Targeted Multi-Agent Communication with Deep Metric Learning

Hua Miao, Nanxiang Yu

***Abstract*—A novel targeted multi-agent communication model based on deep metric learning (DMLTarMAC) is proposed in this paper. The nonlinear relationship between the internal state of an agent and the received message is described by the deep metric learning (DML) module in DMLTarMAC. Compared with the scheme using a linear relationship, DMLTarMAC can improve the accuracy and effectiveness of the receiver's attention. In order to reveal the advantage of the proposed DMLTarMAC, it is evaluated in cooperative and competitive multi-agent tasks with different difficulty levels and environment settings. The experimental results show that DMLTarMAC outperforms the benchmarks, especially in challenging settings. Furthermore, the ablation experiments demonstrate that agents' communication and behavior strategies are effective and intuitive.**

***Index Terms*—Deep Reinforcement Learning, Deep Metric Learning, Targeted Communication, Multi-Agent Systems.**

I. Introduction

COMMUNICATION is the crucial symbol of the intelligent group. Effective communication can ensure the stability of the systems and improve overall productivity. In multi-agent systems with partially observable environments, agents need to share information with their neighbors to achieve cooperative goals efficiently.

Recently, deep reinforcement learning (DRL) has been applied to the field of multi-agent communication [1]–[4], where the communication protocols are determined by DRL instead of predefined rules. The learned protocols are used to answer when and how to communicate. However, as many researchers have pointed out, numerous undifferentiated messages lead to poor performance in communication and harm the cooperative goals when the scale of the multi-agent network increases. In order to manage complex communication effectively, the attention mechanism was introduced as the primary means of message selection and processing in some promising solutions [5]–[8].

Targeted Multi-Agent Communication (TarMAC) [7] was the most prominent of these models, which focused on the attention mechanism of the message receiver. The critical soft attention component in TarMAC originated from the self-attention layer in Transformer [9]. It is well known that Transformer was originally proposed to be applied in natural language processing. The input of the self-attention layer was word vectors, which were considered to have linear relationships in word vector space [10]–[12]. However, the linear description of the self-attention layer in Transformer is unlikely to be applicable to nonlinear relationships in multi-agent communication.

In this paper, a novel DMLTarMAC is proposed to further improve the attention mechanism for the message receiver. DMLTarMAC introduces deep metric learning [13]–[15] into multi-agent communication learning for the first time to emphasize and systematically describe nonlinearity. With the help of this implicit supervisory signal generated by deep reinforcement learning, the deep metric learning module can be easily integrated into the existing multi-agent communication learning framework. Therefore, by deep metric learning, DMLTarMAC can describe and characterize the complex nonlinear relationship between the agent's internal state and the incoming message and improve the efficiency of targeted multi-agent communication.

The experiments are conducted in three types of environments with different difficulty settings. The experimental results show that DMLTarMAC outperforms the benchmarks with the performance advantage increases with the increase of complexity. The demonstration of the episodes reveals that DMLTarMAC learns more effective communication and cooperation strategies than its ablation without deep metric learning, verifying that DMLTarMAC has better performance and generalization ability.

II. Related Works

A. Multi-agent Communication Learning

Unlike traditional predefined communication, the agents in communication learning develop communication protocols through learning. The learned protocol determines not only the content and timing of communications but also the source and target of communications.

The earliest research in this field can be traced back to the work of Kasai [16] and Giles [17] et al. In their works, the agent learned the communication content through reinforcement learning and genetic algorithm to solve the predator-prey task.

Hausknecht et al. [18] proposed the Deep Recurrent Q-Network (DRQN) to solve the partially observable Markov

Manuscript received October 1, 2022; revised April 20, 2023.

This work is jointly sponsored by National Natural Science Foundation of China under Grant 61673080, Science and Technology Research Program of Chongqing Municipal Education Commission, China under Grant KJZD-K202000601, Chongqing Talent Plan, China under Grant cstc2021ycjhbgzxm0044, the Graduate Student Research Innovation Project of Chongqing, China Grant CYB22245.

Hua Miao a PhD candidate at the School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, China. He is also a lecturer at the School of Computer Science and Engineering, Chongqing University of Technology, Chongqing, China (e-mail:miaohua@cqut.edu.cn).

Nanxiang Yu is a lecturer at the Key Laboratory of Intelligent Analysis and Decision on Complex Systems, Chongqing University of Posts and Telecommunications, Chongqing, China (e-mail:yunx@cqupt.edu.cn).

decision process (POMDP) of a single agent. Foerster et al. extended DRQN to the multi-agent environment and presented Deep Distributed Recurrent Q-Networks (DDRQN) [19]. DDRQN successfully realized multi-agent communication learning for the first time. Based on DDRQN, Foerster et al. developed Differentiable Inter-Agent Learning (DIAL) [1] by establishing a differentiable channel for communication action. This differentiable channel allowed the gradient and feedback information to be pushed to its neighboring agents to improve the training effect.

At the same time, Sukhbaatar et al. reported a similar multi-agent reinforcement learning communication network called CommNet [2]. CommNet uses continuous and differentiable communication channels to learn and train communication actions through the back-propagation algorithm. By simply averaging all the messages on the shared channel, the agents exchanged information sufficiently and improved cooperation efficiency.

Mao et al. [20] imposed the limited-bandwidth restriction in multi-agent communication learning to apply the above methods to real-world systems. They integrated their approach into the real-world packet routing system. In addition to bandwidth-related constraints, SchedNet [21] considered sharing the communication medium, mainly when agents communicate over wireless channels. Different from the methods of downsizing the communication group via a scheduler, Informative Multi-Agent Communication (IMAC) [22] compressed communication messages from the information theory perspective to save bandwidth.

B. Attention in Multi-agent Communication

In multi-agent communication learning, many unprofitable messages are broadcast at each time step. These redundant messages not only bring problems such as limited bandwidth and communication delay but also impair the communication learning process. In order to solve these problems, attention-based solutions were introduced. The first attentional communication model, ATOC [5], encoded local observation and action intention as thought. Based on the thought, the attention module decided whether and whom to communicate with at each step.

In the light of CommNet, Singh et al. proposed Individualized Controlled Continuous Communication Model (IC3Net) [6] by implementing a hard attention mechanism for the message sender. The agent in IC3Net decided when to communicate, and the communication vector was calculated by averaging the hidden states of other active agents gated by the Communication-Action module. The results showed that IC3Net has better training efficiency than the simple continuous communication models and can be applied to mixed or competitive settings.

Meanwhile, Das et al. offered TarMAC, applying a soft attention mechanism, for the message receiver. Each agent in TarMAC learned what message needed to be sent and who would be the receiver of the message. TarMAC can be easily integrated with the attention model of the sender, such as IC3Net, which can improve performance and reduce sample complexity in a mixed or competitive environment.

Recently, Structured Attentive Reasoning Network (SARNet) [23] employed an attention unit to inform the agent of the most critical entities at each time step. With guidance from the attention unit, SARNet used previous memories to extract the shared information that was the most relevant at the reasoning stage. Multi-Agent Graph-attentIon Communication (MAGIC) [8] utilized a Scheduler consisting of a graph attention encoder and a differentiable hard attention mechanism to decide when to communicate and whom to communicate with. In Target-oriented Multi-agent Communication and Cooperation(ToM2C) [24], each agent inferred the intention and observation of other agents according to the Theory of Mind (ToM) module. With the blessing of this ”social skill”, the agents decided when and with whom to share their intention to achieve sub-goals and reach team consensus. The agent of the Multi-Agent Incentive Communication (MAIC) [25] learned the targeted teammate model according to its local messages. Using a mutual information regularizer, this model can generate targeted messages for relevant agents without expanding the policy space. Li et al. used mutual information to characterize attention in a multi-agent environment and proposed Progressive Mutual Information Collaboration (PMIC) [26] for more effective MI-driven cooperation. Since the mutual information measured the relationship between global states and joint actions in their work, the core idea of PMIC was to maximize mutual information about positive cooperative behaviors and minimize mutual information about poor cooperative behaviors.

