Scalable Communication for Multi-Agent Reinforcement Learning via Transformer-Based Email Mechanism

Xudong Guo , Daming Shi , Wenhui Fan
Department of Automation, Tsinghua University
{gxd20, shidm18}@mails.tsinghua.edu.cn, fanwenhui@tsinghua.edu.cn

Abstract

Communication can impressively improve cooperation in multi-agent reinforcement learning (MARL), especially for partially-observed tasks. However, existing works either broadcast the messages leading to information redundancy, or learn targeted communication by modeling all the other agents as targets, which is not scalable when the number of agents varies. In this work, to tackle the scalability problem of MARL communication for partially-observed tasks, we propose a novel framework **Transformer-based Email Mechanism (TEM)**. The agents adopt local communication to send messages only to the ones that can be observed without modeling all the agents. Inspired by human cooperation with email forwarding, we design message chains to forward information to cooperate with the agents outside the observation range. We introduce Transformer to encode and decode the message chain to choose the next receiver selectively. Empirically, TEM outperforms the baselines on multiple cooperative MARL benchmarks. When the number of agents varies, TEM maintains superior performance without further training.

1 Introduction

Multi-agent reinforcement learning (MARL) has achieved remarkable success in many complex challenges, especially in game playing [OpenAI *et al.*, 2019; Vinyals *et al.*, 2019]. MARL shows great potential to solve cooperative multi-agent real-world tasks, such as autonomous vehicle teams [Shalev-Shwartz *et al.*, 2016], robotics control [Kober *et al.*, 2013] and intelligent traffic control [Wei *et al.*, 2019]. However, some essential obstacles still exist for MARL to reach satisfactory performance. When training the MARL algorithms, the agents keep updating their policies and causing dynamics in the environment, which may hinder the model convergence. Worse still, in most cooperative multi-agent tasks, agents can only observe part of the environment. Partial observability and non-stationarity make it harder to successfully cooperate, even though some works employ centralized training and decentralized execution (CTDE) paradigm to import a critic to coordinate the whole team [Yu *et al.*, 2021; Lowe *et al.*, 2020; Son *et al.*, 2019; Rashid *et al.*, 2018]. Inspired by the ways how humans and animals cooperate, communication is introduced to share information between agents. Some works broadcast the messages to all the other agents [Zhang *et al.*, 2019; Sukhbaatar *et al.*, 2016; Foerster *et al.*, 2016], and other recent works try to learn targeted peer-to-peer communication to reduce the communication bandwidth [Ding *et al.*, 2020; Jiang and Lu, 2018; Yuan *et al.*, 2022]. Attention mechanism from Transformer [Vaswani *et al.*, 2017] is also employed to learn the communication [Jiang and Lu, 2018]. However, the existing methods rely on modeling every teammate in the environment by ID to decide whether to communicate, which will bring huge computational overhead when the number of agents is large. As the modeling network is trained by a specific amount of IDs, the learned communication mechanism is not scalable to reuse when the number of agents changes. In fact, the agent cannot know the state of an agent outside the observation range, and cannot judge whether the information is useful for

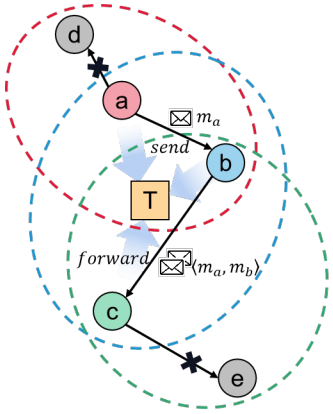

Figure 1: Message chain formed by email forwarding in the Transformer-based Email Mechanism (TEM). The agents (circles) are trying to surround and capture the target (square). The dotted circle is the observation range for the agent with the same color. The black lines are message chains. $\langle \cdot \rangle$ denotes concatenating. The cross denotes not sending or forwarding after receiver selection. The agents *a*, *b* and *c* indirectly cooperate by sending (m_a) and forwarding ($\langle m_a, m_b \rangle$) messages to capture the target *T*. As they are not in the same observation range, forwarding like emails is necessary.

it, so it is unreasonable to directly share information with such an agent. For example, for applications to autonomous vehicles, only the vehicles nearby are worth communicating with to avoid collisions. Moreover, communication with other vehicles should adapt to different numbers of agents as the traffic situation varies a lot.

In this work, to tackle this **new problem** - the scalability of MARL communication, we propose a scalable multi-agent communication mechanism via Transformer-based Email Mechanism (TEM) to tackle the abovementioned challenges as shown in Fig 1. We adopt local communication to send messages only to the agents in the observation range, without modeling all the agents. The agent will decide whom to communicate with by its own intention and by observing the agents in the range. Thus, no matter how the overall number of the agents changes, the learned communication mechanism is scalable. To better utilize the key information and indirectly cooperate with the agents outside the range, we design a **new communication mechanism** like email forwarding to form a message chain. The agents can send and forward the messages so that the chain connects agents from different ranges. For example, the agent *a* in Fig 1 would like to surround and capture the target *T* with other agents, thus the agent *a* may send a message to the agent *b* instead of *d*, though *d* is the nearest. Then the agent *b* can forward the message together with the information from itself to *c*, so that *a*, *b* and *c* can cooperate for the same goal though there is no direct communication between them. Similarly, in our daily life, cooperation in a big company or organization relies on such forwarding emails to share information, as it is always hard to directly find the exact contact in another department.

