Cooperative Multi-agent Attentional Communication for Large-scale Task Space

Qijie Zou,1 Youkun Hu,1 Dewei Yi,2 Bing Gao,1 and Jing Qin1

1 Department of Information Engineering Faculty, Dalian University of China, 116622, China.
2 Department of Computing Science, University of Aberdeen, Aberdeen AB24 3UE, UK.

Correspondence should be addressed to Youkun Hu; huyoukun163@163.com

Abstract

With the rapid development of mobile robots, they have begun to be widely used in industrial manufacturing, logistics scheduling, intelligent medical and other fields. For large-scale task space, the communication between multi-agent is the key to affect cooperation productivity, and agents can coordinate more effectively with the help of dynamic communication. However, the traditional communication mechanism uses simple message aggregation and broadcast, and in some cases lacks the distinction of the importance of information. Multi-agent deep reinforcement learning (MDRL) is valid to solve the problem of informational coordination strategies. However, how different messages affect each agent's decision-making process remains a challenging task for large-scale task. To solve this problem, we propose IMANet (Import Message Attention Network), It divides the decision-making process into two sub-stages: communication and action, where communication is considered to be part of the environment. First, an attention mechanism based on query vectors is introduced. The correlation between the query vector agent's own information and the current state information of other agents is estimated, and then the results are used to distinguish the importance of information from other agents. Second, LSTM network is used as the unit controller for each agent, and individual rewards are used to guide the agent training after communication. Finally, IMANet is evaluated on tasks on challenging multi-agent platforms, Predator and Prey (PP) and Traffic junction. The results show that IMANet can improve the efficiency of learning and training, especially when applied to large-scale task space, with a success rate 12% higher than CommNet in baseline experiments.

1. Introduction

Multi-agent system is very practical in distributed control, remote scheduling, and modeling analysis [1]. Compared with a single agent, it can complete tasks more effectively and has better robustness, reliability, and scalability. Communication is the basis for maintaining the efficiency and organization of multi-agent systems [2]. In the DECPOMDP (Decentralized Partially Observable Markov Decision Process) environment, through communication [3], agents can exchange their observations to better discover the current global state and understand the actions and intentions of other agents. However, traditional pre-defined communication protocols and broadcast messages cannot allow multi-agent to effectively "learn to communicate" in a large-scale task. Reinforcement learning (RL) is mainly to study how agents choose actions by perceiving local and global states, and constantly interact with the dynamic environment in order to find optimal policy that maximize cumulative rewards [4, 5].

The reinforcement learning problem in multi-agent scenarios is more complex than in single-agent scenarios. Because the agent interacts with
the environment at the same time and treats other agents as part of the environment, the agent usually faces the dimensionality disaster (grows exponentially with the number of agents, so the multi-agent system dimension is very large and computationally complex) [6], credit assignment (identify the impact of the agent’s behavior on this global return) and other issues [7]. As the number of agents increases, the task space is extended to large-scale and cooperative information is augmented. Deep learning is an efficient representation learning that can discover the key information in the original message [8]. The main reason is that the neural network can process the input high-dimensional data and extract useful expressions [9,10]. The main advantage of deep reinforcement learning (DRL) is that it can extend RL to high-dimensional state and action spaces [11].

In recent years, multi-agent deep reinforcement learning (MDRL)[12], a combination of deep learning (DL) and multi-agent reinforcement learning (MARL), has been successful in many complex environments. Such as StarCraft [13], particle environment [14]. However, these tasks either assume that the environment is completely observable, or there is a lack of communication between agents.

In this paper, we will investigate how to use Deep MARL’s approach for effective communication learning in a partially observable distributed environment.

In the centralized training paradigm, as the number of agents increases, the linear growth of the input dimension and the exponential growth of the output space make, the centralized training method using the traditional central controller makes the algorithm not easily scalable to large-scale tasks, the convergence of the algorithm becomes poor or even unable to converge, and the problem of poor scalability. In the communication process, broadcast communication is a common setting for the study of "learning communications" between agents, but this does not allow selective attention to the observations and actions of other agents, does not provide useful information to agents in the decision-making process, and leads to unstable learning processes. These problems are caused by the inability of traditional reinforcement learning methods to learn cooperative strategies through effective communication in DEC-POMDP conditions [15]. To this end, we propose the IMANet method for multi-agent deep reinforcement learning.

The main contributions of this paper are as follows:

1) We use a query vector-based attention mechanism in the IMANet communication structure to identify the messages contained in more favorable specific agents. In this algorithm, the local observations of each agent are encoded and attention is directed to different agents according to the magnitude of the attention weights, generating dynamically changing communication vectors to coordinate policies. The problem of inability to judge and distinguish the importance of messages is solved to make the learning process more efficient and stable.

2) Using a single individual LSTM network as a controller for each agent [16], individual observations and communication vectors from the agents themselves are processed. Our independent control model selectively outputs important information, thus alleviating the problems associated with dimensional explosion, which makes it possible for agents to learn coordination strategies in large-scale spaces.

2. Notation and Background

2.1 Technical Background

Dec-POMDP is a multi-agent extension of partially observable Markov decision process. First, our approach requires the introduction of the necessary notation, and then three common reinforcement learning framework structures are introduced.

Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs). We consider a fully cooperative multi-agent setting that can be formulated as DEC-POMDP [17]. It is formally defined as a tuple \( \langle N, S, A, T, R, 0, Z, \gamma \rangle \), Where \( N \) is the number of agents, \( S \) is the state space; \( A = A_1 \times A_2 \times \ldots \times A_N \) is the action space of all agents, and \( A_j \) is the set of local action \( a_j \), that agent \( j \) can take; \( T : S \times A \times S \rightarrow [0,1] \) is the state transition probability. \( R = (r_1, r_2, \ldots, r_N) \) is the set of rewards,
Where: $r^t$ : $S \times A \times S \rightarrow \mathbb{R}$ is the reward function of an agent $j$; $O = \{O_1, ..., O_n\}$ is the set of joint observation $o$; $Z : S \times A \rightarrow O$ is observation function; $\gamma \in [0, 1]$ is the discount factor. The set of action policies is $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$, each policy is represented by a neural network and its parameter set $\theta = \{\theta_1, \theta_2, ..., \theta_n\}$.

A policy $\pi(a^t_j | h^t_j; \theta^t)$ is the probability of taking action $a^t_j$ when encountering history $h^t_j$ under the policy parameters $\theta^t$. The agent’s action $a^t_j$ depends on the encountered history $h^t_j$. We generally omit the parameter $\theta^t$ in policy $\pi(a^t_j | h^t_j; \theta^t)$ for brevity, and denote the policy as $\pi(a^t_j | h^t_j)$ . We use $h^t_j$ to denote the history of individual observations $o^t_j$, individual rewards $r^t_j$, and individual actions $a^t_j$ encountered by the agent $j$ following policy $\pi$ at the time $t$ and is defined as $h^t_j = \{s^t_j, o^t_j, r^t_j, a^t_j, ..., s'^{t-1}_j, a'^{t-1}_j, s'^{t-1}_j\}$ . Sequence $h^0_j = \{s^0_j\}$ denotes the history at time $t = 0$, contains only the star state $s^0_j$ of the agent $j$. The return of an agent $j$ that interacts with the environment to produce a history $h^t_j$ is written as $R(h^t_j) = \sum_{t=0}^{T} y^t r^t_j$.

### 2.2 Multi-agent reinforcement learning architecture

Three architectures of multi-agent reinforcement learning: **Decentralization:** Without a central controller, the agent makes independent decisions based on its own policy network. **Fully centralized:** The central controller makes decisions for all agents. **Centralized training and decentralized execution:** The central controller is used only by the training process. Each agent makes a decision on its own policy network by its own observations.

**Decentralization.** All agents are independent individuals, and they do not communicate with each other. As shown in Figure 1. Each agent independently interacts with the environment to obtain individual observation $o^t_i$ and individual rewards $r^t_i$. Each agent deploys its own policy network and trains its own policy network independently, exactly the same as the reinforcement learning of a single agent. After the training, each agent utilizes its own policy network to make a decision, and the observed $o^t_i$ is the input of the policy network and the probability distribution $\pi(a^t_i | o^t_i; \theta^t)$ is the output. To produce a discrete action $a^t_i$, we sample from this distribution: $a^t_i \sim \pi(\cdot | o^t_i; \theta^t)$, and then execute the action $a^t_i$.

Regardless of training or execution, there is no communication between agents. The essence of such decentralization is single-agent reinforcement learning, rather than true multi-agent reinforcement learning. Although this decentralized structure can well deal with the problems caused by the growth of the number of agents, the effect of single-agent reinforcement learning for multi-agent reinforcement learning does not usually work. The single-agent algorithm assumes that these functions are stable, and in a multi-agent scenario, it will face the problem of environmental instability. The reason is that the influence between them should not be ignored.

$$
\pi(a^t_i | o^t_i; \theta^t) \pi(a^t_j | o^t_j; \theta^t) \pi(a^t_\theta | o^t_\theta; \theta^t)
$$

**Fully centralized.** There are $n$ agents interacting with the environment, and each agent will change the environment, thereby affecting other agents. As shown in Figure 2. There is no policy network on the agent, so the agent cannot make decisions by itself, and all have to be commanded by the central controller.

During training, the agent reports its observation $o^t_i$ and reward $r^t_i$ to the central government. The policy network is in the center, and the center transmits the decision $a^t_i$ to the agent $i$. The agent
performs actions in accordance with the instructions of the center, and interacts with the environment to do it in the center. The central controller uses all observation $o$, reward $r$, and action $a$ to train the policy network. Even after the training is completed, the central controller is needed when it comes to decision-making.

$N$ policy networks are trained on the central controller, and their network structures are the same, but the parameters may be different. Use $\theta$ to denote the parameters of the i-th agent. The input of the policy network is all the observations $o^i$ of the agent. The i-th policy network determines the action $a^i$ of the agent $i$, and the decision can only be made by the central government. This is because the policy network needs to use the observations of all agents. An agent only knows its own observation $o^i$, it does not have enough information to make a decision, so the policy network cannot be deployed on the agent, and can only stay on the central controller. In the execution, all agents report their observations to the central government, then the central government determine each agent what to do. The center transmits $a^1$ to $a^n$ to the corresponding agent.

\[ \pi(a^i | o^1, ..., o^n; \theta), \text{for all } i = 1, 2, ..., n \]

![Figure 2: Fully centralized architecture.](image)

The advantage of centralization is to know the overall information, which can help all agents make better decisions. But centralization also has disadvantages. The execution speed is slow. The agent itself has no decision-making power, and all decisions are made by the central government. The agent reports all its observations to the central government, and the central government collects global information before making decisions. The biggest problem faced by this centralized structure is the huge input and output space. With the increase in the number of agents, the input dimension increases linearly, and the space exponential of the output joint policies increases.

