state (detailed in supplementary materials) and selects one of the following actions: move north, south, east or west, attack one of its enemies, stop and the null action. Agents belonging to the same side receive the same joint reward at each time step that equals to the total damage on the enemy units. Agents also receive a joint reward of 10 points after killing each opponent, and 200 points after killing all opponents. The game ends when all agents on one side die or the time exceeds a fixed period.

Note that previous works (Foerster et al., 2018b; Rashid et al., 2018; Samvelyan et al., 2019) tested on StarCraft II reduce the learning complexity by manually adding a rule that forbids each agent to select an invalid action, e.g., attack an opponent that beyond the attack range and move beyond the grid border. We relax this setting since it requires additional prior knowledge, which is unrealistic in the real world. Thus, the following results are based on the setting that each agent can select an action that causes an invalid effect, and in result, the agent will standstill at the current time step. We also test ASN under previous settings (adding the manual rule in StarCraft II that forbidding the invalid actions) which can be found in the supplementary materials.

In StarCraft II 8m map (8 Marines vs 8 Marines), each agent is homogeneous to each other, so we adopt **Homogeneous** ASN to evaluate whether it can efficiently characterize action semantics between two agents. Figure 4 (a), (b) and (c) show the performance of ASN on an 8m map compared with vanilla, dueling and attention networks under different RL algorithms (IQL(Mnih et al., 2015), QMIX, VDN). We can see that ASN performs best among all of network structures in terms of both convergence rate and average win rates achieved. By separating the observation information into several parts and process each part using different subnetworks, ASN enables an agent to learn the right timing to attack different opponents to maximize its total damage on opponents. In contrast, existing network architectures simply mix all information into one network, thus an agent cannot distinguish the difference of effects that different actions may have on the opponents and may choose the suboptimal opponent to attack, thus resulting in lower performance than ASN. Dueling and attention networks show very similar performance with the vanilla network among all methods, therefore, we only present results of ASN-QMIX compared with the vanilla network under QMIX (denoted as vanilla-QMIX) in the following sections.



(a) Toy scenario 1 (b) Vanilla-QMIX on toy scenario 2 (c) ASN-QMIX on toy scenario 2

Figure 6: The attack action's Q-values of ASN and vanilla under different circumstances.

Next, we consider a more complex scenario: StarCraft II 2S3Z (2 Stalkers and 3 Zealots vs 2 Stalkers and 3 Zealots) which contains two **heterogeneous** groups, each agent inside one group are homogeneous and can evaluate the performance of **Mixed ASN** compared with vanilla-QMIX. From Figure 5 (a) we can observe that Mixed ASN-QMIX perform better than vanilla-QMIX. The reason ASN efficiently identifies action semantics between each type of two agents, thus it selects more proper attack options each time and achieves better performance last vanilla-QMIX.

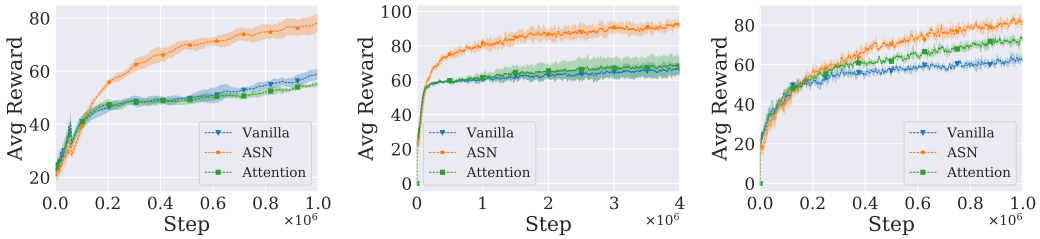
To show the robustness of ASN, we further test on a **large-scale** agent space on a 15m map. Figure 5 (b) depicts the dynamics of the average win rates of ASN-QMIX and vanilla-QMIX. We can see that ASN-QMIX quickly learns the average win rates of approximately 80 %, while vanilla-QMIX fails, with the average win rates of approximately only 20 %. From Figure 4 (b) and 5 (b) we can find that with the increase of the agent number, the margin becomes larger between two methods. Intuitively, ASN enables an agent to explicitly consider more numbers of other agents' information with a larger agent space. However, for an agent using the vanilla network, it is more difficult to identify the action influence on other agents from a larger amount of mixed information, which results in lower average win rates than ASN. An interesting observation for vanilla-QMIX is that they will run away to avoid all being killed, and testing videos can be found in our anonymous website*.

