Learning to Communicate via Supervised Attentional Message Processing

Zhaoqing Peng
University of Chinese Academy of Sciences
pengzhaoqing16@mails.ucas.ac.cn

Libo Zhang*
Institute of Software Chinese Academy of Sciences
zsmj@hotmail.com

Tiejian Luo
University of Chinese Academy of Sciences
tjluo@ucas.ac.cn

ABSTRACT

Many tasks in AI require the collaboration of multiple agents. Generally, these agents cooperate with each other by message-passing communication. However, agents may suffer from being overwhelmed by massive received messages and have difficulties in obtaining useful information. To this end, we use an attention-based message processing (AMP) method to model agents' interactions by considering the relevance of each received message. To improve the efficiency of learning correct interactions, a supervised variant SAMP is then proposed to directly optimize the attentional weights in AMP with a target auxiliary interaction matrix from the environment. The empirical results demonstrate our proposal outperforms other competing multi-agent methods in "predator-prey-toxin" domain, and prove the superiority of SAMP in correctly guiding the optimization of attentional weights in AMP.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent reinforcement learning**; *Multi-agent systems*; *Intelligent agents*;

KEYWORDS

multi-agent communication, message-passing, attention mechanism, deep reinforcement learning, supervised learning

ACM Reference Format:
Zhaoqing Peng, Libo Zhang, and Tiejian Luo. 2018. Learning to Communicate via Supervised Attentional Message Processing. In *CASA 2018: 31st International Conference on Computer Animation and Social Agents, May 21–23, 2018, Beijing, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3205326.3205346

1 INTRODUCTION

Many real-world problems involve multiple agents with partial observability and limited communication [19], but learning is difficult in these settings due to the partial observability and local view points of agents, which perceive the environment as non-stationary due to teammates' actions [16]. Therefore, the communication between agents is essential for them to extract useful information from others to model the environment.

To learn the communication protocol by agents themselves, most of works apply the reinforcement learning (RL) approach to the multi-agent domain. They embrace deep learning techniques and adopt the message-passing to transmit necessary information between agents. Among them, different methods are used to process the received messages: reference [6] simply concatenates all received messages together and feed to the target network (*e.g.*policy network). This concatenated method suffers from a problem that agents will be overwhelmed by massive received messages if the agent number is large. CommNet [20] tries to obtain an integrated communication vector $\widetilde{m}$ for each agent by averagely pooling over all N messages:

$$\widetilde{m} = \frac{1}{N-1}\sum_{j\neq i} m_j, \tag{1}$$

where m_j denotes the message broadcast from agent j. However, a significant drawback of this representation is not explicitly modeling the interactions and putting the whole communication burden on the message extractor [7]. To model distinct interactions between agents, VAIN [7] introduces an attentional vector a_i to measure the communication strength of agent i, and the interaction between agent i and agent j is modulated by kernel function $e^{|a_i - a_j|^2}$. Thus, the communication vector $\widetilde{m}_i$ for agent i is given by:

$$\widetilde{m}_i = \sum_{j\neq i} Softmax(|a_i - a_j|^2) \times m_j, \tag{2}$$

Although the interactions are able to be modulated, VAIN is less suitable for the case that interactions are not sparse such that the K most important interactions [7].

In this work, we also focus on modeling multi-agent distinct interactions for solving cooperative tasks. To this end, we adopt an *attentional message processing* (AMP) method that is capable of directly modeling the relevance between each received message, and based on AMP, we propose a *supervised attentional message processing* (SAMP) method, which applies supervised learning on the attentional weights in AMP by constructing a *target auxiliary interaction matrix* from the environment. An end-to-end training with deep RL approaches can be performed on AMP to learn coordinated policies, and SAMP is not only compatible with RL approaches but also can be jointly trained with supervised learning approaches.

The benefits of our SAMP are: 1) the interactions are modulated more straightforward by considering the relevance of received messages, which enables each agent to pay more attentions to useful information w.r.t its current observation, 2) the learning efficiency of correct interactions among agents can be dramatically improved

*Libo Zhang is the corresponding author.