**Centralized Training Decentralized Execution (CTDE).** In the traditional CTDE architecture, each agent has its own policy network and the team has a central controller during training, which helps the agent train the policy network. After training, there is no need for a central controller, and each agent interacts with the environment independently. The agent has own policy network to act based on your own local observations. There are many different models of CTDE, which are popular nowadays (such as MAAC proposed by Sha et al [18], COMA proposed by Foerster et al [19]).

3. Related Work

Recently, the direct communication method of embedding communication channels in deep neural networks (direct communication:The use of specific communication channels can selectively complete the information exchange between various agents [20]). For learning communication protocols, it has been proven to be very effective [21-23]. Under normal circumstances, the continuous transmission between agents through the network forms a Communication channel, which makes the agents consider local information and global information at the same time during the learning process. The protocol can be optimized at the same time as the network is optimized.

Learning an effective multi-agent communication protocol is mainly divided into the following two aspects.

3.1 Broadcast partial observations of each agent to all agents

Sukhbaatar proposed the CommNet algorithm. CommNet is just a single network designed for all agents [24]. It solves the Dec-POMDP problem. It follows the centralized training and decentralized execution method. This single-network communication channel cannot be easily extended to a large-scale agent environment. Due to continuous communication, the controller can learn through
backpropagation. However, the information transmission method adopted by this algorithm is to broadcast communication content to all agents, which will also cause waste of bandwidth resources. Aiming at the problem of channel occupation when broadcasting a message to each agent within the communication range. Kim proposed SchedNet [25], a learning method for multi-agent deep reinforcement learning. This method introduces the CSMA protocol, which is a contention-type access medium. When the agent sends a message to the channel, it monitors whether the channel is occupied at this time, and stops sending if it is occupied. When a conflict occurs, it will stop sending the message, wait for a suitable time, and send it randomly. This method also alleviates the message loss problem caused by the agent through broadcast communication. The Message-Dropout MADDPG uses the message dropout technology in multi-agent reinforcement learning in order to allocate communication resources reasonably [26]. This method uses a centralized training decentralized execution framework under fully or partially observable conditions, discards the received message with a certain probability in the training phase, and compensates by multiplying the weight of the discarded block unit by the correction probability this influence. This method is also robust against communication errors.

3.2 Selective and targeted communication through the use of attention mechanism

As it is difficult for multi-agents to distinguish between valuable information and shared information, Lu proposed the ATOC based on the actor-critic framework [27]. In this algorithm, the local observation value of each agent is coded. The attention unit is used to determine which agents to communicate with (screening agents for information sharing), and a two-way LSTM network is used as the communication channel between communication groups. Agents in this communication group exchange information with each other in the communication channel. The attention module is an RNN network and will face the problem of vanishing gradients. Without a centralized controller, decentralized agents cannot learn effective and decentralized cooperation policies in a complex environment. Wang proposed a new attention-based communication neural network (CommAttn) [28]. CommAttn can use the display communication method to automatically learn and explore the cooperation policies in the problem, by modeling the interaction between agents and introducing the attention mechanism. Calculate the relevance of the received message, and determine whether communication between agents is required, so that the agent can decide who to communicate with. The learned communication model of the system is more able to adapt to the dynamically changing environment. Facebook AI Research proposed a collaborative multi-agent deep reinforcement learning method, namely TarMAC [29]. Allow targeted communication between agents. The purpose is similar to ATOC's attentional mechanism. It not only determines who to send the message to, but also the part of the observation that is most relevant to the goals of other multi-agents. This targeted communication behavior is achieved through a signature-based attention mechanism: Together with the message, the sender broadcasts a key that encodes the attributes of the multi-agent targeted by the message and is used by the receiver to measure the relevance of the message. To cope with how to select important messages and how to process important messages efficiently. Mao proposed a DRL method called Double Attentional Actor-Critic Message Processor (DAACMP) [30], which embeds a first class attention mechanism in the actor part, where the importance of a message is positively related to the distance between two agents, the Actor Attention is able to pay attention to the messages of nearby agents adaptively. Embedding the second type of attention mechanism in the Critic part, where the joint action policies of team-mates are modeled using Q-values, and similar joint actions are grouped and processed instead of processing the action policy of individual agent, the Critic Attention has a more sophisticated ability to process all important messages.

In the above method, using the method of multi-agent deep reinforcement learning with direct communication, there will be an obvious information transmission process between the agents, and the communication objects can be selected to reduce the problem of the explosion of the joint action space or by optimizing the communication content. This method allows the agent to pay atten-
tion to more important messages when sending information, reduce the bandwidth occupation in the communication channel, and improve calculation efficiency.

The IMANet method in this article can be seen as an extension of the CommNet. The CommNet algorithm is to chain the messages together and broadcast them, IMANet adds an attention vector to learn the importance weight of each message from other agents, obtain their weighted sum and use it to perform operations. You can selectively conduct partial interactions and determine which multi-agents provide shared information that can improve performance and make the training process more stable. And because CommNet uses a central controller for centralized training, the scalability is poor, so we have an independent controller.

4. IMANet method

4.1 Basic description

We first introduce the necessary assumptions for our approach, and then describe in detail our multi-agent communication architecture.

Hypothesis 1 The decision-making process is divided into two sub-phases: communication sub-phase and action sub-phase.

Hypothesis 2 During training, the current content of the agent’s communication is related to the agent’s own encounter history and the encounter histories of other agents (i.e., the hidden layer states in the next section).

Hypothesis 3 Each agent’s policy depends only on its own hidden layer state.

4.2 IMANet architecture

We propose a new deep MARL framework with targeted communication, called IMANet, whose overall architecture is depicted in Figure 3. IMANet consists of the following three levels of control structure.