C. Deep Metric Learning

Metric learning aims at learning similarity measures from data. It plays a vital role in machine learning, especially in computer vision, and has many applications [15], [27]–[29]. By mapping input vectors into a feature space, metric learning makes similar samples close and different samples far apart.

Due to the limitation of linear mapping, traditional metric learning suffered from poor performance in capturing the nonlinear relationship of complex data. Deep metric learning (DML), however, is the combination of deep learning and metric learning. It maps samples into a feature space through a multi-layer nonlinear transformation of the deep neural network, which unifies discriminative feature embedding and metric learning into a joint learning framework [13].

In general, DML can be achieved through pair-based losses and classification-based losses. Pair-based losses operate on the relationships between samples in a batch [14], while classification-based losses benefit from class labels and discriminative feature vectors. Almost all the pair-based losses are rooted in the contrastive loss [30] and the triplet loss [31]. The principle behind these two works is that the loss function should narrow the distance between positive samples as much as possible and push the distance between negative samples over a threshold. In light of this fundamental concept, a wide variety of losses, such as lifted structure loss [32], N-pair loss [33], and triplet Center loss [34], have been proposed. These follow-up methods took full advantage of

the structural information on sample pair or hard negative mining [15] through the objective function.

The most typical and essential classification-based loss is softmax loss. A-softmax [35], L2-constrained softmax loss [36], and NormFace [37] normalized or constrained the features or weights to get more discriminative features or solve the problem of data imbalance. On the other hand, L-softmax loss [38], AM-Softmax [39], and ArcFace [40] introduced margin into the softmax loss to encourage intra-class compactness and inter-class separability.

III. Model

A. Communication Architecture

DMLTarMAC integrates deep metric learning into the communication framework proposed by TarMAC. The communication architecture of DMLTarMAC is illustrated in Figure 1.

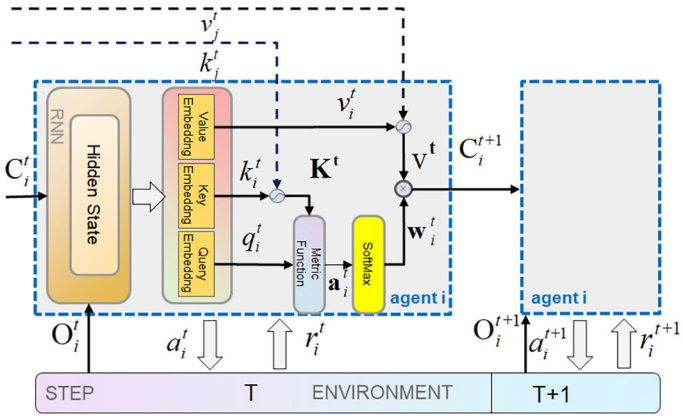

Fig. 1. Communication architecture of DMLTarMAC

In DMLTarMAC, each agent maintains a recurrent neural network (RNN) to hold the memory through its hidden state. At each step t, the i-th agent$(i = 1, 2, \cdots, N)$ makes a discrete action decision a_i^t and sends a continuous communication message m_i^t according to the local observation O_i^t, the weighted communication C_i^t, and the internal state of RNN h_i^t. In other words, the agent interprets the newly received aggregate messages C_i^t and local observations O_i^t according to its internal state h_i^t, and produces the corresponding action a_i^t and the message v_i^t that maximize the cumulative team reward.

The state embedding and message aggregation modules form the core of DMLTarMAC's communication architecture. The state embedding module encodes the internal state of agents into three related vectors: query vector q, key vector k, and value vector v. Then, the generated vectors and the received messages are fed into the message aggregating process to develop the weighted message. The following is a detailed description of these two modules.

State Embedding Module

At each step, the internal state of RNN is updated with the local observations O_i^t and the aggregate communication C_i^t. This internal state is also considered the internal state of an agent. From the system point of view, it encodes all communication and observation sequences of the i-th agent up to time t into hidden state h_i^t in combination with its state.

$$h_i^t = RNN\left(O_i^t \parallel C_i^t, h_i^{t-1}\right).$$

Then, the state embedding module transforms the hidden state into three related feature vectors. The value vector v_i^t is just a linear transformation of the hidden state h_i^t. Unlike TarMAC, DML is used in DMLTarMAC to train the nonlinear mappings φ and τ of the query vector q_i^t and key vector k_i^t.

$$\begin{aligned} &q_i^t = \varphi\left(h_i^t\right), \quad k_i^t = \tau\left(h_i^t\right), \quad v_i^t = \psi\left(h_i^t\right), \\ &q_i^t, k_i^t \in \mathbb{R}^{\mathrm{d}_q}, \quad v_i^t \in \mathbb{R}^{\mathrm{d}_v}. \end{aligned}$$

The query vector q_i^t and key vector k_j^t are designed to measure the correlation between the hidden state of i-th agent and the message from j-th agent. These correlations are then used to weigh received messages and generate the following aggregation message C_i^{t+1}. The value vector v_i^t is the message the i-th agent sends at step t. It is aimed at extracting information needed by relevant agents from its internal state. For the convenience of implementation, the value vector is usually sent with its key vector k_i^t.

$$[k_i^t \parallel v_i^t], \qquad k_i^t \in \mathbb{R}^{\mathrm{d}_q}, \quad v_i^t \in \mathbb{R}^{\mathrm{d}_v}.$$

At the same time, the RNN network delivers a discrete action as its response to the environment according to the internal state. The cardinality of discrete action space is M.

$$a_i^t = \pi\left(h_i^t\right), \quad a_i^t \in [0, 1, \cdots, M-1].$$

Message Aggregating Module

After receiving all the keys from other agents, the i-th agent includes its key to forming a key matrix $\mathbf{K}^t$.

$$\mathbf{K}^t = \left[k_1^t, k_2^t, \cdots, k_N^t\right], \quad \mathbf{K}^t \in \mathbb{R}^{d_q \times \mathrm{N}}.$$

The key matrix $\mathbf{K}^t$ and the query vector q_i^t are input into the metric function to get the attention for incoming messages at step t. This attention essentially reflects the correlation between the internal states of agents.

$$\mathbf{a}_i^t = \mathrm{d}\left(q_i^t, \mathbf{K}^t\right), \qquad \mathbf{a}_i^t \in \mathbb{R}^{\mathrm{N}\times 1}.$$

The attention vector $\mathbf{a}_i^t$ is then fed into the softmax function for stretching and normalization to convert into the attention weights $\mathbf{w}_i^t$.

$$\mathbf{w}_i^t = \mathrm{softmax}\left[\mathbf{a}_i^t\right] = \mathrm{softmax}\left[\left(a_{i1}^t, a_{i2}^t, \cdots, a_{iN}^t\right)^T\right].$$

By multiplying the value matrix $\mathbf{v}^t = [v_1^t, v_2^t \cdots v_N^t]$ by the attention weights $\mathbf{w}_i^t$, the input aggregate message of agent i at step $t+1$ is obtained.

$$C_i^{t+1} = \mathbf{v}^t \cdot \mathbf{w}_i^t = \sum_{\mathrm{j}=1}^{\mathrm{N}} \mathrm{a}_{\mathrm{ij}}^{\mathrm{t}} v_j^t.$$

B. Deep Metric Learning Module

The signature-based soft attention is a fundamental component of TarMAC. In essence, it can be expressed as the scaled inner product of the query vectors and the signatures of incoming messages followed by a softmax function.

$$\mathbf{a}_j = \text{softmax}\left[\frac{\left(\mathrm{W_q} h_{\mathrm{j}}^{\mathrm{t}+1}\right)^T (W_k h_1^t)}{\sqrt{d_k}} \cdots \frac{\left(\mathrm{W_q} h_{\mathrm{j}}^{\mathrm{t}+1}\right) (W_k h_N^t)}{\sqrt{d_k}}\right]$$
$$= \text{softmax}\left[\frac{\mathrm{q}_{\mathrm{j}}^{\mathrm{t}+1^{\mathrm{T}}} k_1^t}{\sqrt{d_k}} \cdots \frac{\mathrm{q}_{\mathrm{j}}^{\mathrm{t}+1^{\mathrm{T}}} k_N^t}{\sqrt{d_k}}\right].$$

Through this soft attention, TarMAC can effectively measure the internal state correlation between agents to achieve better performance. However, in DMLTarMAC, deep metric learning is employed to unify feature embedding and metric learning according to different applications. The nonlinear transformation of deep metric learning can capture the nonlinear relationship of data, which can be used to describe and characterize the complex correlation between the agent state and messages. Equipped with deep metric learning, DMLTarMAC can further improve the accuracy and efficiency of targeted multi-agent communication. Specifically, the scaled inner product used by TarMAC can be considered a particular case of the metric learned by DMLTarMAC.