CASA 2018, May 21–23, 2018, Beijing, China

ACM ISBN 978-1-4503-6376-1/18/05…$15.00
https://doi.org/10.1145/3205326.3205346

through supervised target auxiliary matrix, which is constructed according to the priori knowledge from the environment 3) each agent is able to integrate more comprehensive information for its decision making by learning the K most relevant received messages.

We test our proposal on a new "predator-prey-toxin" environment and compare it with the prior works in standard environmental settings. The empirical results also demonstrate the superiority of SAMP in coordinating agents' behaviors for solving cooperative tasks. To further interpret the well behaviors of agents in our proposals, we then visualize their attentional weights under a few example states. The result highlights the fact that: agents in AMP are able to extract effective interaction information through giving more attention to useful messages, and the supervised signals in SAMP can correctly guide agents to coordinate more efficiently. These learned interaction patterns illustrate a well established communication protocol by agents themselves. Our proposal has been fully proved and well evaluated to provide as a new solution to the multi-agent communication method for cooperative tasks.

2 RELATED WORKS

Serval works in deep RL have been applied to multi-agent domain such as [5, 10] where multiple independent learners (IL) are applied to control each agent with its own unique learning process. However, those independent learners inevitably face the challenge of not being able to tackle the non-stationary environments since the environment will keep changing for each agent [16]. A recent work [4] addresses this problem by using importance sampling on the experience replay memory to naturally decay obsolete data.

A few notable approaches [11, 18] involve learning the implicit communication of agents by utilizing the actor-critic architecture in RL: COMA [3] also uses a centralised critic to estimate the Q-function but they focus on solving multi-agent credit assignment issues by a counterfactual baseline. BiCNet [18] takes advantage of the bi-directional recurrent neural network (RNN) to unroll the recurrent cell across multiple agents, and the communication information is passed with the hidden states of each recurrent cell.

External communication examples include blackboard, signaling, and message-passing [17]. Some attempts [15] are made to allow sending and receiving signals among agents to coordinate their behaviors, but the signals are hand-craft and pre-determined according to the specific task. In contrast, the recent work DIAL [2] adopts the insights of reinforcing agents to learn communication messages by themselves. Each individual agent learns its policy based on the generated messages of other agents in previous time-step. However, these messages are delayed for one time-step, the environment will keep changing before action is made [11]. Another work [14] has been done to further investigate in learning communication with streams of abstract discrete symbols uttered by agents over time.

The most close work CommNet [20] uses a single network in the multi-agent setting through passing the averaged message over the agent modules between layers. However, since all the agents share almost the same message information, it may lack the ability of handling heterogeneous agent types [7]. By contrast, VAIN [7] extends the attentional mechanism to CommNet by allowing the different strength of interactions among agents. In this work, we also adopts the same attentional settings but we introduce a target interaction matrix to guide the optimization of learning communication.

2.1 Preliminaries

We briefly review the standard single-agent RL, which is formulated as a finite Markov Decision Process (MDP) over a number of discrete time steps. Specifically, the MDP is defined as a four-tuple $\langle S, A, T, R\rangle$, where S is a set of environment states and A is a set of agent actions. The transition function $T(s,a) : S \times A \rightarrow S$ indicates that the environment transmits from state $s \in S$ to a new state $s' = T(s,a) \in S$ after an agent taking action $a \in A$ according to the agent's policy $\pi(a|s)$. The reward function $R(s,a) : S \times A \rightarrow \mathbb{R}$ returns the immediate reward $R(s,a) \in \mathbb{R}$ to the agent after performing action a in state s.

In RL, the goal of agent is to maximize, at each time-step t, the accumulated discounted return R_t for each state s and action a :

$$R_t = \sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k})|s_t, a_t = s, a, \tag{3}$$

where $\gamma \in [0,1)$ is the discount factor. The action value $Q^\pi(s,a) = \mathbb{E}_\pi[R_t|s_t, a_t = s, a]$ is the expected return for selecting action a in state s according to policy π. Similarly, the value of state s under the policy π is defined as $V(s) = \mathbb{E}_\pi[R_t|s_t = s]$.