Next we further investigate why ASN performs better. Table 1 presents the average percentages of choosing a valid action for ASN-QMIX and vanilla-QMIX on a 15m map. Note that we remove the manually added rule (which makes the percentage of choosing a valid action to be 100% percent), and agents would probably select the invalid action and standstill, which increases the learning difficulties. We can see that ASN-QMIX achieves an average percentage of approximately 71.9% for choosing a valid action. However, vanilla-QMIX only achieves an average percentage of approximately 44.3%. This phenomenon confirms that ASN effectively exploits action semantics between agents and enables agents to learn which action can be chosen at each time step, facilitating more robust learning, even in large-scale MASs.

Table 1: Percentages of choosing a valid action for ASN-QMIX and vanilla-QMIX.

	ASN	Vanilla
PCT	$71.9 \pm 0.15\%$	$44.3 \pm 0.11\%$

Finally, we investigate whether ASN can efficiently characterize the action semantics and facilitate multiagent coordination. To make the analysis more clear, we test the model learnt on a 15m map on two toy scenarios: 1) a one-on-one combat scenario that the distance between two agents is dynamically changing; 2) a one Marine vs two Marines scenario that the blood volume of two opponents are dynamically different). Figure 6 (a) shows the dynamics of the attack action's Q-value with the distance change of an agent and its opponent. We can observe that the Q-value of the action that the ASN agent attacking its opponent decreases as the distance of the agent and its opponent increases, and stabilizes when the distance exceeds the attack range. However, the vanilla agent keeps the Q-value of the attack action nearly unchanged. Figure 6 (b) and (c) shows the dynamics of the attack action's Q-value of ASN agent and vanilla agent with the two opponent's blood difference changing (i.e., the blood delta equals to the blood volume of opponent 1 minus the blood volume of opponent 2). The ASN agent holds a higher attack action's Q-value on opponent 1 when opponent 1's blood volume is lower than opponent 2 and vice versa. The symmetric curve of



(a) PPO (b) ACKTR (c) A2C

Figure 8: Average rewards of various methods on Neural MMO.

ASN is due to the toy scenario that the state description of two opponents is very similar. However, the vanilla agent always keeps a higher attack action's Q-value on Opponent 1 than on Opponent 2, which means it always selects to attack Opponent 1. These results show that ASN effectively exploits the action semantics between agents and facilitates robust learning among agents.

4.2 NEURAL MMO

The Neural MMO(Suarez et al., 2019) is a massively multiagent environment that defines combat systems for a large number of agents. Figure 7 illustrates a simple Neural MMO scene with two groups of agents on a 10×10 tile. Each group contains 3 agents, each of which starts at any of the tiles, with 100 drops of blood. At each step, each agent loses one drop of blood, observes local game state (detailed in supplementary materials) and decides on an action, i.e., moves one tile (up, right, left, down and stop) or makes an attack using any of three attack options (shown in the left part in Figure 7: "Melee" with the attack distance is 2, the amount of damage is 5; "Range" with the attack distance is 4, the amount of damage is 2; "Mage" with the attack distance is 10, the amount of damage is 1). Each action that causes an invalid effect (e.g., attack an opponent that beyond the attack range and move beyond the grid border) would make the agent standstill. Each agent gets a penalty of -0.1 if the attack fails. The game ends when all agents in one group die or the time exceeds a fixed period, and agents belonging to the same group receive a joint reward, which is the difference of the total blood volumes between itself and its opposite side.

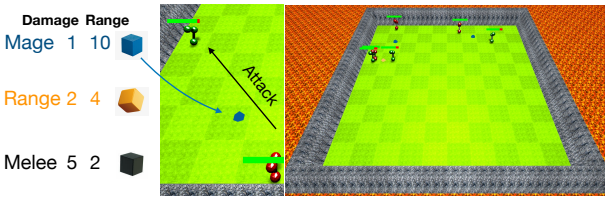


Figure 7: An illustration of Neural MMO.

In Neural MMO, an agent can attack one of its opponent using one of three different attack options, which can be used to evaluate whether **multi-action** ASN can efficiently identify the multiple action semantics between two agents. Figure 8 (a), (b) and (c) present the performance of multi-action ASN on Neural MMO compared with vanilla and attention networks under different IL methods (PPO, ACKTR(Wu et al., 2017) and A2C(Mnih et al., 2016)). We can observe that ASN performs best under all three IL approaches in terms of average rewards. This is because ASN can learn to choose appropriate actions against other agents at different time steps to maximize the damage on others. However, the vanilla network just mixes all information together which makes it difficult to identify and take advantage of the action semantics between agents, thus it achieves lower performance than ASN. Although the attention network computes the weights of the input, the information is mixed initially, it is hard to distinguish which part of the information is more useful. Therefore, it also achieves lower performance than ASN but slightly better than vanilla approaches.