![Diagram](Image)

Figure 3: An overview of our IMANet model (a) Optimize the individual rewards of each agent based on the observation results. (b) The produce of communication vectors for each agent in a single communication step. (c) The module view of the LSTM unit.

NC (NO Communication) is an independent control that uses LSTM network structure, where each agent is controlled by an independent LSTM. IMANet without communication is exactly NC. For the j-th agent, its policy is defined as equation (1)-(3).

\[ s_{j}^{t+1} = \text{LSTM} (e(o_{j}^{t}), h_{j}^{t}, s_{j}^{t}) \]  \hspace{1cm} (1)

\[ h_{j}^{t+1} = \text{LSTM} (e(o_{j}^{t}), h_{j}^{t}, s_{j}^{t}) \]  \hspace{1cm} (2)

\[ a_{j}^{t} = \pi(h_{j}^{t}) \]  \hspace{1cm} (3)

Where \( o_{j}^{t} \) is the observation of the agent \( j \) at a time \( t \), \( e(.) \) is the encoder function parameterized by the fully connected neural network, and \( \pi \) is the ac-
tion policy of the agent. In addition, $h^j_t$ and $s^j_t$ respectively represent the hidden layer state and memory cell state of the agent $j$ at a time $t$. Use the same LSTM model for all agents and share their parameters. All agents share a unit, which has a higher utilization rate of samples, shared parameters reduce model complexity.

**IMANet** extends this independent controller model NC, a vector $c_t$ containing communication is introduced, which allows agents to obtain local information observed by other agents through exchange to observe the global state of the system. As in hypothesis 1, the IMANet model divides the decision-making into two sub-phases: communication and action, as shown in Figure 4. Before choosing an action, use the method based on the attention mechanism to decide which important information to pay attention to.

![Diagram of Decision sub-stage](image)

**Figure 4:** Decision sub-stage.

The policy for the $j$-th agent in the IMANet network is defined as equation (4)-(6).

$$s^{j+1}_t = \text{LSTM}(e(o^j_t), c^j_t, h^j_t, s^j_t)$$  \hfill (4)

$$h^{j+1}_t = \text{LSTM}(e(o^j_t), c^j_t, h^j_t, s^j_t)$$  \hfill (5)

$$\alpha^j_t = \pi(h^j_t)$$  \hfill (6)

According to Figure 3(c), the candidate memory cells $\tilde{s}^{j+1}_t$ and the gate values functions $\gamma_u$, $\gamma_f$, and $\gamma_o$ are defined as equation (7)-(10).

$$\tilde{s}^{j+1}_t = \tanh(\tilde{w}_s[h^j_t, e(o^j_t), c^j_t] + b_s)$$  \hfill (7)

$$\gamma_u = \delta(w_u[h^j_t, e(o^j_t), c^j_t] + b_u)$$  \hfill (8)

$$\gamma_f = \delta(w_f[h^j_t, e(o^j_t), c^j_t] + b_f)$$  \hfill (9)

$$\gamma_o = \delta(w_o[h^j_t, e(o^j_t), c^j_t] + b_o)$$  \hfill (10)

Among them, since the same LSTM model is used for each agent, that is, the update gate $\gamma_u$, which determines what information we want to store in the cell state, has parameters $w_u$ and $b_u$. The forget gate $\gamma_f$, which determines what information we want to discard from the cell state, has parameters $w_f$ and $b_f$. The next tanh layer creates a candidate vector $\tilde{s}^{j+1}_t$, which will be added to the cell state. In the next step, we will combine these two vectors to create the update value, which includes parameters $\tilde{w}_s$ and $b_s$. Finally, the output gate $\gamma_o$, where we need to decide what we want to output, will be based on our cell state, which includes parameters $w_o$ and $b_o$. That is, all the parameters are in the four $w$ and $b$ $\delta$ represents the sigmoid function, which makes the gate value very close to 0 or 1. At each time step, Through an activation function $\tanh$, the current input $e(o^j_t)$, $c^j_t$, and the $h^j_t$ passed down from the previous state are spliced and trained to obtain $\tilde{s}^{j+1}_t$. Here, tanh is used because $\tilde{s}^{j+1}_t$ is used as input data instead of a gate signal. The three gate values above the update gate $\gamma_u$, the forget gate $\gamma_f$, and the output gate $\gamma_o$ allow the values flowing through the network to be adjusted.

The update gate and the forget gate are used to update the value of the state $s^{j+1}_t$, the state value is defined as equation (11).

$$s^{j+1}_t = \gamma_u * \tilde{s}^{j+1}_t + \gamma_f * s^j_t$$  \hfill (11)

Specifically, $\gamma_f$ is used as a forgetting gate to control which information of the previous state $s^j_t$ should be retained and which should be forgotten. As an update gate $\gamma_u$, select and memorize the input coding observations $e(o^j_t)$ and communication vectors $c^j_t$ at this stage, and record more important contents. Adding the results of the above two, it means that part of the information of the current state $h^j_t$ is deleted and some information of the new input $\tilde{s}^{j+1}_t$ is added to obtain the next state $s^{j+1}_t$.

The output gate is used to update the hidden layer state function $h^{j+1}_t$, the hidden layer state is defined as equation (12).
\[ h_j^{t+1} = \Gamma_0 \ast \tanh s_j^{t+1} \] (12)

The \( s_j^{t+1} \) obtained in the previous stage is scaled by an activation function \( \tanh \) and controlled by the output gate \( \Gamma_0 \). This stage will determine which states will be used as the output of the current hidden layer state \( h_j^{t+1} \).