Notably, there exists at least one optimal policy π^* that gives the optimal action value $Q^{\pi^*}(s,a) = \max_\pi Q^\pi(s,a)$. In practice, it is not feasible to directly mapping from state-action pairs to the corresponding Q-value due to the huge searching space of large states and actions. In model-free RL methods, the recent deep Q-networks (DQN) [13] introduces the neural network to approximate optimal action-value function $Q(s,a;\theta)$ with parameter configuration θ, *i.e.*, $Q^{\pi^*}(s,a) \approx Q(s,a;\theta)$. At each time step t, the parameters θ are optimized by minimizing the loss function:

$$L_t(\theta_t) = \mathbb{E}(y - Q(s,a;\theta_t))^2, \tag{4}$$

where $y = R_t(s,a) + \gamma \max_{a'} Q(s', a'; \theta_{t-1})$, s' is the next state transmitted from state s, and a' is the action of the next state.

The policy gradient [21] method in RL directly parameterizes the policy $\pi(a|s;\theta)$ and updates the parameters θ in the direction of $\nabla_\theta \log \pi(a_t|s_t;\theta)R_t$. To reduce the variance of the estimated direction, we usually subtract a relative baseline $b_t(s_t)$ to obtain the advantage of action a_t in the current state s_t. In [12], the advantage of action a_t in state s_t can be defined as $A(a_t, s_t) = Q(a_t, s_t) - V(s_t)$ and the resulting gradient becomes $\nabla_\theta \log \pi(a_t|s_t;\theta)A(s_t, a_t)$. This approach can be achieved by actor-critic methods in RL where the policy π is produced by the actor and the baseline $b_t(s_t)$ is generated by the critic.

3 METHOD

We consider a partially observable and fully cooperative game with N agents as Figure 1(b) shows: each agent i observes its local state o_i^t at the time-step t and produces a message m_i^t by its message extractor $m_i^t = \mathcal{M}_i(o_i^t; \phi_i)$ with parameters ϕ_i. The communication occurs when each agent shares its message with the others, and we denote the set of all generated messages as $\mathbf{m}^t = \{m_1^t, m_2^t, ..., m_N^t\}$ where each $m_i^t \in \mathbb{R}^l$ consists of l continuous values. Let $\mathbf{m}_{-\mathbf{i}}^{\mathbf{t}}$ be the received message set for agent i, which is a subset of $\mathbf{m}^{\mathbf{t}}$ except for

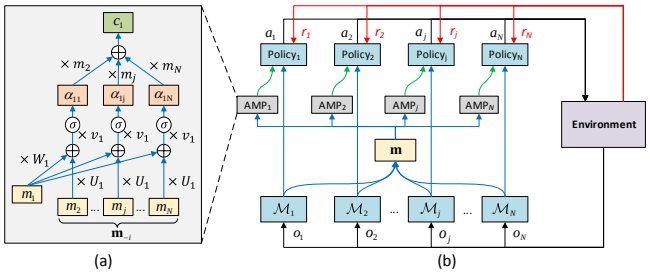

Figure 1: AMP method for agent 1 (a) and multi-agent communication framework (b).

m_i^t. Each agent chooses the action a_i^t with its own policy according to its observation and received messages. After agents perform their actions to the environment, each agent receives a local reward $r_i^t \in \mathbb{R}$. The objective for each agent is to maximize its own expected return $R_i = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_i^t]$ where $\gamma \in [0, 1)$ is the discount factor.

The key to cooperative tasks is that: agents should learn to obtain more information from the received messages to know about the other agents' states, and then coordinate their behaviors to earn the rewards. Since not all received messages are useful for an agent's decision making under its observation, learning to extract useful and relevant messages is essential while agents communicating.