To suit the unfixed length of the message chain and ensure the communication mechanism is scalable, we design a **new message network** and employ Transformer to encode and decode the sequence of messages. Furthermore, augmented by the attention mechanism in the Transformer, the communication is selective by modeling the correlation between the messages and the observation. The message network is independent and can be plugged into any CTDE method. What's more, we design a loss to guide the agent to estimate the impact of the message on other agents. Note that we do not introduce the broadcast mechanism from email to keep the communication efficient.

For evaluation, we test TEM on three partial-observation cooperative MARL benchmarks: the Starcraft Multi-Agent Challenge (SMAC) [Samvelyan *et al.*, 2019], Predator Prey (PP) [Kim *et al.*, 2019] and Cooperative Navigation (CN) [Lowe *et al.*, 2020], where TEM reaches better performance than the baselines. We also evaluate the scalability of TEM. Without extra training, TEM can suit both situations where the number of agents increases and decreases, and still outperforms the baselines.

In sum, the proposed TEM has three advantages: ① **decentralized** without a central communication scheduler, which is more practical; ② **selective** to send and forward information to the agent who needs it most, while other methods use broadcast which may bring redundant information and the communication cost is high; ③ **scalable** without retraining since TEM does not model all the agents as other methods.

2 Related Works

Learning how to communicate is a popular research domain in multi-agent reinforcement learning. Research in this domain focus mainly on cooperative scenarios, where agents could communicate with each other explicitly. In the early works, RIAL and DIAL [Foerster *et al.*, 2016] are designed to learn communication, where messages are delivered from one timestep to the next timestep in a broadcast way. CommNet [Sukhbaatar *et al.*, 2016] proposes a hidden layer as communication and allows the agents to communicate repeatedly in each step. IC3Net [Singh *et al.*, 2018] brings in the gating mechanism to control communication based on CommNet. Both of BiC-Net [Peng *et al.*, 2017] and ATOC [Jiang and Lu, 2018] implement the communication layer as bidirectional RNN, which inputs the observations of all agents and outputs the action or the integrated thought of each agent. However, these methods either broadcast the messages or rely on a centralized communication layer, which is high-cost and not stable. Communication should not only serve as an integration of information, instead, the agents should share information selectively through peer-to-peer communication.

To avoid broadcasting messages, recent works try to design more intelligent communication mechanisms. CTDE paradigms are also imported to implement decentralized communication. Some works are based on QMIX [Rashid *et al.*, 2018]: VBC [Zhang *et al.*, 2019] proposes a request-reply mechanism and a variance-based regularizer to eliminate the noisy components in messages. TMC [Zhang *et al.*, 2020] maximizes the mutual information between the decentralized Q functions and the communication messages while minimizing the entropy of messages between agents. MAIC [Yuan *et al.*, 2022] allows each agent to learn to generate incentive messages by modeling every teammate and bias other agents' value functions directly. Some are based on another CTDE framework MADDPG [Lowe *et al.*, 2020]: TarMAC [Das *et al.*, 2019] proposes a targeted communication behavior via a signature-based soft attention mechanism. Besides the message, the sender broadcasts a key used by the receivers to gauge the message's relevance. I2C [Ding *et al.*, 2020] learns a prior net via causal inference for peer-to-peer communication. The influence of one agent on another is inferred via the joint action-value function and quantified to label the necessity of peer-to-peer communication. Nevertheless, the methods above need to model every other agent in the environment to achieve individual communication, which is not scalable and practical.

To the best of our knowledge, none of the existing MARL communication methods considers the communication scalability and the forwarding protocol inspired by email.

3 Background

3.1 Policy Gradient (PG) Reinforcement Learning

Policy Gradient (PG) reinforcement learning has the advantage of learning a policy network explicitly, in contrast to value-based reinforcement learning methods. PG methods optimize the policy parameter θ to maximize its objective function $J(\theta) = E_S(V_{\pi_\theta}(s))$. However, due to the variance of environments, it is hard to choose a subtle learning rate in

reinforcement learning. To resolve this problem and ensure the safe optimization of policy learning, the Trust Region Policy Optimization (TRPO) [Schulman *et al.*, 2015] increases the constraint of the parameter difference between policy updates.

Based on TRPO, a simplified version Proximal Policy Optimization [Schulman *et al.*, 2017] is carried out, maintaining the motivation to constrain the learning step while more efficient and easy to be implemented. The object function of PPO can be written as:

$$\mathcal{L}(s,a,\theta_k,\theta) = min[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A_{\pi_{\theta_k}}(s,a), clip(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon)A_{\pi_{\theta_k}}(s,a)], \tag{1}$$

which forces the ratio of $\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$ to locate in the interval $(1-\epsilon, 1+\epsilon)$, so that the new θ is not too far away from old θ_k.