**Training.** We use the reinforcement learning method based on Policy gradient to train the action policy. IMANet uses independent controllers to train different agents for guidance, executed in a decentralised manner, as each agent only needs its local state vector and the weighted sum of incoming messages to act. As shown in Figure 5.

![Figure 5: IMANet controller model.](image)

Each controller trains only one agent and the system has \( n \) agents. As shown in Figure 3(a), the agent \( j \) interacts with the environment at time \( t \) to obtain individual observations \( o_j^t \), individual actions \( a_j^t \) and individual rewards \( r_j^t \) from the environment. For training, the agent \( j \) needs an independent controller. As shown in Figure 5, for agent \( j \) at time \( t \), the attention unit performs a purposeful fusion based on the received hidden layer state \( h_i^t \sim h_n^t \) (\( i \neq j \)) of other agent to generate a communication vector \( c_j^t \). The hidden layer state \( h_j^t \) of agent \( j \) and the communication vector \( c_j^t \) containing the encounter histories of other agents are sent to the independent controller of agent \( j \). Through a communication mechanism, the different agents can exchange information about their observations, actions and intentions to stabilize their training process. After training, the agent \( j \) makes its own decisions based on its own hidden layer state(encounter history) \( h_j^t \). And the policy network outputs a probability distribution \( \pi(a_j^t | h_j^t) \).

The overall performance measure of the policy \( \pi \), is denoted as \( J(\pi) \). It is defined as equation (13).

\[
J(\pi) = E\{R(h_j^t)\} = \int p(h_j^t)R_jdh_j^t \tag{13}
\]

where \( p(h_j^t) \) is the probability of the existence of each sequence \( h_j^t \) given the parameter \( \theta_j^t \) of the policy \( \pi \).

To optimize the policy \( \pi \), we want to update the parameter \( \theta_j^t \) along the gradient of \( J \) to an optimal, it is defined as equation (14).

\[
\theta_j^{t+1} = \theta_j^t + \alpha \nabla_{\theta_j} J(\pi) \tag{14}
\]

where \( \alpha \) is the learning rate, the policy are updated by ascent with the following gradient, the policy gradient is defined as equation (15).

\[
\nabla_{\theta_j} J(\pi) = \nabla_{\theta_j} \int p(h_j^t)R_jdh_j^t = \int \nabla_{\theta_j} p(h_j^t)R_jdh_j^t \tag{15}
\]

**Communication.** Establishing effective collaboration policies requires targeted communication, that is, the ability to send specific messages to agents. We use the attention mechanism based on query vector in the communication structure to identify more beneficial specific agent messages and realize the fusion of messages, leading to specific communication links according to the size of the attention weight. The attention model based on query vector is shown in Figure 6.

![Figure 6: Attention model for query vector.](image)
As shown in Figure 3(b) and Figure 6, in the attention model. The communication vector \( C'_j \) is defined as equation (16)-(18).

\[
C'_j = \sum_{i \neq j} w_i^t h_i^t \tag{16}
\]

\[
e'_t = \text{Score}(h'_j, h'_i) \tag{17}
\]

\[
w_i^t = \frac{\exp(e'_i)}{\sum_{n=1} \exp(e'_n)} \tag{18}
\]

In the communication process, we try to understand the content of the communication received by agent \( j \) from other agents. \( C'_j \) is the communication vector of the agent \( j \) at time \( t \). The score function(\( \text{Score}(h'_j, h'_i) \)) is used to calculate the hidden layer state \( h'_j \) at the current moment as the query vector, and do the inner product operation with the hidden layer vectors \( h'_i - h'_n (i \neq j) \) passed by other agents at the current moment, respectively, to get the weighting coefficient \( w'_i \) of the similarity size, and the size of the coefficient reflects the importance of the content at the same time. The attention mechanism module is a simple LSTM network. The communication vector at the current moment can be obtained through the weighted addition of the hidden layer. At this time, the agent focuses on more important information. Take the agent’s own local observation and state coding as input, and use the communication vector generated by the query vector attention mechanism, that is, the state information observed by other agents as additional input, and the fused state of the hidden layer is output to guide the cooperation policies \( \pi \).

5. Experiment

5.1 Experimental Setup

5.1.1 Experiment environment settings. We evaluated IMANet, CommNet, and NC in the traffic junction task (Sukhbaatar et al., 2016) and Predator and Prey task [31]. CommNet is a communication method that uses broadcast. The detailed experimental environment will be described in detail in the following subsections. The experimental hardware environment uses Intel(R) Core(TM) i7-7700 CPU+GeForce GTX 1650+16GB; the software environment for the experiments uses PYTorch+Gym [32]; the agent updates the policy according to their respective reward functions. Using the RMSProp approach, the configuration of learning rate hyperparameters is done automatically by the algorithm, and RMSProp can be targeted to provide different learning rates for each parameter, this method was proposed by Geoff Hinton [33], thus improving the problem of fading learning rate.

We use a learning rate of 0.001, set the hidden size to 128 units, do 10 weight updates every round, and conduct 1000 rounds of experiments in a Predator and Prey task and traffic junction task. Use LSTM to realize the attention unit.

5.1.2 Scenarios. Predator and Prey: In the grid map of different sizes, there are \( n \) agents as predators to capture a static prey, and each predator has the same size of perception area. When the prey is caught, a positive reward is given. Until the end of this round, that is, other agents have reached the position of the prey, as shown in Figure 7.

5.1.2 Scenarios. Predator and Prey: In the grid map of different sizes, there are \( n \) agents as predators to capture a static prey, and each predator has the same size of perception area. When the prey is caught, a positive reward is given. Until the end of this round, that is, other agents have reached the position of the prey, as shown in Figure 7.