In this paper, we first use an attention-based message processing method (AMP) to reinforce each agent to find the relevant received messages w.r.t its current observation. Based on AMP, we then provide a supervised variant SAMP to enable agents learn interactions more efficiently by providing a target auxiliary interaction matrix from the environment.

3.1 Attention-based Message Processing

For each agent, AMP transforms the received messages $\mathbf{m}_{-i}^t$ to an integrated message vector c_i^t, which represents the state information of other agents. Our goal is to make c_i^t contain as much as relevant and useful message from $\mathbf{m}_{-i}^t$ under the observation o_i. To measure the relevance of messages, we introduce an additive attention weight α_{ij} [1] for each received message m_j, and as Figure 1(a) illustrates, our c_i^t then becomes the weighted sum of all these received messages. In principle, each message m_i can be somehow regarded as a hidden representation of observation o_i, thus we use m_i to indirectly describe the current observation o_i.

In specific, an energy score e_{ij} is modeled by an alignment function a_i to evaluate how relevant between its own message m_i and received message m_j:

$$e_{ij} = a_i(m_i, m_j) = v_i \tanh(W_i m_i + U_i m_j), \tag{5}$$

where the W_i, U_i, v_i are actually learned parameters of AMP. Then, the attentional weight α_{ij} of each m_j can be measured by the sigmoid non-linear activation of e_{ij}:

$$\alpha_{ij} = \sigma(e_{ij}), \tag{6}$$

Note that sigmoid activation is identical to our method since it can release the normalisation restriction compared with Softamax operation in traditional attention mechanism. In this case, agents are free to learn K most relevant received message w.r.t its own produced message. Our integrated message vector c_i is then computed by the weighted sum of all received messages:

$$c_i = \sum_{j \neq i} \alpha_{ij} m_j \tag{7}$$

The simple way of optimizing the attentional weights α is to jointly train them with all the other components, and get updated by the gradients derived from RL. But in order to learn the attentional weights more efficiently and accurately, an auxiliary supervised signal from the environment can be constructed for directly learning attentional weights in the next section.

3.2 Supervised Learning with Target Auxiliary Interaction Matrix

Based on AMP, we propose a supervised variant SAMP that directly learns attentional weights through supervised signals. In this section, the attentional weights are described in form of an interaction matrix I with a graph structure, and a target auxiliary interaction matrix I' is constructed to optimize I.

We model the interaction relationship of all agents by a graph neural network (GNN) [8] $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each agent is denoted as a node $v_i \in \mathcal{V}$ and a directed edge $e_{ij} = (v_i, v_j)$ represents the interaction of $v_i \rightarrow v_j$. In our case, the directed edge e_{ij} is actually the attention weight α_{ij}, which represents the interaction strength of agent i w.r.t to agent j. A global representation of all attention weights can be described in a form of interaction matrix $I \in \mathbb{R}^{n \times n}$ (similar to an adjacency matrix):

$$I = \begin{bmatrix} 0 & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & 0 & \dots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \dots & 0 \end{bmatrix}$$

where the diagonal element is set to zero since there is no interaction with itself. The feature for each node v_i is described by its produced message m_i, thus the feature description of all nodes is summarized as a matrix $M \in \mathbb{R}^{n \times l}$. Actually, our c_i is the i-th row vector of a matrix C where $C = IM$.

Note that graph $\mathcal{G}$ is dynamic since both I and M rely on each agent's observation, which changes with the environment transiting from one state to another. Thus it may not be very easy to learn both I and M efficiently with only reinforcement signals. To address this, we use the target interaction matrix I' from the environment to directly optimize I. Generally, the I' can be any representative information about a graph structure in the matrix form, which is helpful to improve the agents' interactions. For instance, considering a game where the rewards are given under the collaboration of agents, the simplest case of I' may be a symmetric joint reward matrix:

$$I' = \begin{bmatrix} 0 & r_{12} & \dots & r_{1n} \\ r_{21} & 0 & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \dots & 0 \end{bmatrix}$$

where r_{ij} is a 0/1 binary value representing whether agent i and agent j simultaneously get rewards in current time-step. The idea

behind this I' is very straightforward that we want to strengthen the interaction α_{ij} when agent i and agent j get reward.