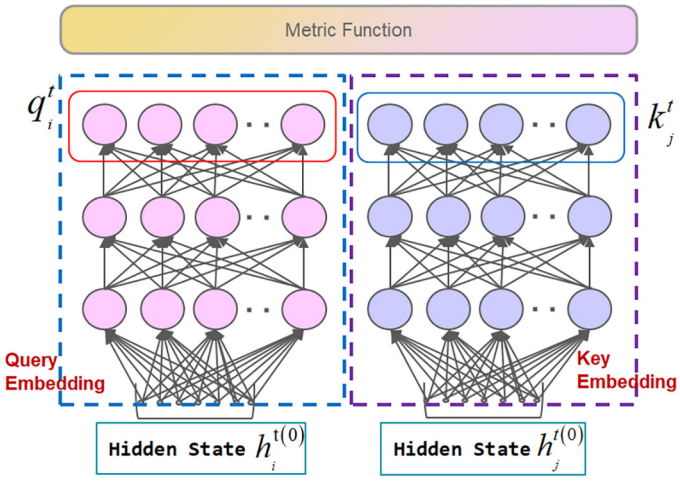

Fig. 2. Illustration of the deep metric learning module using the simple multi-layer perceptron in DMLTarMAC

For simplicity, the deep metric learning module's deep neural network in DMLTarMAC is instantiated as a simple multi-layer perceptron(MLP). And the same neural network structure is applied to the query embedding and key embedding, respectively. The deep metric learning module in DMLTarMAC is shown in Figure 2.

The inputs to the networks are the hidden states of the agent i and agent j at the time step t.

$$x = h_i^{t(0)} \in \mathbb{R}^{r^{(0)}}, y = h_j^{t(0)} \in \mathbb{R}^{r^{(0)}}.$$

The outputs of m-th layer can be represented as:

$$h_i^{t(\mathrm{m})} = \varphi\left(W^{(m)} h_i^{t(\mathrm{m}-1)} + b^{(m)}\right) \in \mathbb{R}^{r^{(m)}},$$
$$h_j^{t(m)} = \tau\left(V^{(m)} h_j^{t(\mathrm{m}-1)} + \mathrm{s}^{(m)}\right) \in \mathbb{R}^{r^{(m)}}, 1 \leqslant m \leqslant M.$$

The matrices $W^{(m)} \in \mathbb{R}^{r^{(m)} \times r^{(m-1)}}$,$\mathbf{V}^{(m)} \in \mathbb{R}^{r^{(m)} \times r^{(m-1)}}$ and the vectors $b^{(m)} \in \mathbb{R}^{r^{(m)}}$, $s^{(m)} \in \mathbb{R}^{r^{(m)}}$ are the weights and biases of the m-th layer of the MLPs. The M is the total number of layers, and the $r^{(m)}$ is the number of neurons in layer m. $\varphi : \mathbb{R}^i \mapsto \mathbb{R}^o$ and $\tau : \mathbb{R}^i \mapsto \mathbb{R}^o$ are nonlinear activation functions(e.g., Sigmod and Relu), where i and o are the dimensions of the input and output vectors, respectively. Thus, the top layer outputs of the MLPs, namely query vector and key vector, can be represented as:

$$q_i^t = f_\theta(x) = h_i^{t(\mathrm{M})} = \varphi\left(W^{(\mathrm{M})} h_i^{t(\mathrm{M}-1)} + b^{(\mathrm{M})}\right) \in \mathbb{R}^{r^{(\mathrm{M})}},$$
$$k_j^t = g_\varpi(y) = h_j^{t(\mathrm{M})} = \tau\left(V^{(\mathrm{M})} h_j^{t(\mathrm{M}-1)} + s^{(\mathrm{M})}\right) \in \mathbb{R}^{r^{(\mathrm{M})}},$$

where the mappings $f : \mathbb{R}^{r^{(0)}} \mapsto \mathbb{R}^{r^{(M)}}$ and $\mathrm{g} : \mathbb{R}^{r^{(0)}} \mapsto \mathbb{R}^{r^{(M)}}$ are nonlinear parametric functions which are determined by the parameters of their feedforward neural networks. The parameter sets can be expressed as follows:

$$\theta = \left\{\mathbf{W}^{(\mathrm{m})}, \mathbf{b}^{(m)}\right\}_{m=1}^{M}, \quad \varpi = \left\{\mathbf{V}^{(\mathrm{m})}, \mathbf{s}^{(m)}\right\}_{m=1}^{M}.$$

With the mappings, f and g, the hidden states of agent i and agent j are mapped to the deep metric space. Thus, the distance between the mapping points of the hidden states in the deep metric space can be expressed as follows:

$$d\left(q_i^t, k_j^t\right) = d\left(f_\theta(x), g_\varpi(y)\right),$$

where d is a metric function such as cosine similarity.

In summary, the feature embedding process of the deep metric learning module in DMLTarMAC is to learn the f and g mappings. With the help of the metric function, these mappings transform the hidden states into a common feature space which encourages intra-class similarity and inter-class separation. In practice, the nonlinear mappings f and g can embrace different network structures according to their applications.

C. Training

DMLTarMAC follows the training framework adopted by IC3Net and TarMAC, which uses Actor-Critic(A2C) [41] as the multi-threaded reinforcement learning algorithm. Different worker threads in training share the same parameters. As shown in Figure 3, DMLTarMAC updates its network parameters with the gradient average of N worker threads.

In each worker thread, DMLTarMAC maintains actor and critic networks. These two networks generate the policy $\pi(a|s)$ and state value $V(s)$ of the agent at each step when the hidden state sequence of the RNN network is given. The critic model ϕ is trained by minimizing the squared error between the estimated value $V_\phi(s)$ and the Monte Carlo return.

$$\arg\min_\phi \sum_{i=1}^{N} \sum_{t=1}^{T} \left(V_\phi\left(s_i^t\right) - R_i^t\right)^2.$$

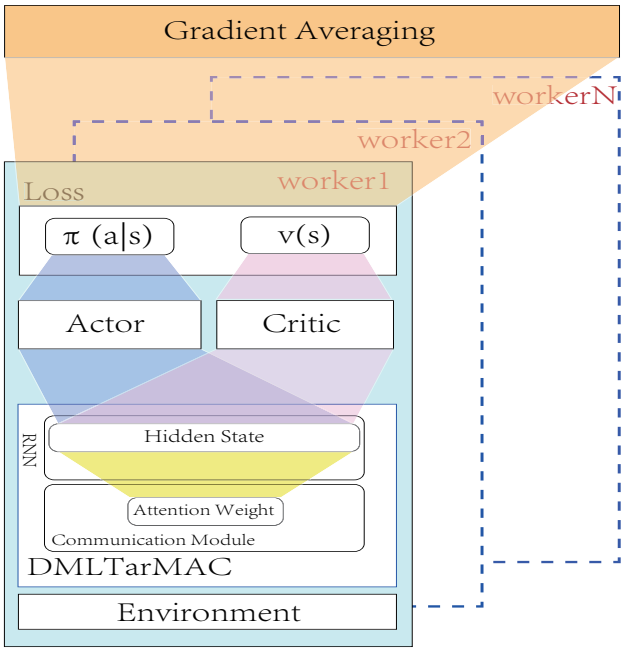

Fig. 3. Illustration of the multi-threaded synchronous training framework of DMLTarMAC

Here, R_i^t is the discounted Monte Carlo return for agent i from step t, and T is the horizon of the episode. Then, the state value function $V(s)$ is introduced into the actor model as a baseline to reduce variance and improve sampling efficiency. Thus, the policy gradient of actor model ϑ can be expressed as the following formula.

$$\nabla J_\vartheta(\vartheta) = \sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\vartheta \log \pi_\vartheta \left(a_i^t | s_i^t\right) \left(Q(s_i^t, a_i^t) - V(s_i^t)\right),$$

$$Q(s_i^t, a_i^t) = r(s_i^t, a_i^t) + V(s_i^{t+1}).$$

Here, $Q(s_i^t, a_i^t)$ is the state-action value and $r(s_i^t, a_i^t)$ is the reward for taking action a_i^t in state s_i^t. Thus, the advantage value $A(s_i^t, a_i^t) = Q(s_i^t, a_i^t) - V(s_i^t)$ is independent of the actor parameters ϑ.