3.2 MAPPO Algorithm

Multi-agent PPO (MAPPO) introduces PPO into the multi-agent scenario [Yu *et al.*, 2021]. MAPPO mainly considers decentralized partially observable Markov decision processes (DEC-POMDP). In an environment with n agents, $s \in S$ denotes the state of the environment. The agent i only has a local observation of environment $o_i = O(s)$ and chooses its action based on its observation and policy $a_i = \pi_i(a_i|o_i)$. The joint action $A = (a_1, ..., a_n)$ denotes the set of actions of all agents. Then, the environment transits its state based on the transition probability $P(s'|s, A)$. In MARL, all the agents will get rewards based on the transition of state and their actions (or more likely joint action) $r_i = \mathcal{R}(s, A)$. Each agent is supposed to get a higher accumulated reward $\sum_t r_i^t$. Therefore, the agents optimize their policy to maximize the discount accumulated reward $J(\theta) = E_{a^t,s^t}[\sum_t \gamma^t \mathcal{R}(s^t, a^t)]$, where $\gamma \in (0, 1]$ is the discount factor.

MAPPO utilizes parameter sharing within homogeneous agents, i.e., homogeneous agents share the same set of network structure and parameters during training and testing. MAPPO is also a CTDE framework, namely, each PPO agent maintains an actor network π_θ to learn the policy and a critic network $V_\phi(s)$ to learn the value function, where θ and ϕ are the parameters of policy network and value network, respectively. The value function requires the global state and only works during training procedures to reduce variance. In our work, we take MAPPO as our baseline and backbone, and add TEM as the communication mechanism into MAPPO.

4 Methods

In this section, we introduce the detailed structure and design of TEM. Before each action decision-making, the agents communicate with each other following the designed protocol, sharing the key information efficiently and selectively. We design a message module based on Transformer to encode the messages received. At the same time, the module is able to decide whether to communicate and whom to communicate with. The message module works together with the original action decision module from MAPPO, to form the actor network in the CTDE structure. The workflow of TEM is presented in Fig 2 and Algorithm 1. We design an independent loss to encourage the message module to maximize the messages' impact on other agents. The whole model has the scalability to transfer from one scenario to another. As the message module is parallel to the action module, our model can be plugged into any CTDE structure.

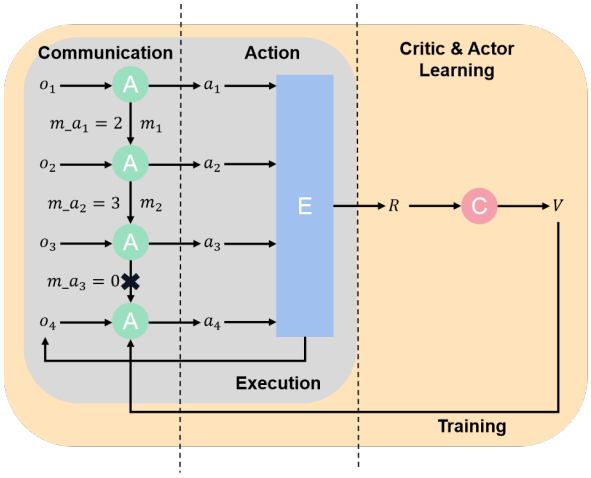

Figure 2: Workflow of TEM during one time step. A denotes the actor network, C denotes the critic network, E denotes the environment. One training step has three phases: communication, action and learning. The execution only includes the first two phases and the critic will not work.

4.1 Communication Protocol Design

We design a communication protocol following the way how humans communicate by email. The information flow is like a forwarding chain: the chain starts with an agent with key information to share, and the following agents merge their own information into the chain and then forward the new message to the next agent. The chain ends when the final agent finds the message useless for others, or there are no more potential communication objects.

When designing the communication protocol, we mainly consider the following questions: (1) Whether to communicate? (2) Whom to communicate with? (3) What to communicate? (4) How to utilize the messages?

(1) Whether to communicate? As shown in Fig 2, in every step of execution and training, the first stage is communication. When the communication stage is done, the actor networks for each agent will make the action decisions by the observations and messages. In the communication stage, each agent has the chance to decide whether to start a new chain and send a message. And the agents who receive messages can decide whether to continue forwarding the messages. Multiple message chains are allowed and the information from different chains is merged if a shared node exists.

(2) Whom to communicate with? For partial-observation (PO) problems, communication with all the agents is not reasonable and effective. The direct communication with the agent outside the observation range may not bring helpful information as the sender does not even know the receiver's