Notably, there is no limitation to construct I' as long as it can guide the optimization of communication by using the priori knowledge of the environment. Essentially, this auxiliary information can be regarded as a form of intrinsic rewards in [9], which is task-specific and needs appropriately construction according to the task scenario. We update the parameters of AMP of each agent i by minimizing the objective function L_i, which is the row sum of point-wise entropy-cross losses between I and I':

$$L_i = \sum_{j=1}^{n} -r_{ij}\log(\alpha_{ij}) - (1 - r_{ij})\log(1 - \alpha_{ij}), \tag{8}$$

Although I' can help the agent learn to correctly pay more attention to useful messages, the final cooperative performance also depends on each agent' learned policy updated by RL.

3.3 Actor Critic Reinforcement Learning

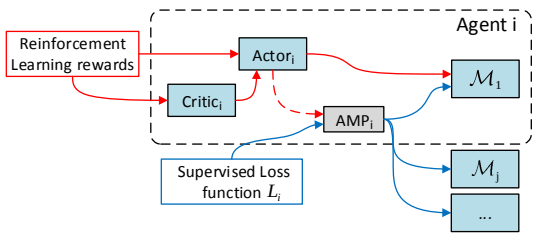

Figure 2: The learning process in SAMP

We use actor-critic architecture of RL to make each agent learn its action policy. For each agent i, the actor outputs the action policy $\pi(a_i|m_i, c_i; \theta_i)$ while the critic approximates value function $V(m_i, c_i; \omega_i)$ with paramters ω_i. The actor and critic are updated according to asynchronous advantage actor-critic (A3C) [12]. At the time-step t, the actor is optimized through policy gradients as $\nabla_\theta \log \pi(a_i^t|m_i^t, c_i^t; \theta_i)A_i^t$ where the advantage function A_i^t is given by $R_i^t - V(m_i^t, c_i^t; \omega_i)$, and the critic is optimized with gradients $\partial(R_i^t - V(m_i^t, c_i^t; \omega_i))^2/\partial\omega_i$. The policy and the value function are updated after every action until when a terminal state is reached, thus R_i^t can be computed by $R_i^t = r_i^t + \gamma V(m_i^{t+1}, c_i^{t+1}; \omega_i)$ where m_i^{t+1} and c_i^{t+1} are the produced message and the integrated message of next time step. The critic shares all the non-output layers with the actor and outputs one linear value V_i

Since all the functions and operations are continuous and differentiable, an end-to-end training is allowed to apply for the whole model. Figure 2 shows the learning process of SAMP: the red lines denote RL derived gradients, while blue lines indicate the gradients derived from supervised loss function L_i. The learning process of AMP is similar with SAMP except for the gradients from L_i, which are replaced by the gradients from actor denoted as the red dot line. The message extractor $\mathcal{M}_i(o_i^t, \phi_i)$ is optimized by the gradients passed down from each communicating module through backpropagation.

4 EXPERIMENT

In this section, we compare our SAMP and AMP with DIAL, IL, CommNet, VAIN in a fully cooperative, partially observable, multi-agent benchmark task. The SAMP and AMP share the same individual model architecture. For brevity, we describe only the AMP here: the messages extractor uses one hidden MLP that generates message with length $l = 255$, and the sizes of W, U, v are set to $255 \times 255, 255 \times 255, 255 \times 1$ respectively. The actor is implemented by one stack layer LSTM with hidden size of 255, unrolled in two steps where the first feeding is c_i and then follows agent's own produced message m_i. We execute 32 actor-learner threads running on 32 environment instances and the parameters are optimized by Adam with default hyperparameters and a learning rate of 2×10^{-4}.