![Figure 7: predator-prey environments’ Visualizations.](image-url)
**Traffic junction:** In the task of traffic junction, the total number of cars is fixed at, cars avoiding collisions while passing through the intersection. The cars randomly follow one of the possible routes to reach the grid destination and are again added back to the environment with a different route assignment. We set up two one-way roads on a 7 × 7 grid, as shown in Figure 8(a), and the four connected junctions of two-way roads in 15×15 grid, as shown in Figure 8(b).

Figure 8 shows different colored circles representing cars controlled by agents, while different colored dashed lines represent possible routes. The cars take a probability from entering the arrivals point and the car can take two actions: braking and forward. When the car reaches the target position on the edge of the grid, it will be removed. One or more road intersections exist in different maps, so the diverse routes are optional. It is considered a collision that two cars occupy the same position at the same time, and the agents will be penalized, but will not affect the simulation. The identification of each car, its current location and the assigned route are encoded in a one-hot binary vector set, each car is finitely perceived as a grid of area 2×2 and can communicate with other cars under certain conditions.

5.1.3 **Metrics.** We will evaluate our method and baseline algorithm in terms of the following indicators.

In the Predator and Prey task:

(1) **Maximum allowable steps taken** $T_{\text{max}}$:

$$ T_{\text{max}} = kT_{\text{finish}} $$

where $T_{\text{finish}}$ is the average time period from the start of the movement of the first predator to the location of the last predator to reach the prey. Select $k$ values according to different map sizes and number of agents.

(2) **Success rate** $\beta$:

$$ \beta = \frac{n_{\text{success}}}{n} $$

where $\beta$ is the ratio of the number of successful events to the number of test events. An event is considered successful if all predators capture prey before the maximum allowed step taken $T_{\text{max}}$.

(3) **Agent density** $\rho_{\text{agent}}$:

$$ \rho_{\text{agent}} = \frac{n_{\text{agent}}}{W \times H} $$

where $\rho_{\text{agent}}$ denotes the sparsity of the environment, $n_{\text{agent}}$ is the number of agents in the map, and $W \times H$ is the size of the map.

(4) **Average step taken** $T_{\text{steps-taken}}$:

$$ T_{\text{steps-taken}} = \frac{T_{\text{total}}}{n_{\text{agent}}} $$

where $T_{\text{steps-taken}}$ denotes the average step taken of each agent to reach the goal, $T_{\text{total}}$ is the number of steps to complete the epoch, and $n_{\text{agent}}$ is the sum of the number of agents.
(5) Average reward \( r_{\text{average}} \):

\[
    r_{\text{average}} = \frac{1}{n_{\text{agent}}} \sum_{i=1}^{n} r_i
\]

where \( r_{\text{average}} \) denotes the average of rewards obtained by each agent at each epoch, \( r_i \) is the individual reward of agent \( i \), and \( n_{\text{agent}} \) denotes the number of agents.

In the Predator and Prey environment, the reward function for agent \( i \) is set as follows:

\[
    r_i(t) = (1 - \alpha_i) * r_{\text{penalty}} + \alpha_i * n_i * r_{\text{success}}
\]

Where \( \alpha_i \) indicates whether the agent \( i \) has captured the prey, and it is set to 1 or 0. 1 indicates that the agent \( i \) has captured the prey at a time \( t \), otherwise it is 0. \( r_{\text{penalty}} \) represents the penalty value at a time step, equal to -0.05, which can encourage the agent to actively explore the environment. \( n_i \) represents the number of agents that captured the prey at a time \( t \). \( r_{\text{success}} \) represents the reward value for catching the prey, which is equal to 0.05.

In the traffic junction task:

1. Maximum allowable steps taken \( T_{\text{max}} \):

\[
    T_{\text{max}} = kT_{\text{finish}}
\]

where \( T_{\text{finish}} \) is the average time period from the start of the movement of the first car to the arrival of the last one at the target position at the edge of the grid. Selecting \( k \) values according to different size of tasks.

2. Success rate \( \beta \):

\[
    \beta = \frac{n_{\text{success}}}{n}
\]

where \( \beta \) is the ratio of the number of successful events to the number of test events. The simulation ends after the Maximum allowable steps taken \( T_{\text{max}} \). No collision is classified as a success, and if one or more collisions occur, it is classified as a failure.

In the traffic junction environment, the reward function for agent \( i \) is set as follows:

\[
    \tilde{r}_i(t) = r_{\text{coll}} d_i + r_{\text{time}} \tau_i
\]

where \( r_{\text{coll}} \) denotes the penalty incurred when two cars collide, \( r_{\text{coll}} = -10 \). \( d_i \) is the number of times car \( i \) collides at time \( t \), but the collision does not affect the car's route. The actions the car can take at each time step are braking and moving forward. \( \tau_i \) is the number of steps that car \( i \) has elapsed from the time it starts to the moment \( t \), \( r_{\text{time}} \tau_i = -0.01 \tau_i \).

5.2 Ablation Experiment

IMANet is attentional communication model, we conduct ablation experiments on the structure of the attention unit, and NC is a simplified version of IMANet without communication. NC has to train an independent policy network for each agent. In order to verify the influence of independent controllers in IMANet on the increase in the number of agents and the advantages of communication. We increase the number of agents on a map of the same size and the density of agents increases accordingly. We can test this by performing it in three different Predator and Prey environments with sparse, normal and crowded agent densities \( \rho_{\text{agent}} \). With the success rate \( \beta \) and average reward \( r_{\text{average}} \), we can evaluate the performance of the IMANet and NC. The success rate \( \beta \) is shown in Table 1 and the average reward value \( r_{\text{average}} \) is shown in Figure 9.