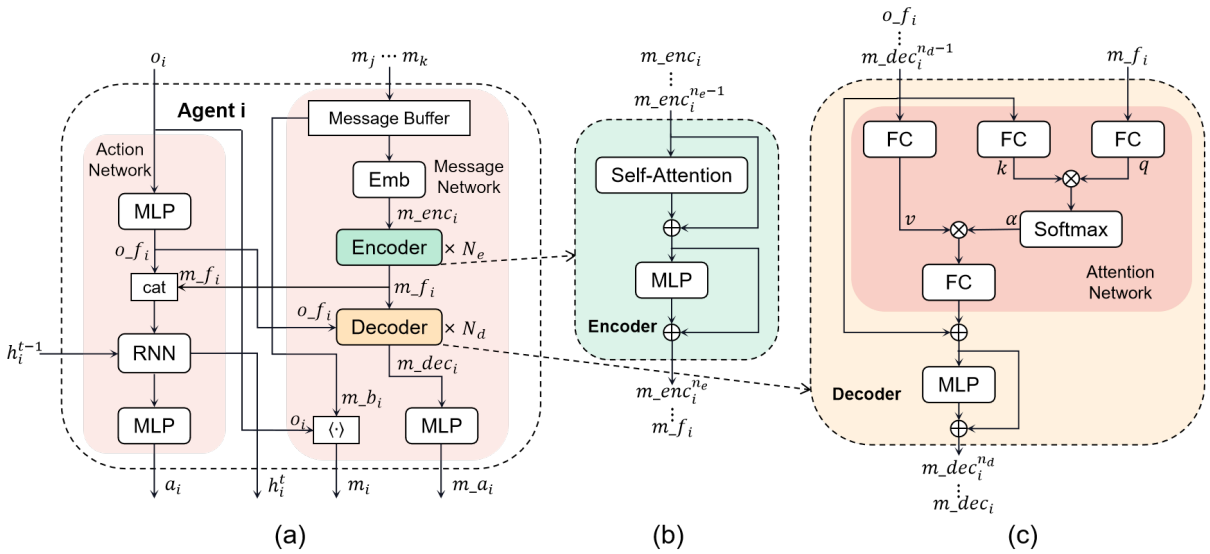

Figure 3: Network structure of TEM. (a) Actor network of agent *i*, including an action network and a message network. Emb denotes the embedding network. (b) Encoder module. (c) Decoder module, where $m\_dec_i^0 = o\_f_i$.

state. Therefore, we do not model all the other agents to decide whether to communicate with them like in some previous works [Ding *et al.*, 2020; Yuan *et al.*, 2022]. Instead, when the agent *i* chooses communication objects, we only consider the agents in the observation range $\mathcal{O}_i$, and in our experiments, $\mathcal{O}_i$ includes the nearest several agents of the agent *i*. By training the message module, the agent can predict the impact of the message on other agents, and is more likely to choose the one with the highest impact to communicate.

We combine the two decisions (1) and (2) into one communication action. The communication actions of agent *i* m_a_i include not sending at all $m\_a_i = 0$, and sending to one agent *j* in the observation range $m\_a_i = j,\ (j \in \mathcal{O}_i)$. Namely:

$$P(m\_a_i = 0) + \sum_{j \in \mathcal{O}_i} (P(m\_a_i = j)) = 1. \tag{2}$$

This way, the agent can decide when and whom to communicate by one action, simplifying the modeling and learning.

Algorithm 1 Execution workflow of agent i in one time step

$stop\_flag_i = 0$, m_re_i =Null, $sender\_list_i$ =Null
for $com\_step_i = 0 \rightarrow max\_com\_steps$ **do**
  Merge the received message m_re_i to the buffer m_b_i, and add the senders of m_re_i to $sender\_list_i$
  if $stop\_flag_i == 0$ **then**
    Get the target m_a_i by o_i and m_b_i
    if m_a_i in $sender\_list_i$ **then**
      $stop\_flag_i = 1$
    end if
    if $m\_a_i \neq 0$ and $stop\_flag_i \neq 1$ **then**
      Send the message to the agent m_a_i as $m\_re_{m\_a_i}$
    end if
  end if
end for
Compute the action a_i by o_i and m_b_i
Interact with the environment to get the new o_i

(3) What to communicate? To keep the information from the head nodes in the chain, and merge the information from different chains, every agent maintains a message buffer to store the messages. In practice, the message buffer is implemented as a queue, with a fixed storage length, but can flexibly push in and pop out elements as the communication goes (First Input First Output, FIFO). We use m_b_i to denote all the messages inside the agent *i* 's message buffer. When sending the new message, the agent *i* merges its own observation into the chain, then the message chain expands to $\langle m\_b_i, o_i \rangle$. Here, the operation $\langle \cdot \rangle$ denotes pushing into the queue to concatenate the messages. The buffer is clear when every step starts.

(4) How to utilize the messages? Instead of some previous works [Zhang *et al.*, 2019; Yuan *et al.*, 2022], we do not think the messages directly influence the value estimation of other agents is the natural way of communication. The information exchange should be separated from the information utilization. And the final effects of the messages should be determined by the receiver instead of the sender. Thus, in our model, messages are taken as a counterpart of the observation, serving as part of the inputs of the actor network.

4.2 Network Design

The schematics of the network design in our model are shown as Fig 3. Each agent has an actor network to observe the environment and communicate with other agents. The actor network of the agent *i* will output the action to interact with the environment a_i, the action to communicate m_a_i (whether to communicate and whom to communicate with), and the corresponding message to be sent m_i. To better utilize the history information and get a smoother action sequence, an RNN is employed in the actor network. Thus the agent *i* also keeps a hidden state h_i^t, and updates it every time step.

The actor network consists of two sub-networks, the action network and the message network. The action network

mainly concentrates on the task itself and tries to get better rewards by outputting reasonable actions. The message network concentrates on the communication to share information with other agents instead. The two sub-networks exchange the representation feature of the observations o_f_i and that of the messages m_f_i to merge the information.

In the action network, o_f_i is learned by a multi-layer perceptron (MLP), and then the action network concatenates o_f_i and m_f_i to input into the RNN together with the hidden state from the last time step h_i^{t-1}. Another MLP, in the end, processes the output of the RNN to get the action a_i.

On the other hand, the messages from other agents like $m_j \cdots m_k$ are stored in the message buffer, like the email inbox. The embedding layer (we implement it as a full connected (FC) layer by practice) converts the messages to fit the input dimensions of the encoders. N_e sequential encoder modules and N_d sequential decoder modules are followed by the embedding layer. The output of encoder modules m_f_i serves as the representation of all the messages in the buffer, with the key information emphasized by the attention mechanism. The decoder modules further combine the information from both of m_f_i and o_f_i to get the output m_dec_i. Finally, one MLP produces the communication decision m_a_i.