4.1 Environment Setup

We consider a typical predator-prey pursuit problem, which has been known as one of the benchmark problems for multi-agent RL [15]. However, previous pursuit problems are in a grid-world where the agent moves only one grid each step in four directions, i.e. up, down, left, right or holding still within the grid boundary. The limitation of these grid-world settings is restricted and discrete state-action space, which is hard to extend in the real-world applications. To better distinguish the performance of each method and simulate a more realistic case, we implemented a partially observed predator-prey environment with continuous state-action space, which increases the complexity of learning coordinated behavior of agents.

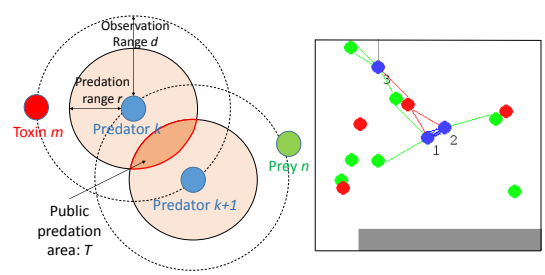

Figure 3: Predator-prey domain with toxin role

Specifically, as Figure 3 (Right) shows, there are n predators (blue), m toxins (red) and k preys (green) in total moving in the environment. The scenario is that we control predators, also called agents, to catch preys and avoid toxins. Each predator has limited visual observation range d and predation range r as in Figure 3 (Left). The overlap of predation circles represents the public predation area T and the prey is caught only when it is fully covered by the public predation area. Each agent is awarded with a positive reward when it catches a prey with the coordination of fellow agents and a negative reward is given if it collides against any toxin.

The optimal strategy of agents is trying to maintain the maximum of public predation area T while cruising around and avoiding the contacts of any toxin at the same time. Each agent has k eye sensors observing the distance to the wall, the prey, the toxin and the fellow agent, and there are 4 actions available for each agent to control the velocity.

Since agents get rewards only when they are close enough to form public predation areas, the r_{ij} in target auxiliary interaction matrix I' is set to the 0/1 binary value representing whether agent j is in observation range of agent i. We consider a standard predator-prey pursuit scenario of 5 predators, 20 preys and 10 toxins (5-20-10) and we carried out the experiment with 40 epochs and one epoch contains 100,000 steps.

4.2 Training Performance

We first focus on a direct comparison between CommNet, VAIN, IL, DIAL and our AMP, SAMP. The performance of each approach is evaluated by the sum of the rewards (also called score) of all the agents in each episode. As Figure 4 depicts, both our SAMP and AMP outperform all the other methods in score and reaches roughly 150% of the score obtained by CommNet and VIAN, and 300% of the DIAL and IL. This result indicates our method has a better learning ability in coordinating the agents to catch preys.

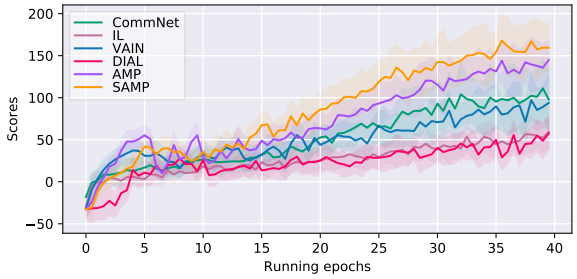

Figure 4: Training Performance

In addition to the score, the introduced toxin role promotes an extra evaluation indicator: the scoring rate (SR), which is the ratio of the score to the collision times of toxins. The scoring rate is proposed to reflect the ability for agents to earn the rewards per collision. We investigate the average performance of each approach and the random policies in the last 20 epoches, and show the statistical results presented in Table 1. The scoring rate in Table 1 shows that both our SAMP and AMP have a strong ability to earn the rewards per collision, which means that the agents in SAMP and AMP could handle massive received messages more efficiently under these environment settings.

Only when two or more agents form the public predation area, do they allow to get rewards, thus the key to this cooperative task is to maintain the distance between agents while they moving, and to avoid getting separated when they encounter toxins. Compared with VAIN, DIAL and CommNet, our AMP indirectly reinforces each agent to give more attention to closer fellow agents by increasing the weights of their sending messages, therefore agents are more likely to group and gather together to earn rewards. This interaction pattern is strengthened in the SAMP since the optimization of attentional weights is directly guided by our target interaction matrix, which accounts for its superiority over AMP.