A coefficient λ balances the value and policy losses. Therefore, the gradient of the overall loss function can be defined by the following formula.

$$\nabla_{\vartheta,\phi} L = \sum_{i=1}^{N}\sum_{t=1}^{T} \begin{pmatrix} \nabla_\vartheta \log \pi_\vartheta \left(a_i^t | s_i^t\right) \left(Q(s_i^t, a_i^t) - V_\phi(s_i^t)\right) \\ +\lambda \nabla_\phi \left(V_\phi\left(s_i^t\right) - R_i^t\right)^2 \end{pmatrix}.$$

To minimize this overall loss, as shown in Figure 3, the gradient propagates back to the RNN network along the actor and critic networks. Since the aggregate message C_i^{t+1} is fed into the RNN network as an input, according to the chain rule, it is adjusted with the update of the RNN network parameters. Consequently, this aggregate message indirectly develops into an implicit supervisory signal of the attention weights $\mathbf{a}_i^t$ for all incoming messages. Therefore, the distance between the query vector and key vector in the mapping space can be defined as the attention weight of the agent i to the message sent by agent j:

$$a_{ij}^t = d\left(q_i^t, k_j^t\right) = d\left(f_\theta\left(x\right), g_\varpi\left(y\right)\right).$$

Under this definition, the implicit supervision signal of the attention weight a_{ij}^t will be used as the "distance label" to train the nonlinear mappings f and g in the deep metric learning module of DMLTarMAC. In this way, the DML module is embedded into the framework of TarMAC without explicit labels.

IV. Experiment

DMLTarMAC is evaluated in three multi-agent environments: Traffic Junction [2], SHAPES [7], and Predator-Prey [42]. DMLTarMAC is mainly compared with TarMAC with the same attention communication architecture and CommNet without an attention mechanism. For a fair comparison, DMLTarMAC and benchmarks use the same parameters in the same environment unless specified otherwise. The hidden state dimension of the GRU contained in agents is 128. The dimension of the query and key vector is 16, and the dimension of the value vector is 32. For simplicity, the depth metric learning module in DMLTarMAC uses two layers of MLP for the depth neural network, with 64 and 16 neurons in each layer. All results are averaged over 4 or 5 independent seeds, and error bars are standard deviations of the mean.

A. Traffic Junction

Task and Setting

The Traffic Junction was first introduced in CommNet. In the simulated traffic junction environment, at each step, the car joins the system with the probability p_{arrive} until the total number of cars reaches $\mathrm{N}_{\max}$. Cars travel along a predefined two-way road with one or more intersections. Each car has 3×3 visual range and can communicate with each other freely. On the line, the car can only take the "Gas" or "Brake" action. Even if there is a collision, the car continues to move. After the agent completes its route, it is removed from the grid immediately and sampled back into the environment. If an agent collides, it gets a penalty $r_{collision} = -10$. In other cases, it gets a reward -0.01τ, where the τ is the number of time steps since the car arrived. The comparative experiment is performed in the difficult mode. In the difficult mode, there are four junctions and eight entry points on a 18×18 grid with $\mathrm{N}_{\max}$ = 20.

Results and Analysis

In order to evaluate the impact of metric functions on performance, three popular metric functions in representation learning [43]–[46] are applied in DMLTarMAC. They are the scaled inner product, the bi-linear inner product, and the L2 norm.

The arrival rate p_{arrive} is set to 0.1, twice the TarMAC hard mode setting to make the task harder. Correspondingly, in order to adapt to this difficulty, curriculum learning [47] [48] is used to train the agents. As shown in Figure 4(a), when p_{arrive} = 0.1, the average success rate of DMLTarMAC equipped with the scaled inner product is 89.69%, which is 5.62% higher than that of TarMAC and much higher than that of CommNet. When the game's difficulty decreases,

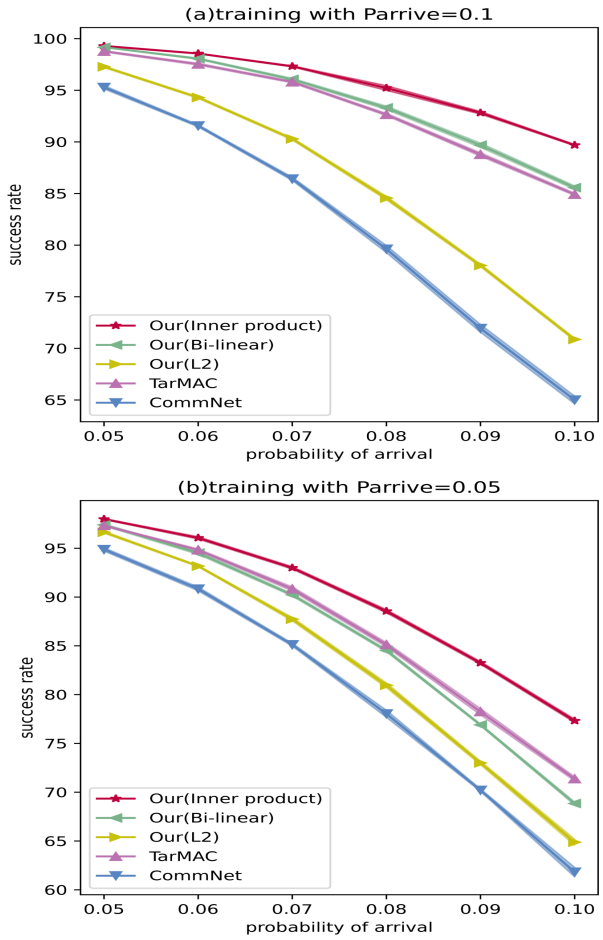

Fig. 4. Average success rates of 6400 episodes under different arrival probabilities. (a) models trained under p_{arrive} = 0.1 (b) models trained under p_{arrive} = 0.05

the success rates of DMLTarMAC and benchmarks begin to converge.

To evaluate the generalization performance of DMLTarMAC, the models trained at p_{arrive} = 0.05 are tested on the arrival rate interval $[0.05, 0.1]$. Figure 4(b) shows that DMLTarMAC using the scaled inner product outperforms the baselines, and the performance advantage increases with the complexity. This result confirms that DMLTarMAC has better generalization performance and can adapt to unknown and difficult situations. It is consistent with the viewpoint mentioned by Karl Cobbe et al. [49] that the larger networks, to some extent, have a higher capacity for generalization in reinforcement learning.

As mentioned earlier, the metric function used in TarMAC is the scaled inner product. The experimental results show that DMLTarMAC achieves the best when using the same metric function as TarMAC. However, when DMLTarMAC adopts the two metric functions of the bi-linear inner product and L2 norm, it achieves comparable and relatively poor results compared with TarMAC. Both cases in Figure 4 indicate that the selection of metric function is the critical factor affecting performance, so appropriate metric functions should be adopted according to different applications.

Ablations

DMLTarMAC uses the scaled inner product as the metric function in the ablation experiment. Figure 5(a) shows the state of the episode at time step 5. In this state, car 3 is closely followed by car 2. As shown in Figure 5(b), in the episode of DMLTarMAC, car 2 is notified by car 3. But in the episode of TarMAC, the message sent by car 3 does not attract enough attention from car 2. Therefore, as shown in Figure 5(c), in the episode of DMLTarMAC, car 2 takes braking action to keep the distance at time step 6. However, in the episode of TarMAC, car 2 continues to follow car 3, causing car 2 and car 3 to collide at step 16. This demonstration shows that the agents in DMLTarMAC have learned the basic driving principle of keeping their distance. And this learned behavior strategy implies that the agents in DMLTarMAC can understand and capture the complex nonlinear relationship between messages and internal states to a certain extent.