For each encoder module, it takes in $m\_enc_i^{n_e-1}$ from the embedding layer or the last encoder, then generates $m\_enc_i^{n_e}$ as the input for the next layer. The transformer in the module can model the sequential information and is flexible to fit message chains with different lengths. Also, the attention mechanism will help the agent to pick out the key information from the chain. n_e implies the position of the layer in the encoder sequence. To prevent gradient vanishing, the encoder module employs the residual connections to link the self-attention mechanism and the MLP [Wen *et al.*, 2022]. The structure of the self-attention mechanism is the same as the attention network in the decoder while k, q and v are generated from the same input $m\_enc_i^{n_e-1}$.

In the decoder module, the first $m\_dec_i^0$ is the representation of the observations o_f_i. In the attention network, full connected layers generates key k and query q by $m\_dec_i^{n_d-1}$ and m_f_i, respectively. Also, the third FC layer generates value v from $m\_dec_i^{n_d-1}$. k and q are used for calculating the weights α of the value v as Equation 3.

$$\alpha = Softmax(exp(\frac{\boldsymbol{q}\boldsymbol{k}^T}{\sqrt{d_k}})). \tag{3}$$

In fact, the weight α learns the correlations between the $m\_dec_i^{n_d-1}$ and m_f_i. By multiplying v and α, then we get the weighted representation of $m\_dec_i^{n_d-1}$ from the ending FC layer. With a similar structure of the residual connections and MLP, we get $m\_dec_i^{n_d}$ as the input for the next layer.

4.3 Loss Function Design

The communication among the agents in a collaborative task aims to share the key information that one believes is useful for some specific agents. So the learning of the message network is driven by the impact of the message to be sent.

As the communication will not change either the action of other agents or the loss of the action network, an independent loss to model the influence of the messages on other agents' actions is needed. We denote the communication loss as $\mathcal{L}_m^{(i)}(\theta)$, where θ is the parameters of the actor network. The action of an agent *j* is sampled from the categorical distribution $P(a_j|o_j, m\_b_j)$ learned by the action network. Then, when considering the new message from the agent *i* m_i, we can estimate the distribution $P(a_j|o_j, \langle m\_b_j, m_i\rangle)$ as the consequence of the communication. Kullback-Leibler (KL) divergence is widely used to measure the discrepancy between these two conditional probability distributions. Thus, the causal effect $\Gamma_j^{(i)}$ of the message from agent *i* on agent *j* can be defined as:

$$\Gamma_j^{(i)} = D_{KL}\left(P(a_j|o_j, \langle m\_b_j, m_i\rangle)||P(a_j|o_j, m\_b_j)\right). \tag{4}$$

By considering all the possible agents to send the message to in the observation range, we can get the expectation of the causal effect of the message $\mathbb{E}\Gamma^{(i)}(\theta)$ by Equation 5:

$$\mathbb{E}\Gamma^{(i)}(\theta) = \sum_{j\in\mathcal{O}_i}\left(P_\theta(m\_a_i = j|o_i, m\_b_i)\Gamma_j^{(i)}\right), \tag{5}$$

where $\mathcal{O}_i$ denotes the observation range of the agent *i*. The communication decision of agent *i* is sampled form the categorical distribution $P_\theta(m\_a_i|o_i, m\_b_i)$ learned by the message network. P_θ denotes that the gradient of this item should be propagated when training.

The expectation $\mathbb{E}\Gamma^{(i)}(\theta)$ represents the overall effect the message m_i can bring to the whole system, which we should maximize in the loss function. However, communication should also be sparse and efficient. If we do not control the communication times by the external guidance, the agents will tend to send as many messages as possible to get higher $\mathbb{E}\Gamma^{(i)}(\theta)$. Therefore, we also designed another item for communication loss to reduce the communication overhead. When the agent *i* chooses not to send the message to any agents in the observation range for most of the times, the probability $P_\theta(m\_a_i = 0|o_i, m\_b_i)$ should be relatively high. So we need to maximize this probability at the same time.

So far, we can get the final communication loss $\mathcal{L}_m^{(i)}(\theta)$ by the following equation and maximize it when training.

$$\mathcal{L}_m^{(i)}(\theta) = \mathbb{E}\Gamma^{(i)}(\theta) + \delta P_\theta(m\_a_i = 0|o_i, m\_b_i), \tag{6}$$

where δ is the weight of the communication reduction.

The loss of the action network $\mathcal{L}_a^{(i)}(\theta)$ is defined followed by Equation 1 in MAPPO as:

$$\mathcal{L}_a^{(i)}(\theta) = min[r_\theta^{(i)}A_{\pi_{\theta_{old}}}^{(i)}, clip(r_\theta^{(i)}, 1-\epsilon, 1+\epsilon)A_{\pi_{\theta_{old}}}^{(i)}], \tag{7}$$

where $r_\theta^{(i)} = \frac{\pi_\theta(a^{(i)}|o^{(i)})}{\pi_{\theta_{old}}(a^{(i)}|o^{(i)})}$, $A_{\pi_{\theta_{old}}}^{(i)}$ is the advantage function.