4.3 Attention visualization

Let's further visualizate the attentional weights generated by AMP and SAMP to understand the nature of interactions between the agents. We randomly sample several states from the environment in the last training epoch and draw corresponding heatmaps of the interaction matrix I in AMP and SAMP respectively in Figure 5.

Table 1: Statistical performance

Approach	Scores	Collisions	SR
Random	-38.1±11.9	53.3±7.6	-0.71
IL	40.7±18.8	16.3±4.3	2.5
DIAL	35.0±18.7	17.6±4.6	2.0
VAIN	69.8±26.5	15.5±4.3	4.5
commNet	82.5±33.0	16.5±4.3	5.0
AMP	**110.3±33.8**	**14.6±3.8**	**7.6**
SAMP	**133.1±36.8**	**13.6±3.8**	**9.8**

Before we dig into the details, we first give an analysis on the useful interaction pattern: 1) if agents have view of each other, their interactions are effective and transmitted messages are useful to coordinate their actions, such as moving closer to form the public predation area or moving to the same direction to search the preys; 2) in contrast, if an agent is completely isolated from the others, the messages from others are just like noisy information, so it should ignore all the received message and only focus on its own observation to avoid the toxins.

Figure 5 explains four states in a column order. The first column describes a state where agent 1 and agent 4 are in observation range of each other but others are completely isolated. Under this state, the ideal interaction pattern is that: the agent 1 and agent 4 only care about the message sent from each other while the others only care about its own observation. This pattern is correctly learned in SAMP, and also in AMP, but there exist some distractions of the attentions for the isolated agents in AMP. The noisy attentions may account for the reason of the better performance in SAMP with supervised signals compared to that with only RL signals.

The second column depicts a situation where all the agents are separated from each other, and in this case, each agent i should filter all received message ($\alpha_{ij} \approx 0$) and focus only on its own observation, which is correctly learned in SAMP but some noisy attentions are still assigned in AMP. The third column can be similarly explained as the first column. The fourth column shows a case where three agents are holding a public predation area, and the result in SAMP proves that each agent can simultaneously pay the K most attentions on received messages, such as agent 2 gives full attentions on messages from agent 1 and agent 3 ($\alpha_{21} \approx \alpha_{23} \approx 1$), but AMP does not learn this effective pattern ($\alpha_{21} \approx 1$, $\alpha_{23} \ll 1$)

5 CONCLUSION

This paper provides a supervised attention-based message processing method SAMP to directly model the relevance of received messages through the target auxiliary information, which benefits agents in learning correct interactions more efficiently. Serval experiments are conducted to verify the effectiveness of our methods in a benchmark environment, and the results show that our methods outperform existing methods in such environments.