Figure 6 shows the communication attention probabilities at the end of the episodes. Compared with TarMAC, the agent in DMLTarMAC is more focused on messages, so it can better perceive the system situation and make corresponding adjustments.

B. SHAPES

Task and Setting

The SHAPES environment was introduced by Das et al. [7] based on the SHAPES dataset [50]. In our task, four agents are navigated to the target with the specified color((red, red, green, blue). 50 * 50 and 100 * 100 maps are used for evaluation, and the corresponding visual range of the agent is set to 5 * 5 and 7 * 7, respectively. The action space for agents is $\{up, down, left, right, stay\}$. The reward for each agent is $r = \frac{number\ of\ agents\ on\ goal}{number\ of\ agents}$, which is a team-based reward.

Results and Analysis

For a more comprehensive assessment, six commonly used metric functions are evaluated and explored in the DML module of DMLTarMAC. They are the scaled inner product, the bi-linear inner product, cosine similarity, Pearson correlation, L2 norm, and L1 norm, respectively.

As shown in Table 1, DMLTarMAC comprehensively exceeds the baselines. As the difficulty increases, the average success rate of DMLTarMAC increases by 8.4% and 18.9%, respectively, compared with TarMAC and CommNet. Not to mention that TarMAC and CommNet can not train successfully in some cases. In particular, because there is only one red shape in the 100 * 100 map and the search space is enormous, the probability of the agents successfully navigating to the red target is low. The fourth row of Table 1 confirms this analysis. If agents are given more search steps, say from 60 to 80 steps, the success rate of the models will significantly improve.

In addition to being consistent with the conclusion of the previous experiment, the experimental results further show that even in different scenarios of the same experiment, the choice of metric function in DMLTarMAC leads to significant performance differences. In most cases of the investigation, the L2 norm and the scaled inner product are pretty choices.

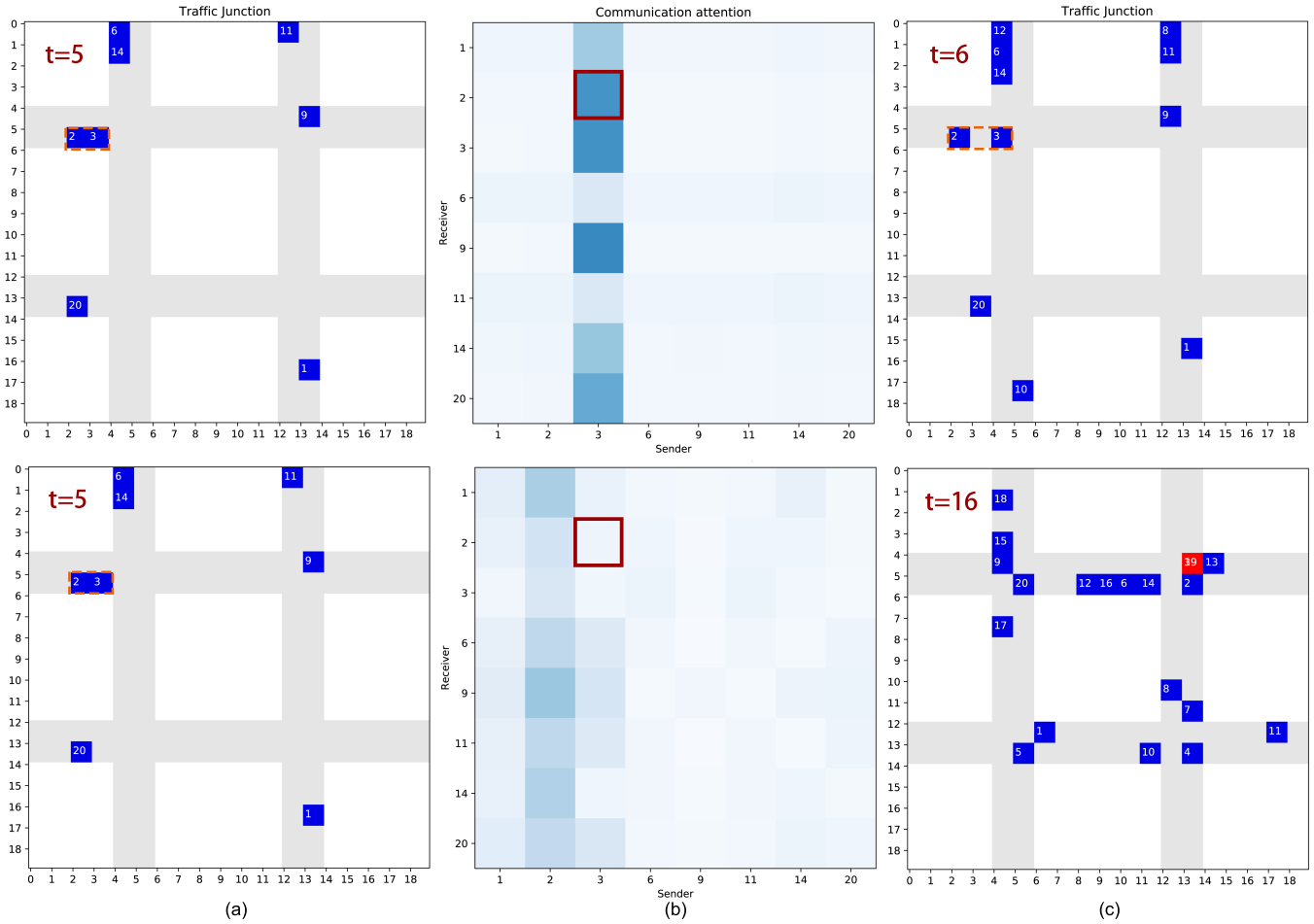

Fig. 5. Illustration of learned cooperative behaviors and communication of DMLTarMAC agents compared to TarMAC agents in Traffic Junction. The first column corresponds to the episode of DMLTarMAC, and the second column corresponds to the episode of TarMAC. Darker color indicates closer communication attention or higher communication attention probabilities.

TABLE I
AVERAGE SUCCESS RATE OF 19200 EPISODES. IF THE SUCCESS RATE IS LESS THAN 30%, IT IS CONSIDERED A TRAINING FAILURE. THE FIRST THREE ROWS OF THE TABLE CORRESPOND TO THE SCENARIOS WITH INCREASING DIFFICULTY.

Scenario	CommNet	TarMAC	DMLTarMAC					
			L1	L2	bi-linear	innerproduct	cosine	Pearson
50*50 step_size=3	93.31±0.07	92.30±0.06	92.71±0.10	92.72±0.07	94.40±0.11	94.16±0.08	93.88±0.09	**96.09±0.06**
50*50 step_size=2	64.72±0.21	28.94±0.18	34.66±0.17	**77.85±0.05**	71.16±0.16	76.94±0.24	72.73±0.11	70.32±0.15
100*100 step_num=60	28.80±0.15	44.07±0.09	41.94±0.20	47.23±0.12	44.27±0.28	**47.78±0.08**	46.05±0.29	45.66±0.12
100*100 step_num=80	28.83±0.13	50.81±0.12	51.08±0.05	**57.50±0.06**	49.85±0.17	55.83±0.09	48.75±0.21	53.37±0.15

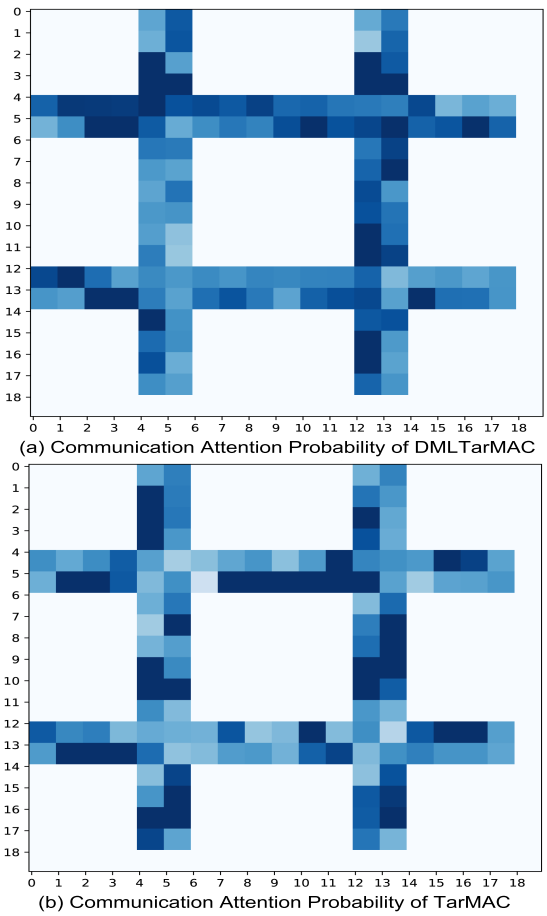

Fig. 6. Comparison of communication attention probability between DMLTarMAC and TarMAC at the end of the episodes. Darker color indicates closer communication attention or higher communication attention probabilities.