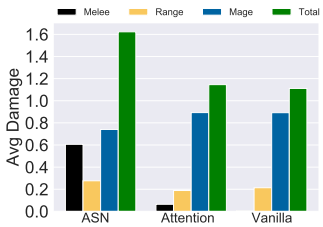


Figure 9: The average damage of choosing each attack under distance $d_{ij} \leq 2$ in Neural MMO.

We further investigate whether ASN can efficiently exploit different action semantics between two agents and enable an agent to identify the best attack option (i.e., an attack that causes the most damage) with the distance between the agent and its opponent changing. Figure 15 shows the average attack damage of each attack option in Neural MMO when the distance between an agent i and its opponent j is less than or equal to 2 ($d_{ij} \leq 2$). The best attack option is "Melee" within this distance range since it causes the maximum damage among three attacks. We can see that the ASN agent causes the highest damage on average. However, the average damage of attention network using "Melee" is approximately 0.05 and the vanilla network never selects the best attack option within such a distance range. This is because the ASN agent has a larger probability to select the best attach option "Melee" than other two networks, thus causes larger total damage. Similar results (when the distance between an agent i and its opponent j is less than or equal to 4 ($d_{ij} \leq 4$) and less than or equal to 10 ($d_{ij} \leq 10$)) can be found in supplementary materials that ASN always causes a higher total damage than two other networks.

5 Conclusion and Future Work

We propose a new network architecture, ASN, to facilitate more efficient multiagent learning by explicitly investigating the action semantics between agents. ASN is the first to characterize the actions semantics in MASs, which can be easily combined with various DRL algorithms to solve complex tasks. ASN greatly improves the performance of state-of-the-art DRL methods compared with a number of network architectures. In this paper, we only consider the direct action influence between any of two agents. As future work, it is worth investigating how to model the action semantics among more than two agents. Another interesting direction is to consider the action semantics between agents in continuous action space.

References

Lucian Bu, Robert Babu, Bart De Schutter, et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998:746–752, 1998.

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130. International Foundation for Autonomous Agents and Multiagent Systems, 2018a.

Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018b.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pp. 66–83. Springer, 2017.

Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pp. 709–715, 2004.

Junling Hu, Michael P Wellman, et al. Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, volume 98, pp. 242–250. Citeseer, 1998.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pp. 6379–6390, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. Lenient multi-agent deep reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 443–451. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Liviu Panait, Karl Tuyls, and Sean Luke. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *Journal of Machine Learning Research*, 9(Mar):423–457, 2008.

Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*, pp. 249–257. Springer, 1994.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4292–4301, 2018.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sandip Sen and Gerhard Weiss. Learning in multiagent systems. *Multiagent systems: A modern approach to distributed artificial intelligence*, pp. 259–298, 1999.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

H Eugene Stanley. *Phase transitions and critical phenomena*. Clarendon Press, Oxford, 1971.

Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents. *arXiv preprint arXiv:1903.00784*, 2019.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pp. 2244–2252, 2016.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2085–2087. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1995–2003, 2016.

David H Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. In *Modeling complexity in economic and social systems*, pp. 355–369. World Scientific, 2002.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pp. 5279–5288, 2017.

Y Yang, R Luo, M Li, M Zhou, W Zhang, and J Wang. Mean field multi-agent reinforcement learning. In *35th International Conference on Machine Learning, ICML 2018*, volume 80, pp. 5571–5580. PMLR, 2018.

A APPENDIX

SUPPLEMENTARY MATERIALS

STATE DESCRIPTION

Neural MMO In a 10x10 tile (where each tile can be set as different kinds, e.g., rocks, grass), there are two teams of agents (green and red), each of which has 3 agents. At the beginning of each episode, each agent appears on any of the 10x10 tiles. The observation of an agent is in the form of a 43-dimensional vector, in which the first 8 dimensions are: time to live, blood volume, remaining foods (set 0), remaining water (set 0), current position (x and y), the amount of damage suffered, frozen state (1 or 0); the rest of 35 dimensions are divided equally to describe the other 5 agents' information. The first 14 dimensions describe the information of 2 teammates, following with the description of 3 opponents' information. Each observed agent's information includes the relative position(x and y), whether it is a teammate(1 or 0), blood volume, remaining foods, remaining water and the frozen state.