What's more, to encourage more exploration when training, we adopt an entropy loss $\mathcal{L}_e^{(i)}(\theta)$ as [Yu *et al.*, 2021]:

$$\mathcal{L}_e^{(i)}(\theta) = S(\pi_\theta(o_i)). \tag{8}$$

We can get the overall loss function for the actor network when training:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n}\left(\mathcal{L}_a^{(i)}(\theta) + \lambda_m\mathcal{L}_m^{(i)}(\theta) + \lambda_e\mathcal{L}_e^{(i)}(\theta)\right), \tag{9}$$

where n is the number of the agents, and λ_m, λ_e are the coefficients to weight the corresponding losses.

The loss of critic network keeps the same as MAPPO.

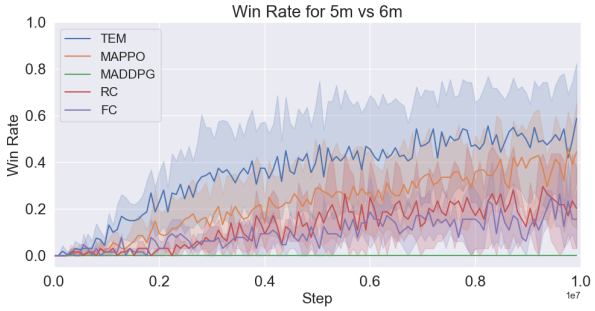

Figure 4: Test win rate for the SMAC map 5m vs. 6m, the shaded regions represent the 95% confidence intervals over 5 seeds. FC: Full Communication, RC: Randomly-stop Communication.

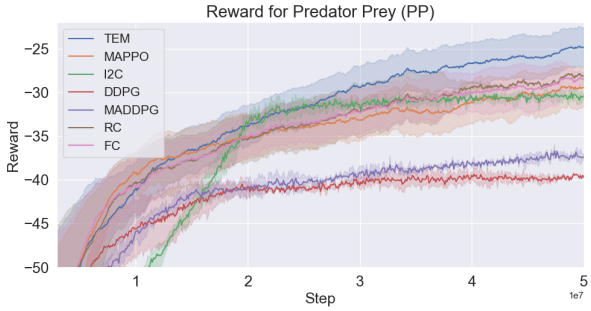

Figure 5: Learning curves for Predator Prey (PP) .

5 Experiments

We evaluate the performance of TEM on three widely-used partially-observed multi-agent cooperative tasks: the Starcraft Multi-Agent Challenge (SMAC), Predator Prey (PP) and Cooperative Navigation (CN). We compare the training process of TEM with the baselines and analyze the performance. We test the scalability of TEM to scenarios with different numbers of agents and targets when zero-shot transferring.

5.1 Starcraft Multi-Agent Challenge

StarCraft II is a real-time strategy game serving as a benchmark in the MARL community. In the Starcraft Multi-Agent Challenge (SMAC) task, N units controlled by the learned algorithm try to kill all the M enemies, demanding proper cooperation strategies and micro-control of movement and attack. We choose the hard map 5m vs. 6m to evaluate TEM. TEM controls 5 Marines to fight with 6 enemy Marines.

The baselines include MAPPO, MADDPG, Full Communication (FC) and Randomly-stop Communication (RC). MAPPO is the CTDE backbone we are using in the following experiments, which is proven to have state-of-the-art performance on several MARL cooperative benchmarks [Yu *et al.*, 2021]. MADDPG is another classic CTDE approach for multi-agent cooperation tasks [Lowe *et al.*, 2020]. FC and RC are two special cases of TEM. We keep the communication protocol the same, but disable the decoder in the message module, instead, the agents choose the communication targets by pre-defined rules. In FC, the agent will keep randomly choosing someone to communicate with, to extend the message chain until no one is available. In RC, the agent will randomly stop the message chain by a probability p, or keep forwarding to a random one.

We run the experiments over five seeds. For each seed, we compute the win rate over 32 test games after each training iteration as shown in Fig. 4. TEM gets the highest win rate over the baselines. FC and RC perform worse than MAPPO benchmark. One possible reason is that targeted communication by TEM could improve cooperation while random communication by FC and RC may bring redundant information for decision-making. The win rate of MADDPG remains zero, showing that it is hard to defeat an army with more units.

5.2 Predator Prey

In the Predator Prey (PP) task, N predators try to chase, surround and finally capture M preys, as shown in Fig 1. The predators are the agents to be trained and the preys flee in the opposite direction of the closest predator at a faster speed following pre-defined rules. So the predators have to be grouped automatically and cooperate to surround each prey, and it is impossible for one predator to capture a prey itself. In practice, we set N as 7 and M as 3, denoted as 7-3 scenario.

Different from the PP task in some previous works, here, the agents can only partially observe the teammates and targets. The rewards are the sum of the agents' negative distances to their closest preys or landmarks. In addition, the agents are penalized for collisions with other agents.

The baselines include MAPPO, I2C, MADDPG, DDPG, FC and RC. I2C proposes an individual communication mechanism [Ding *et al.*, 2020]. DDPG is a classic deep reinforcement learning algorithm for continuous control [Lillicrap *et al.*, 2019]. We apply DDPG independently to each agent as a baseline without considering cooperation.

As shown in Fig 5, while other baselines gradually converge at the last episodes, TEM keeps raising the rewards and improves the final reward by 17.2% compared with MAPPO.