However, other metric functions will be better choices in some specific cases, such as scenario 1.

Ablations

As in the previous experiment, DMLTarMAC uses the scaled inner product as the metric function in the ablation experiment. Figure 7(a) shows the state of the episode at the time of step 1. Agent 4 is on the red square. Since the goal of agent 1 and agent 2 is to navigate to the red shape, as shown in Figure 7(b), the message sent by agent 4 attracts the attention of agent 1 and agent 2 in both models. Thus, agent 1 and agent 2 move to the red square, while agent 4 moves toward its target, the blue shape.

However, from step 28, agent 1 and agent 2 in the episode of TarMAC begin to pay less attention to the message sent by agent 4. And this is particularly evident from step 35 (Figure 7(c)(d)). Correspondingly, from then on, agent 1 and agent 2 seem to lose their targets and begin to wander around. This situation continues until the end of the episode, leading to the failure of the task. In contrast, in the episode of DMLTarMAC, agent 1 and agent 2 always focus on the messages sent by agent 4(Figure 7(d)) and move efficiently. At step 31, agent 1 reaches the target and starts to guide agent 2 to move towards the target. At step 58, all agents achieve their goals, completing the task successfully.

C. Predator-Prey

Task and Setting

The Predator-Prey environment was introduced by Lowe et al. [42]. In our task, with random initial locations, six predators with limited vision need to capture the moving prey within 60 steps. Although the predator and prey have the same speed, acceleration, and visual range, it is very difficult for the predator to capture because the prey is trained by MLP. Once the predator bites the prey, the predator gets a reward of 10, and the prey gets a bonus of -10. The prey is confined to a square with an area of 4. Once it leaves this area, it will be severely punished.

In order to provide a relatively stable learning environment in this very competitive task and ensure the success of training, the alternate training method is used to train the neural network of predator and prey separately, and the alternate interval is set to 100 epochs.

Results and analysis

To take the attention mechanism of the sender into account, DMLTarMAC combines IC3Net to decide when to communicate and which messages to attend to. Therefore, DMLTarMAC+IC3Net(Our) is compared with TarMAC+IC3Net, IC3Net, and CommNet. For the convenience of analysis and comparison, the scaled inner product, which has good performance in most cases, is used as a metric function in the DML module of DMLTarMAC. As shown in Figure 8(a), the reward of the predator team in DMLTarMAC+IC3Net is not dominant in the first 300 epochs. However, as shown in Figure 8(b), DMLTarMAC+IC3Net begins to lead other models as the training progresses.. In particular, during the subsequent prey training phases, which imply more difficult challenges, the advantage of DMLTarMAC+IC3Net becomes more obvious.

In the experiment, the capture time of the first predator is defined as the earliest capture time, and the capture time of the last predator is defined as the overall capture time. Similar to the above reward indicators, Figure 9(a)(b) shows that the overall capture time of DMLTarMAC+IC3Net starts to dominate in the second half of training and performs better in difficult situations.

However, as shown in Figure 10(a)(b), DMLTarMAC+IC3Net has no obvious advantage over the benchmark regarding the earliest capture time. As shown in the following observation, this implies that our model gives up the chance of the fastest capture and adopts a more sophisticated and advanced round-up strategy.

Model Observation The three columns (a), (b), and (c) of a row in Figure 11 correspond to three consecutive 20 steps in an episode. As observed in the top row of Figure 11(a)(d), the predator in DMLTarMAC+IC3Net forms a broader attack surface H, forcing the prey to move towards the boundary. Once the prey is forced to withdraw due to out-of-bounds penalties, the predators surround and bite it. Therefore, DMLTarMAC+IC3Net achieves good experimental results.

However, as shown in the bottom row of Figure 11(a)(d), in order to catch the prey as soon as possible, the predators

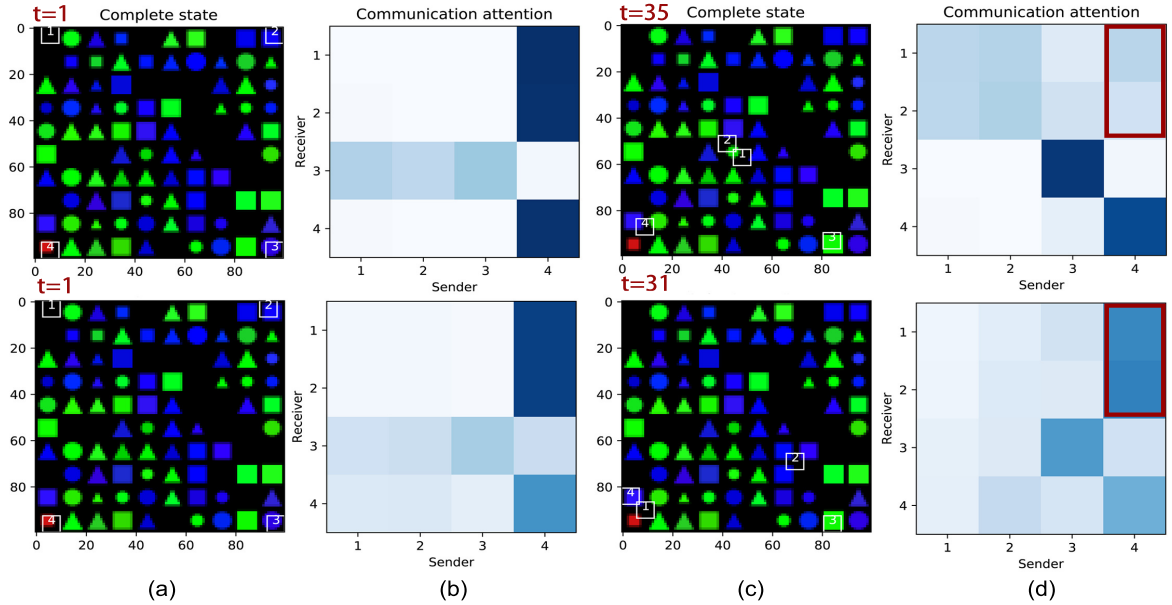

Fig. 7. Illustration of learned targeted communication of DMLTarMAC agents compared to TarMAC agents in SHAPES. The top row corresponds to the episode of DMLTarMAC, and the bottom row corresponds to the episode of TarMAC. Agent 4 is initialized on the red square. Darker color indicates closer communication attention or higher communication attention probabilities.