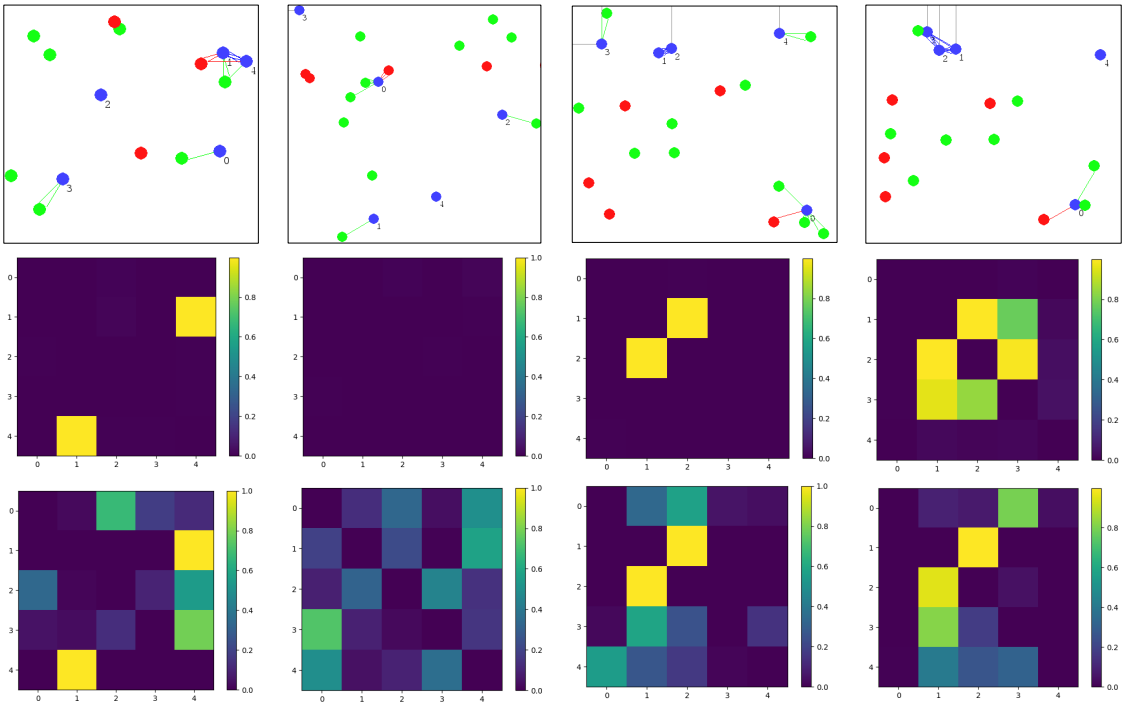

Figure 5: Four sampled screenshot of the environment (the first row). Four corresponding heatmaps of matrix I generated by SAMP (the second row) and AMP (the third row). For each heatmap, the color block at i-th row and j-th column represents the interaction strength of agent i w.r.t agent j, whose value is the attentional weight α_{ij}.

Further investigation includes surveys on more representations of I' and the performance of our methods in more environments such as more toxins and less preys, and so on.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*. 2137–2145.

[3] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926* (2017).

[4] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Philip Torr, Pushmeet Kohli, Shimon Whiteson, et al. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1702.08887* (2017).

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

[6] Matthew Hausknecht and Peter Stone. 2016. Grounded Semantic Networks for Learning Shared Communication Protocols. In *International Conference on Machine Learning (Workshop)*.

[7] Yedid Hoshen. 2017. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems*. 2698–2708.

[8] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[9] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*. 3675–3683.

[10] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 464–473.

[11] Hangyu Mao, Yan Ni, Zhibo Gong, Weichen Ke, Chao Ma, Yang Xiao, Yuan Wang, Jiakang Wang, Quanbin Wang, Xiangyu Liu, et al. 2017. ACCNet: Actor-Coordinator-Critic Net for" Learning-to-Communicate" with Deep Multi-agent Reinforcement Learning. *arXiv preprint arXiv:1706.03235* (2017).

[12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. 1928–1937.

[13] 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. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[14] Igor Mordatch and Pieter Abbeel. 2017. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908* (2017).

[15] Kozue Noro, Hiroshi Tenmoto, and Akimoto Kamiya. 2014. Signal learning with messages by reinforcement learning in multi-agent pursuit problem. *Procedia Computer Science* 35 (2014), 233–240.

[16] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep Decentralized Multi-task Multi-Agent RL under Partial Observability, In International Conference on Machine Learning. *arXiv preprint arXiv:1703.06182*, 2681–2690.

[17] Liviu Panait and Sean Luke. 2005. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems* 11, 3 (2005), 387–434.

[18] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. 2017. Multiagent Bidirectionally-Coordinated nets for learning to play StarCraft combat games. *arXiv preprint arXiv:1703.10069* (2017).

[19] Zhaoqing Peng, Takumi Kato, Hideyuki Takahashi, and Tetsuo Kinoshita. 2015. Intelligent home security system using agent-based IoT Devices. In *Consumer Electronics (GCCE), 2015 IEEE 4th Global Conference on*. IEEE, 313–314.

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

[21] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.