5.3 Cooperative Navigation

In the Cooperative Navigation (CN) task, N agents try to occupy N stationary landmarks separately, as shown in Fig 7. The positions of landmarks and agents are randomly initialized. The best strategy is that each agent has a different target from the beginning through communication instead of rescheduling when collisions happen because of choosing the same target. In practice, we set N as 7, denoted as 7-7 scenario. The baselines and reward settings are the same as PP.

We compare TEM with the baselines on the training performance Fig 6. We can see that TEM converges to the highest

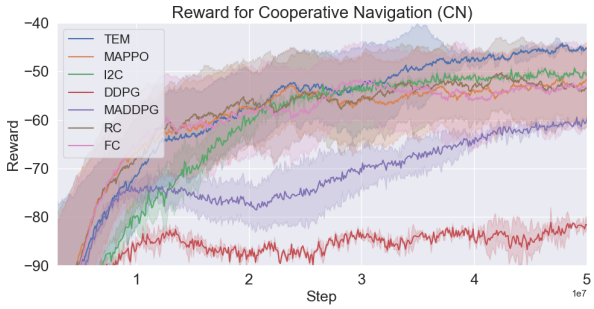

Figure 6: Learning curves for Cooperative Navigation (CN) .

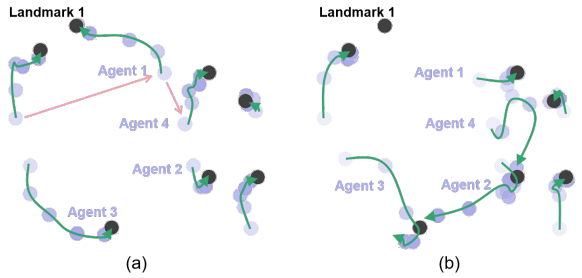

Figure 7: Comparison between (a) TEM and (b) MAPPO on CN. Five timesteps are illustrated. Green: trajectories of the agents (purple) to occupy the landmarks (black). Pink: message chains.

reward than all the baselines. FC and RC are only slightly better than MAPPO, suggesting that the communication actions m_a learned by TEM are targeted, and the message chain brings helpful information to the ones that really need it.

We compare the illustrations on CN between TEM and MAPPO in Fig 7. In (a), the TEM agents Agent 1 and Agent 4 notice Landmark 1 by communication (pink message chain). Thus each agent moves straight forward to the corresponding landmarks. While in (b), the MAPPO agents miss Landmark 1, so for Agent 4, there will be nowhere to go. Agent 4 first tries to scramble with Agent 1 but fails, then turns to Agent 2. Agent 2 is forced to leave to avoid collision and turns to Agent 3. We can see that communication brought by TEM can improve cooperation and reduce internal strife.

5.4 Scalability of TEM

We further examine the scalability of TEM on PP in Table 1. We take average episode rewards (R), successful capture times (S), collision times (C) as the metrics. For R and S, the performance is better when the values are higher, while for C, the performance is better when the values are lower. Note that the existing selective MARL communication approaches are not scalable due to the modeling of each agent, so the baselines are the transferred MAPPO and the specifically trained MAPPO. We directly transfer the learned model from the 7-3 scenario to 9-3 and 3-1 scenarios without further training. For 9-3 scenario, two new agents are included so the task will be easier. But more agents also increase the risks of collision, so the cooperation mode could be different and the agents need to communicate to suit the new scenario. For TEM, the average episode rewards rise from -40.5 to -17.9, and the gain is 55.8%, while for MAPPO, the gain of rewards is 52.3%. TEM does not only perform better after transferring, but also gains more. For 3-1 scenario, both the numbers of the agents and preys change. The results show that TEM still keeps a better performance on all the metrics. Moreover, it shows that after TEM learns how to communicate in a complex scenario, it can successfully transfer to simple ones.

We also train MAPPO from scratch specifically on 9-3 and 3-1 (denoted as MAPPO (learned)), and the performance of transferred TEM (trained on 7-3) is close to MAPPO (learned) on 9-3 without training. But the transferred TEM works worse on 3-1, and we suggest that cooperation by communication may not play an essential role in such a simple environment. We further finetune TEM (7-3) on the new scenarios and the finetuned models even outperform the specially learned MAPPO.

		TEM (7-3)	MAPPO (7-3)	MAPPO (learned)	**TEM** (finetuned)
7-3	R	-40.5±4.7	-44.9±4.3	-	-
	S	61.6±18.3	49.0±16.5	-	-
	C	1.4±0.6	12.6±2.6	-	-
3-1	R	-10.6±4.7	-12.7±5.9	-7.52±2.6	-7.0±2.8
	S	18.7±14.0	13.5±12.7	36.5±13.2	31.7±13.4
	C	1.2±1.2	3.6±2.7	1.8±1.8	0.8 ±0.6
9-3	R	-17.9±5.0	-21.3±7.5	-17.1± 8.2	-14.0 ±2.6
	S	107.2±23.6	69.3±32.0	109.5±18.7	127.9±5.9
	C	16.2±1.8	30.6±6.7	7.2±1.9	5.4±1.6

Table 1: Scalability of TEM on PP. R: average episode rewards, S: successful capture times, C: collision times. TEM (7-3) and MAPPO (7-3) are trained on the scenario 7-3: 7 agents to capture 3 preys, and tested on ten random environments on 7-3, 3-1, 9-3 scenarios. MAPPO (learned) is specifically trained from scratch on the corresponding test environments. TEM (finetuned) is the TEM model trained on 7-3 and tuned on the corresponding test environments.