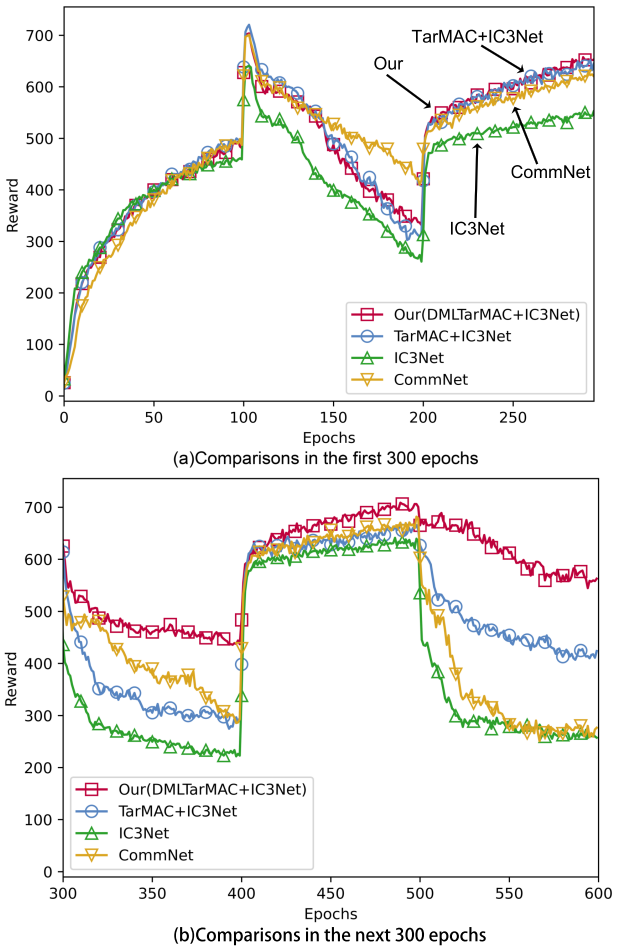

Fig. 8. Average sum of predators rewards in DMLTarMAC+IC3Net against benchmark during alternate training in the Predator-Prey environment.

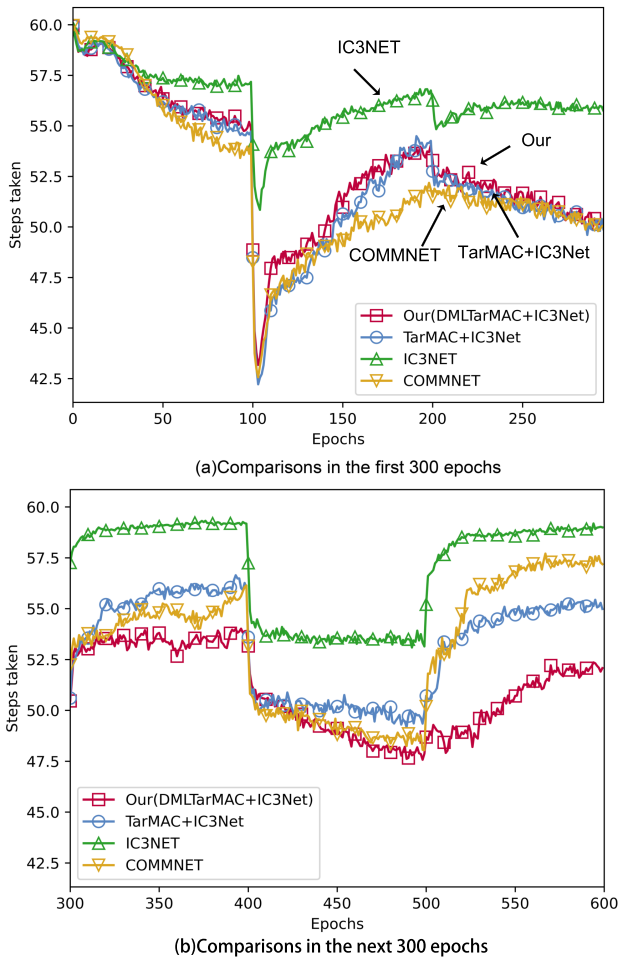

Fig. 9. Average steps of overall capturing time (lower is better) in training.

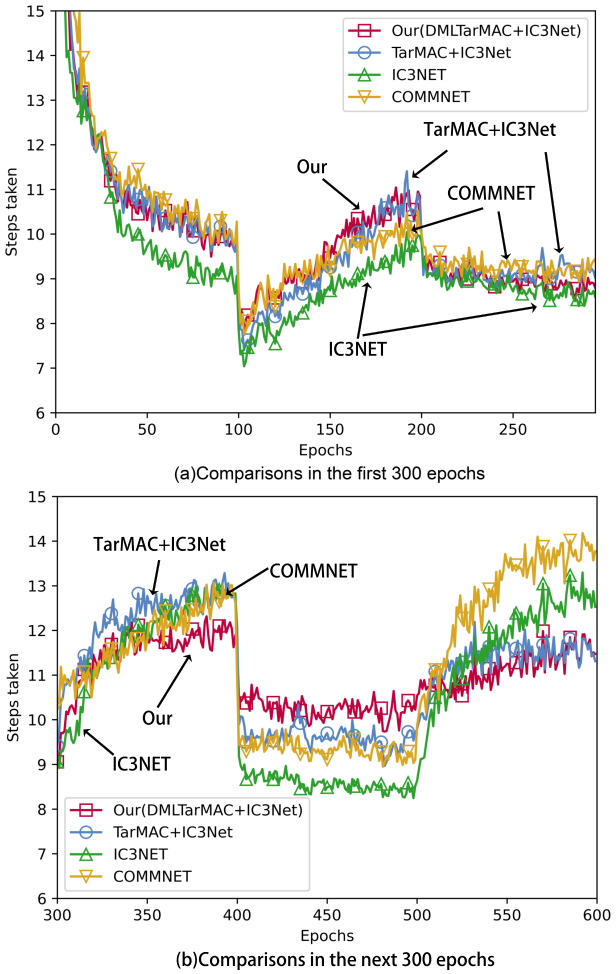

Fig. 10. Average steps of earliest capturing time (lower is better) in training.

in TarMAC+IC3Net adopt the shortest path to the prey, thus forming a narrow attack surface h. Because the prey takes the initiative, it can easily escape from the attack surface of the predator under the premise of the same visual range, speed, and acceleration. As shown in the bottom row of Figure 11(a)-(c), the predators can only follow the prey passively, while the prey leads the predators in circles. In this case, it is difficult for the predators to bite the prey, resulting in relatively poor performance.

V. Conclusion

We introduce the concept of deep metric learning into the attention mechanism for the message receiver in multi-agent communication learning and propose a simple and novel DMLTarMAC. Analysis of three types of environments shows that DMLTarMAC outperforms the existing benchmarks, and its performance becomes more prominent, especially when task complexity and difficulty levels ascend. The ablation experiments imply that DMLTarMAC can learn and describe complex nonlinear relationships of multi-agent communication to some extent. The experimental results further indicate that appropriate metric functions should be used in the DML module of DMLTarMAC for different applications and scenarios. In the future, DMLTarMAC will be explored and applied to the hierarchical communication model to meet the needs of large-scale multi-agent communication.

References

[1] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS 2016)*, 2016, pp. 2145–2153.

[2] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS 2016)*, 2016, pp. 2252–2260.

[3] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang, "Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games," *CoRR*, vol. abs/1703.10069, 2017.

[4] David Simões, Nuno Lau, and Luís Paulo Reis, "Multi-agent actor centralized-critic with communication," *Neurocomputing*, vol. 390, pp. 40–56, 2020.

[5] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018, pp. 7265–7275.

[6] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," in *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019, pp. 1–16.

[7] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "Tarmac: Targeted multi-agent communication," in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, 2019, pp. 1538–1546.

[8] Y. Niu, R. Paleja, and M. Gombolay, "Multi-agent graph-attention communication and teaming," in *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*, 2021, pp. 964–973.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017)*, 2017, pp. 5998–6008.

[10] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the 1st International Conference on Learning Representations (ICLR 2013)*, 2013, pp. 1–12.

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014, pp. 1532–1543.

[12] Carl Allen and Timothy M. Hospedales, "Analogies explained: Towards understanding word embeddings," in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, 2019, pp. 223–231.

[13] J. Lu, J. Hu, and J. Zhou, "Deep metric learning for visual understanding: An overview of recent advances," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 76–84, 2017.

[14] Kevin Musgrave, Serge J. Belongie, and Ser-Nam Lim, "A metric learning reality check," in *Proceedings of the 16th European Conference on Computer Vision (ECCV 2020)*, 2020, pp. 681–699.

[15] Mahmut Kaya and Hasan Sakir Bilge, "Deep metric learning: A survey," *Symmetry*, vol. 11, no. 9, pp. 1–26, 2019.