<table>
<thead>
<tr>
<th>Agent Density (( n_{\text{agent}} ))</th>
<th>Predatory-Prey (Success Rate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10×10, 6 agents, sparse</td>
<td>83.3±0.9%</td>
</tr>
<tr>
<td>10×10, 8 agents, normal</td>
<td>83.4±0.8%</td>
</tr>
<tr>
<td>10×10, 10 agents, crowded</td>
<td>86.8±0.8%</td>
</tr>
</tbody>
</table>

Scene setting: Sparse scene task with \( n_{\text{agent}} = 6 \) and \( \rho_{\text{agent}} = 0.06 \); normal scene task with \( n_{\text{agent}} = 8 \) and \( \rho_{\text{agent}} = 0.08 \); and crowded scene task with \( n_{\text{agent}} = 10 \) and \( \rho_{\text{agent}} = 0.1 \). Set the maximum allowable steps taken \( T_{\text{max}} = 40 \).
As shown in Table 1. NC has success rates $\beta$ of $83.3\% \pm 0.9\%$ and $81.3\% \pm 0.5\%$ on Sparse and Crowded respectively. In the same size map environment, as the number of agents increases, the independent controller structure we use to guide each agent training effectively improves the dimensional explosion problem. As the number of agents increases on the same size map, independent controllers can help IMANet models scale to large teams of agents.

In Figures 9(a) and 9(b), the average reward value $r_{\text{average}}$ of NC under the three task situations of sparse, normal and crowded is close to 4.5. However, the average reward value of IMANet is close to 7.4, and the performance of IMANet is better than NC in all three density levels. This shows that attentional communication can improve the collaboration between agents in the cooperative task scenario.

5.3 baseline

We conducted experiments comparing IMANet with NC and CommNet in predator and prey tasks and traffic junction task - 1) no communication, 2) communication, but broadcast communication, and 3) targeted Communication. and set up task scenarios of different size, the benefits of communication and attention increase with the complexity of the task space. The comparison of IMANet with baseline work is shown in Table 2.

<table>
<thead>
<tr>
<th>Communication Category</th>
<th>Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommNet</td>
<td>Broadcast Communication Decentralized</td>
</tr>
<tr>
<td>NC</td>
<td>No Communication Decentralized</td>
</tr>
<tr>
<td>IMANet</td>
<td>Targeted Communication Decentralized</td>
</tr>
</tbody>
</table>

In the Predator and Prey environment, we set up three grid worlds of different sizes and different numbers of cooperative predators, and we evaluated IMANet against CommNet and NC on small, medium and large scales. The observation horizon of the agents is effective, thus emphasizing the importance of communication.

Scene setting: The small task has 3 agents on a $4 \times 4$ grid with $T_{\text{max}} = 20$, the medium task has 5 agents on a $8 \times 8$ grid with $T_{\text{max}} = 40$ and the
large task has 8 agents on a 15×15 grid with $T_{\text{max}} = 75$.

We compare the convergence effect of the average step taken $T_{\text{steps-taken}}$ in 1000 epochs and the success rate $\beta$ of IMANet, CommNet and NC.

Figure 10 shows the fast and slow convergence of the step taken as well as the smoothness, and Tables 3 and 4 report and evaluate the success rate and average step taken of the three algorithms at different scales, respectively.

![Figure 10: Average steps taken $T_{\text{steps-taken}}$ to complete an episode of Predator-Prey environment](image)

(a) Average step taken $T_{\text{steps-taken}}$ at Small-scale. (b) Average step taken $T_{\text{steps-taken}}$ at medium-scale. (c) Average step taken $T_{\text{steps-taken}}$ at large-scale.

**TABLE 3:** Success rates of cooperative tasks with three different settings in a Predatory-Prey environment.

<table>
<thead>
<tr>
<th>Predatory-Prey (Success Rate)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>map size ($n_{\text{agent}}$)</td>
<td>4×4, 3 agents, small</td>
<td>8×8, 5 agents, medium</td>
<td>15×15, 8 agents, large</td>
</tr>
<tr>
<td>CommNet</td>
<td>99.0±0.9%</td>
<td>86.6±10%</td>
<td>49.0±0.4%</td>
</tr>
<tr>
<td>NC</td>
<td>98.0±0.6%</td>
<td>92.6±0.5%</td>
<td>43.9±6.9%</td>
</tr>
<tr>
<td>IMANet</td>
<td>99.9±0.1%</td>
<td>99.0±0.9%</td>
<td>67.3±6%</td>
</tr>
</tbody>
</table>

**TABLE 4:** Avg. number of steps taken to complete the episode in three different environment sizes settings

<table>
<thead>
<tr>
<th>Predatory-Prey (Avg. Steps)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>map size ($n_{\text{agent}}$)</td>
<td>4×4, 3 agents, small</td>
<td>8×8, 5 agents, medium</td>
<td>15×15, 8 agents, large</td>
</tr>
<tr>
<td>CommNet</td>
<td>4.1±0.01</td>
<td>18.1±0.13</td>
<td>74.7±0.04</td>
</tr>
<tr>
<td>NC</td>
<td>4.0±0.00</td>
<td>18.7±0.04</td>
<td>73.9±0.06</td>
</tr>
<tr>
<td>IMANet</td>
<td>4.0±0.00</td>
<td>17.6±0.03</td>
<td>58.3±0.09</td>
</tr>
</tbody>
</table>
In the traffic junction task, we can test what communication methods the agent performs during the task to avoid collisions while going through the intersection. At any time, the maximum number \( N_{\text{max}} \) of car is set at the intersection, each time a new car is added to the environment with a probability \( p_{\text{arrive}} \) from either of the different directions. We study IMANet with CommNet and NC in both easy and difficult traffic junction environments.