Similar experiments are conducted on CN as shown in Table 2. TEM keeps the scalability when transferred from 7-7 scenario to 6-6 and 9-9, and outperforms the transferred MAPPO. Surprisingly, the transferred TEM even outperforms the MAPPO trained from scratch (denoted as MAPPO (learned)) on most metrics. It suggests that CN requires more communication to coordinate the agents to explore all the landmarks. The results also show that the communication pattern learned from 7-7 still works well in other scenarios. Similarly, the finetuned TEM gets even better performance.

		TEM (7-7)	MAPPO (7-7)	MAPPO (learned)	**TEM** (finetuned)
7-7	R	-38.8±15.1	-46.6±14.8	-	-
	S	35.8± 6.2	23.3±9.0	-	-
	C	2.8±0.3	4.2±0.2	-	-
6-6	R	-39.8±5.3	-45.0±8.0	-43.6±10.1	-36.7 ±5.2
	S	35.3±6.2	20.0±5.9	19.3±6.8	36.1±5.9
	C	7.2±0.4	8.4±0.4	4.8±0.4	4.8 ±0.2
9-9	R	-45.8±23.9	-57.5±25.3	-50.8±12.9	-41.1 ±11.4
	S	39.4±4.9	26.6±8.7	29.0±6.6	38.9±5.2
	C	9.0±0.3	28.8±0.4	10.8±0.2	8.0±0.2

Table 2: Scalability of TEM on CN. TEM (7-7) and MAPPO (7-7) are trained on the scenario 7-7: 7 agents to occupy 7 landmarks, and tested on ten random environments on 7-3, 6-6, 9-9 scenarios.

6 Conclusions

To tackle the scalability problem of MARL communication, this paper proposes a novel framework Transformer-based Email Mechanism (TEM). The agents adopt local communication to send and forward messages like emails to form message chains, which set up bridges among partial-observation ranges. We introduce Transformer to encode and decode the message chain to choose the next receiver selectively. Empirical results in diverse multi-agent cooperative tasks show that our method outperforms the baselines. Furthermore, we can directly apply TEM to a new environment with a different number of agents without retraining. Better performance than the baselines when zero-shot transferring shows the scalability of TEM. Based on TEM, communication for hundreds of agents and further tailored message generation can be developed, which may be an important step for MARL applications to real-world tasks.

Acknowledgments

This work was supported by the National Key Research & Development Program of China (No. 2021YFC1809003).

References

[Das *et al.*, 2019] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. TarMAC: Targeted Multi-Agent Communication. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1538–1546. PMLR, May 2019.

[Ding *et al.*, 2020] Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning Individually Inferred Communication for Multi-Agent Cooperation. In *Advances in Neural Information Processing Systems*, volume 33, pages 22069–22079, 2020.

[Foerster *et al.*, 2016] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[Jiang and Lu, 2018] Jiechuan Jiang and Zongqing Lu. Learning Attentional Communication for Multi-Agent Cooperation. November 2018. arXiv: 1805.07733.

[Kim *et al.*, 2019] 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. February 2019. arXiv: 1902.01554.

[Kober *et al.*, 2013] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, September 2013.

[Lillicrap *et al.*, 2019] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. July 2019. arXiv: 1509.02971.

[Lowe *et al.*, 2020] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. March 2020. arXiv:1706.02275.

[OpenAI *et al.*, 2019] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning, December 2019. arXiv:1912.06680.

[Peng *et al.*, 2017] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games, September 2017. arXiv:1703.10069.

[Rashid *et al.*, 2018] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning, June 2018. arXiv:1803.11485.

[Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge, December 2019. arXiv:1902.04043.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1889–1897. PMLR, June 2015.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. arXiv:1707.06347.

[Shalev-Shwartz *et al.*, 2016] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving, October 2016. arXiv:1610.03295.

[Singh *et al.*, 2018] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to Communicate at Scale in Multiagent Cooperative and Competitive Tasks, December 2018. arXiv:1812.09755.

[Son *et al.*, 2019] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5887–5896. PMLR, May 2019.

[Sukhbaatar *et al.*, 2016] Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[Vinyals *et al.*, 2019] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff,

Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, November 2019.

[Wei *et al.*, 2019] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. CoLight: Learning Network-level Cooperation for Traffic Signal Control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1913–1922, November 2019.

[Wen *et al.*, 2022] Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem, May 2022. arXiv:2205.14953.

[Yu *et al.*, 2021] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games, July 2021. arXiv:2103.01955.

[Yuan *et al.*, 2022] Lei Yuan, Jianhao Wang, Fuxiang Zhang, Chenghe Wang, ZongZhang Zhang, Yang Yu, and Chongjie Zhang. Multi-Agent Incentive Communication via Decentralized Teammate Modeling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9):9466–9474, June 2022.

[Zhang *et al.*, 2019] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Efficient Communication in Multi-Agent Reinforcement Learning via Variance Based Control. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[Zhang *et al.*, 2020] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Succinct and Robust Multi-Agent Communication With Temporal Message Control. In *Advances in Neural Information Processing Systems*, volume 33, pages 17271–17282, 2020.