[16] T. Kasai, H. Tenmoto, and A. Kamiya, "Learning of communication codes in multi-agent reinforcement learning problem," in *2008 IEEE Conference on Soft Computing in Industrial Applications (SMCia 2008)*, 2008, pp. 1–6.

[17] C. Lee Giles and Kam-Chuen Jim, "Learning communication for multi-agent systems," in *Innovative Concepts for Agent-Based Systems, First International Workshop on Radical Agent Concepts (WRAC 2002)*, 2002, pp. 377–392.

[18] Matthew J. Hausknecht and Peter Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 AAAI Fall Symposium Series*, 2015, pp. 29–37.

[19] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson, "Learning to communicate to solve riddles with deep distributed recurrent q-networks," *CoRR*, vol. abs/1602.02672, 2016.

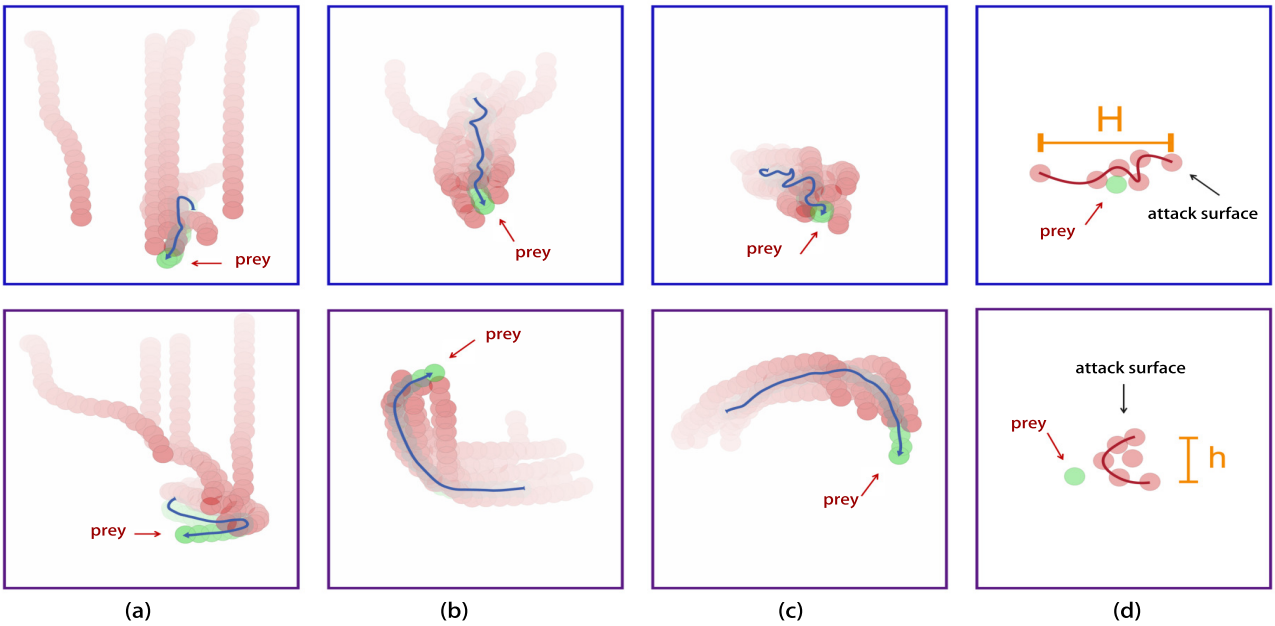

Fig. 11. Illustration of learned cooperative behaviors and strategy of DMLTarMAC+IC3Net agents compared to TarMAC+IC3Net agents in Predator-Prey. The top row corresponds to DMLTarMAC+IC3Net, and the bottom corresponds to TarMAC+IC3Net. Green and red circles are trajectories of prey and predators, respectively, and darker circles are more recent positions. The blue trajectory in (a)-(c) highlights the movement of the prey, and the red curve in (d) highlights the attack surface of the predators.

[20] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, Zhibo Gong, and Yan Ni, "Learning agent communication under limited bandwidth by message pruning," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*, 2020, pp. 5142–5149.

[21] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi, "Learning to schedule communication in multi-agent reinforcement learning," in *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019, pp. 1–17.

[22] Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich, "Learning efficient multi-agent communication: An information bottleneck approach," in *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, 2020, pp. 9908–9918.

[23] M. Rangwala and R. Williams, "Learning multi-agent communication through structured attentive reasoning," in *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020, pp. 10 088–10 098.

[24] Yuanfei Wang, Fangwei Zhong, Jing Xu, and Yizhou Wang, "Tom2c: Target-oriented multi-agent communication and cooperation with theory of mind," in *Proceedings of the Tenth International Conference on Learning Representations (ICLR 2022)*, 2022.

[25] Lei Yuan, Jianhao Wang, Fuxiang Zhang, Chenghe Wang, Zongzhang Zhang, Yang Yu, and Chongjie Zhang, "Multi-agent incentive communication via decentralized teammate modeling," in *Proceedings of the 36th AAAI Conference on Artificial Intelligence(AAAI 2022)*, 2022, pp. 9466–9474.

[26] Pengyi Li, Hongyao Tang, Tianpei Yang, Xiaotian Hao, Tong Sang, Yan Zheng, Jianye Hao, Matthew E. Taylor, Wenyuan Tao, and Zhen Wang, "Pmic: Improving multi-agent reinforcement learning with progressive mutual information collaboration," in *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*, 2022, pp. 12 979–12 997.

[27] Jashila Nair Mogan, Chin Poo Lee, Kalaiarasi Sonai Muthu Anbananthen, and Kian Ming Lim, "Gait-densenet: A hybrid convolutional neural network for gait recognition," *IAENG International Journal of Computer Science*, vol. 49, no. 2, pp. 393–400, 2022.

[28] Yi Li, Dan Yang, and Xi Gong, "Patient similarity via medical attributed heterogeneous graph convolutional network," *IAENG International Journal of Computer Science*, vol. 49, no. 4, pp. 1152–1161, 2022.

[29] Wenqian Qin, Wencang Zhao, and Ming Li, "Multi-level feature representation and multi-layered fusion contrast for few-shot classification," *IAENG International Journal of Computer Science*, vol. 49, no. 2, pp. 318–324, 2022.

[30] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2006 (CVPR 2006)*, 2006, pp. 1735–1742.

[31] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015 (CVPR 2015)*, 2015, pp. 815–823.

[32] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016 (CVPR 2016)*, 2016, pp. 4004–4012.

[33] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS 2016)*, 2016, pp. 1857–1865.

[34] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai, "Triplet-center loss for multi-view 3d object retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018 (CVPR 2018)*, 2018, pp. 1945–1954.

[35] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017 (CVPR 2017)*, 2017, pp. 6738–6746.

[36] Rajeev Ranjan, Carlos Domingo Castillo, and Rama Chellappa, "L2-constrained softmax loss for discriminative face verification," *CoRR*, vol. abs/1703.09507, 2017.

[37] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "Normface: L2 hypersphere embedding for face verification," in *Proceedings of the 25th ACM International Conference on Multimedia (MM 2017)*, 2017, pp. 1041–1049.

[38] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," in *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, 2016, pp. 507–516.

[39] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Process. Lett.*, vol. 25, no. 7, pp. 926–930, 2018.

[40] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition 2019 (CVPR 2019), 2019, pp. 4690–4699.

[41] Y. Wu, E. Mansimov, S. Liao, A. Radford, and J. Schulman, "Openai baselines: Acktr & a2c," *url: https://openai. com/blog/baselines-acktr-a2c*, 2017.

[42] R. Lowe, Y. I. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017)*, vol. 30, 2017.

[43] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, 2020, pp. 5639–5650.

[44] O. Henaff, "Data-efficient image recognition with contrastive predictive coding," in *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, 2020, pp. 4182–4192.

[45] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020 (CVPR 2020)*, 2020, pp. 9726–9735.

[46] Aäron van den Oord, Yazhe Li, and Oriol Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018.

[47] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, 2009, pp. 41–48.

[48] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 4555–4576, 2022.

[49] Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman, "Quantifying generalization in reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, 2019, pp. 1282–1289.

[50] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein, "Neural module networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016 (CVPR 2016)*, 2016, pp. 39–48.