Scene setting: The easy version is a connecting junctions consisting of two \( 7 \times 7 \) one-way roads with \( N_{\text{max}} = 4 \), \( p_{\text{arrive}} = 0.25 \) and \( \tilde{T}_{\text{max}} = 20 \). The difficult version is a four-connection intersection consisting of four \( 15 \times 15 \) two-way roads with \( N_{\text{max}} = 14 \), \( p_{\text{arrive}} = 0.05 \) and \( \tilde{T}_{\text{max}} = 80 \).

In order to analyze the impact of target communication on car through intersections in traffic scenarios, we use success rate \( \hat{\beta} \) evaluation of IMANet with CommNet and NC, Tables 5 report and evaluate the success rate.

**TABLE 5: Success rate on various difficulty levels**

<table>
<thead>
<tr>
<th>Traffic junction (Success Rate)</th>
<th>Level</th>
<th>Easy</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommNet</td>
<td></td>
<td>93.2±2.5%</td>
<td>53.4±3.7%</td>
</tr>
<tr>
<td>NC</td>
<td></td>
<td>74.0±0.9%</td>
<td>50.3±0.7%</td>
</tr>
<tr>
<td>IMANet</td>
<td></td>
<td>93.0±5.2%</td>
<td>68.3±7.8%</td>
</tr>
</tbody>
</table>

**5.4 Analysis.**

As shown in Table 3, on small, both commNet, NC and IMANet get close to 99.9% , a without communication baseline NC has success rates of 43.9±6.9% on large-scale, a broadcast communication baseline CommNet has success rates of 49.0±0.4% on large-scale, targeted communication IMANet has success rates of 67.3±6% on large-scale, which is an 12% absolute improve over CommNet.

As shown in Table 5, in the traffic junction experiment, there can be no good performance without communication. The baseline NC proves this. NC has success rates of 74.0±0.9% and 50.3±0.7% on easy and difficult respectively. On easy, both CommNet and IMANet get close to 93%. IMANet has success rates of 68.3±7.8% on difficult, which is an 10% absolute improve over CommNet.

Figure 10 shows the average step taken convergence efficiency on cooperation task in Predator and Prey environment. As shown in Figures 10(a) and 10(b), on small-scale, the average step taken convergence efficiency of IMANet, CommNet and NC are all close to 95%, on medium-scale, the average step convergence efficiency obtained by CommNet is 71%, and the average step efficiency obtained by NC and IMANet is close to 92%. As shown in Figure 10(c), on large-scale, the average step taken convergence efficiency of IMANet is 62%, which is much higher than the CommNet and the IC. As shown in Table IV, IMANet agents take 58.3 steps to capture prey on average vs. 73.9 for NC vs. 74.7 for CommNet, IMANet largely outperforms all the baselines. The above experiments verify that our method can be effectively extended to large-scale agent teams.

**NC VS IMANet.** This communication-free approach has the advantage that it can cope well with the increase or decrease of agents, but it cannot observe the information of other agents and cannot coordinate the actions of agents. IMANet outperforms NC, we can see communication indeed helps. **CommNet VS IMANet.** However, CommNet also has communication, Why does it perform so much worse in a large-scale task space? CommNet uses centralized training and decentralized execution, where the policies of multi-agents form a large network. In this structure, all hidden layer vectors are stacked though broadcast communication to generate a communication vector. Input the joint observation value of the agent, and the output is the action of all the agents. The action generation of this structure is based on a joint policies. As the number of agents increases, the number of observations increases linearly and
the space of joint actions increases exponentially. Therefore, as shown in Figure 10(b), the average step taken learning training process for the baseline experiment appears to be unstable. Our IMANet uses an independent LSTM network as a controller to guide the generation of action strategies for the agents. The agent uses an attention unit to weight and aggregate the hidden layer vectors passed by each agent, and since each agent pays more attention to information that is more similar to its own observations, the agent has continuity in its observations and so does the generated communication messages, which makes the training process smoother. This decentralized execution allows the agents to generate appropriate actions based on their own hidden layer states. This independent network structure allows each agent to update its policy according to its own reward function. It can effectively deal with the problems caused by the increase in dimensionality. Through the above experiments, we can conclude that our content-based optimization method IMANet has better scalability and stability in the cooperative agent setting compared to the traditional broadcast communication method. It can better handle the problems that arise with the increase of task space.

6. Conclusions

We propose IMANet to learn the communication between agents in a fully cooperative multi-agent task. In IMANet, we embed attention units based on query vectors. The attention units can compress the state values of the hidden layer more efficiently by assigning different attention sizes, and since the observations of the agents have continuity, the generated communication information also has continuity, which makes the training process smoother. Also this independent network structure allows each agent to update its policy according to its own reward function. This decentralized execution allows the agents to generate more beneficial value actions based on their own hidden layer states. The algorithm performs significantly better than the other two algorithms in experiments with Predator and Prey and traffic junction. Moreover, the agent remained significantly more scalable in a larger task space. Attention-based communication can indeed help the agent to perform tasks more effectively in the MARL environment.

Data Availability

The data used to support the findings of this study are available from the authors upon request.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Dalian University Research Platform Project Funding: Dalian Wise Information Technology of Med and Health Key Laboratory, the National Natural Science Foundation of China: Research on the stability of multi-surface high-speed unmanned boat formation and the method of cooperative collision avoidance in complex sea conditions, NO.61673084.

References