Each Agent chooses an action from a set of 14 discrete actions: stop, move left, right, up or down, and three different attacks against one of its opponent ("Melee" with the attack distance is 2, the amount of damage is 5; "Range" with the attack distance is 4, the amount of damage is 2; "Mage" with the attack distance is 10, the amount of damage is 1).

Each agent gets a penalty of -0.1 if the attack fails. They get a -0.01 reward for each ticks and a -10 penalty for being killed. The game ends when a group of agents die or the time exceeds a fixed period, and agents belonging to the same group receive the same reward, which is the difference of the total number of drops of blood between itself and its opposite side.

StarCraft II In StarCraft II, we follow the settings of previous work(Rashid et al., 2018; Samvelyan et al., 2019). The local observation of each agent is drawn within their field of view, which encompasses the circular area of the map surrounding units and has a radius equal to the sight range. Each agent receives as input a vector consisting of the following features for all units in its field of view (both allied and enemy): distance, relative x, relative y and unit type. More details can be found at https://github.com/multiagent-arn/ASN or https://github.com/oxwhirl/smac.

NETWORK STRUCTURE

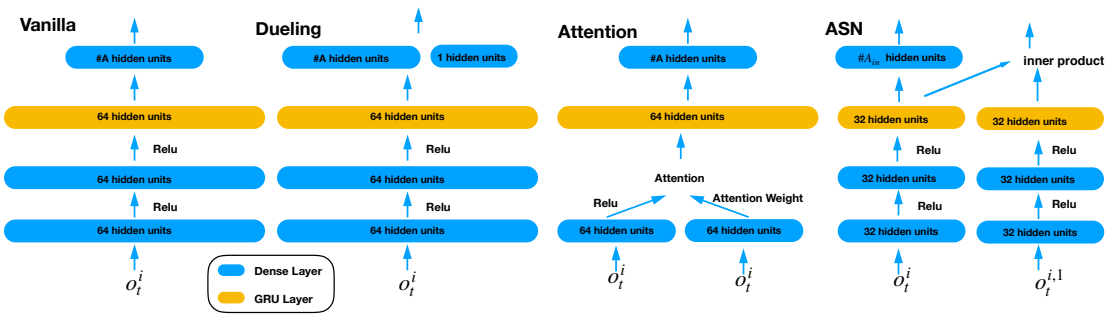


Figure 10: ASN structure on a StartCraft II 8m map.

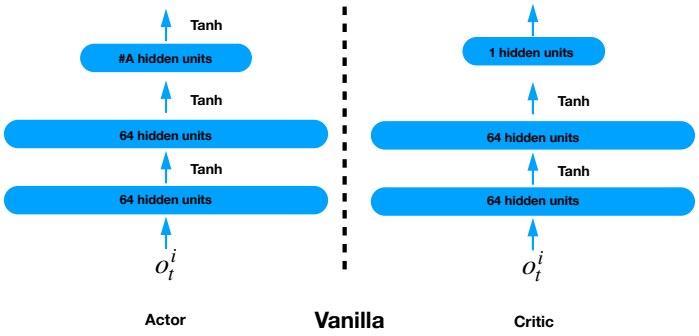


Figure 11: Vanilla structure on Neural MMO.

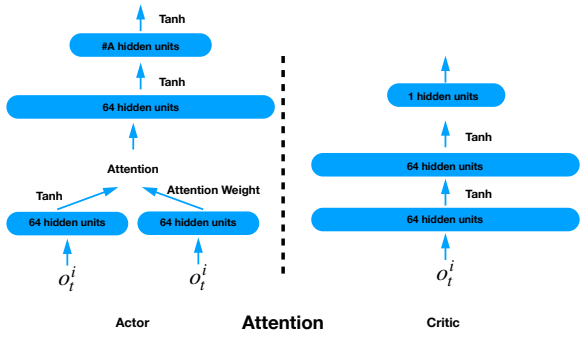


Figure 12: Attention structure on Neural MMO.

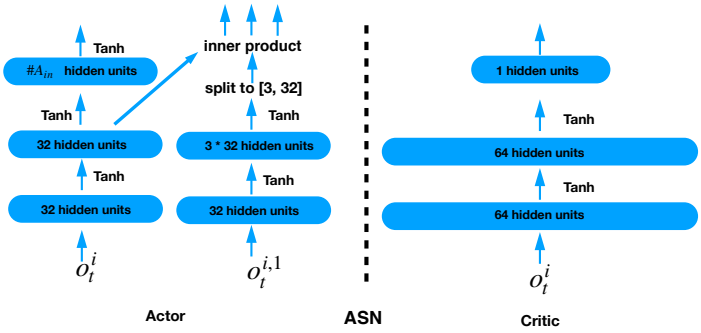


Figure 13: ASN structure on Neural MMO.

PARAMETER SETTINGS

Here we provide the hyperparameters for StarCraft II and Neural MMO[†].

[†]More details can be found at https://github.com/multiagent-arn/ASN

Table 2: Hyperparameters used for StarCraft II.

Hyperparameter	Value
Batch-size	32
Replay memory size	5000
Discount factor(γ)	0.99
Optimizer	RMSProp
Learning rate	$5e-4$
α	0.99
e	$1e-5$
Gradient-norm-clip	10
Action-selector	ϵ-greedy
ϵ-start	1.0
ϵ-finish	0.05
ϵ-anneal-time	50000 step
target-update-interval	200

Table 3: A2C hyperparameters used for Neural MMO.

Hyperparameter	Value
Number of processes	5
Discount factor(γ)	0.99
Optimizer	RMSProp
Learning rate	$7e-4$
α	0.99
e	$1e-5$
Gradient-norm-clip	0.5
Entropy term coefficient	1e-2
Value loss coefficient	0.5
Actor loss coefficient	1

Table 4: ACKTR hyperparameters used for Neural MMO.

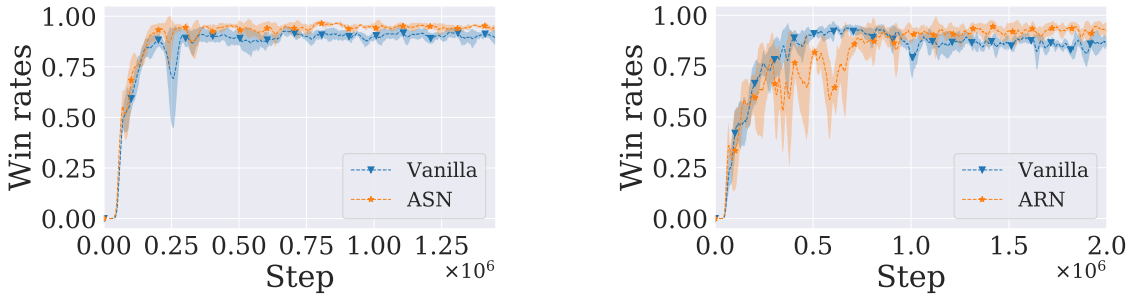
Hyperparameter	Value
Number of processes	5
Discount factor(γ)	0.99
Optimizer	KFACOptimizer
Learning rate	0.25
Momentum	0.9
Stat decay	0.99
KL clip	1e-3
Damping	1e-2
Weight decay	0
Entropy term coefficient	1e-2
Value loss coefficient	0.5
Actor loss coefficient	1

Table 5: PPO hyperparameters used for Neural MMO.

Hyperparameter	Value
Number of processes	1
Discount factor(γ)	0.99
Optimizer	Adam
Learning rate	$7e-4$
e	$1e-5$
Entropy term coefficient	1e-2
Value loss coefficient	0.5
Actor loss coefficient	1

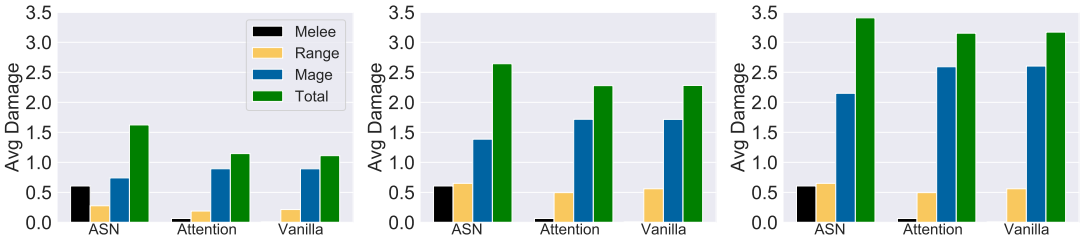
EXPERIMENTAL RESULTS

The following results present the performance of ASN-QMIX and vanilla-QMIX under different StarCraft II maps with adding the manually rule (forbids the agent to choose the invalid actions).



(a) 5m (b) 8m

Figure 14: Win rates of ASN-QMIX and vanilla-QMIX under different StarCraft II maps.



(a) $d_{ij} \leq 2$ (b) $d_{ij} \leq 4$ (c) $d_{ij} \leq 10$

Figure 15: The average probabilities of choosing each attack under different distance d_{ij} in Neural MMO.

The above results present the average attack damage of each attack option under different distance between the agent and its opponent.