

# Communication-robust multi-agent learning by adaptable auxiliary multi-agent adversary generation

Lei YUAN<sup>1,2\*</sup>, Feng CHEN<sup>1\*</sup>, Zongzhang ZHANG<sup>1</sup>, Yang YU (✉)<sup>1,2</sup>

1 National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

2 Polixir Technologies, Nanjing 211106, China

© The Author(s) 2023. This article is published with open access at [link.springer.com](http://link.springer.com) and [journal.hep.com.cn](http://journal.hep.com.cn)

**Abstract** Communication can promote coordination in cooperative Multi-Agent Reinforcement Learning (MARL). Nowadays, existing works mainly focus on improving the communication efficiency of agents, neglecting that real-world communication is much more challenging as there may exist noise or potential attackers. Thus the robustness of the communication-based policies becomes an emergent and severe issue that needs more exploration. In this paper, we posit that the ego system<sup>1)</sup> trained with auxiliary adversaries may handle this limitation and propose an adaptable method of Multi-Agent Auxiliary Adversaries Generation for robust Communication, dubbed MA3C, to obtain a robust communication-based policy. In specific, we introduce a novel message-attacking approach that models the learning of the auxiliary attacker as a cooperative problem under a shared goal to minimize the coordination ability of the ego system, with which every information channel may suffer from distinct message attacks. Furthermore, as naive adversarial training may impede the generalization ability of the ego system, we design an attacker population generation approach based on evolutionary learning. Finally, the ego system is paired with an attacker population and then alternatively trained against the continuously evolving attackers to improve its robustness, meaning that both the ego system and the attackers are adaptable. Extensive experiments on multiple benchmarks indicate that our proposed MA3C provides comparable or better robustness and generalization ability than other baselines.

**Keywords** multi-agent communication, adversarial training, robustness validation, reinforcement learning

## 1 Introduction

Communication plays a crucial role in Multi-Agent Reinforcement Learning (MARL) [1], with which agents can

share information such as experiences, intentions, or observations among teammates to facilitate the learning process, leading to a better understanding of other agents (or other environmental elements) and better coordination as a result. Previous works mainly concentrate on improving communication efficiency from multiple aspects, either by applying (designing) efficient communication protocols [2–5], or combining the nature of multi-agent systems to promote communication [6–8], etc, and have been widely demonstrated to alleviate the partial observability caused by the nature of the environment or the non-stationary caused by the simultaneous learning of multiple agents in a multi-agent system, achieving remarkable coordination for a wide range of tasks like StarCraft Multi-Agent Challenge (SMAC) [8]. However, the mainstream communication methods are still difficult to be applied in the real world, as these methods popularly assume the policies are trained and tested in a similar or identical environment, seldom considering the policy drift led by noise or hostile attacks in the environment.

Let's review the numerous successes achieved in modern Reinforcement Learning (RL). Most approaches depend highly on deep neural networks, which are, however, shown to be vulnerable to any adversarial attacks [9], i.e., any slight perturbation in the input may lead to entirely different decision-making of a Deep Reinforcement Learning (DRL) agent [10]. This poses a significant risk to the application of most DRL algorithms, including MARL communication algorithms, because the noise or hostile attacks in the environment can cause the system to crash. Thus, improving the robustness of the decision system, which means that we hope the system still works well when attacked, is an emergent and severe issue. For the mentioned problem in a single-agent system, many efficient methods are proposed, including adversarial regularizers designing [11,12]. They enjoy theoretical robustness guarantee, but with limited robustness ability [13]. On the other hand, other approaches introduce auxiliary adversaries to promote robustness via adversarial training, model the training process from a game theory

Received December 3, 2022; accepted June 18, 2023

E-mail: [yuy@nju.edu.cn](mailto:yuy@nju.edu.cn)

\* These authors contributed equally to this work.

<sup>1)</sup> Here ego system means the multi-agent communication system itself. We use the word ego to distinguish it from the generated adversaries.

perspective to gain the worst-case performance guarantee and show high effectiveness in different domains [14,15]. As a consequence, the MARL community also investigates the robustness of a multi-agent system from various aspects, including the uncertainty in local observation [16], model function [15], action making [17], etc. However, the communication process in MARL is much more complex. For instance, if we consider a fully connected multi-agent with  $N$  agents, there are total  $N \times (N - 1)$  message channels. If we train an adversary to attack these channels, the attacker's action space may grow dramatically with the number of agents. Previous works make strong assumptions to alleviate this problem, such as some default channels suffering from the same message perturbation [18] or only a limited number of agents sustaining some heuristic noise injection. Despite the difficulty of considering the communication robustness of the multi-agent system, it is an important topic because the communication channels are likely to be under noise or hostile attacks in some application scenarios. This current state motivates us to design reasonable approaches to improve the robustness of the communication system when message attacks are possible in the environment.

In this work, we take a further step towards achieving robustness in MARL communication via auxiliary adversarial training. We posit a robust communication-based policy should be robust to scenarios where **every** message channel may be perturbed under **different** degrees at **any time**. Specifically, we model the message adversary training process as a cooperative MARL problem, where each adversary obtains the local state of one message sender, then outputs  $N - 1$  stochastic actions as message perturbations for each message to be sent to other teammates. For the optimization of the adversary, as there are  $N$  adversaries coordinating to minimize the ego system's return, we can use any cooperative MARL approach to train the attacker system. Moreover, to alleviate the overfitting problem of using a single attacker [19], we introduce an attacker population learning paradigm, with which we can obtain a set of attackers with high attacking quality and behavior diversity. The ego system and the attacker are then trained in an alternative way to obtain a robust communication-based policy. Extensive experiments are conducted on various cooperative multi-agent benchmarks that need communication to coordination, including Hallway [20], two maps from StarCraft Multi-Agent Challenge (SMAC) [20], a newly created environment Gold Panner (GP), and Traffic Junction (TJ) [21]. The experimental results show that MA3C outperforms multiple baselines. Further, more results validate it from other aspects, like generalization and high transfer ability for complex tasks.

## 2 Related work

**Multi-agent communication** Communication is a significant topic in MARL under partial observability, which typically studies **when** to send **what** messages to **whom** [1]. The early relevant works mainly consider combining communication with any existing MARL methods, using broadcasted messages to promote coordination within a team [6] or designing end-to-end training paradigms that update the

message network and policy network together with the back-propagated gradients [22]. To improve communication in complex scenarios, researchers investigate the efficiency of communication from multiple aspects like designing positive listening protocol [23,24]. To avoid redundant communication, some works employ techniques such as gate mechanisms [2,4,25] to decide whom to communicate with, or attention mechanisms [5,21,26,27] to extract the most valuable part from multiple received messages for decision-making. What content to share is also a critical point. A direct practice is to only share local observations or their embeddings [6,20], but it inevitably causes bandwidth wasting or even degrades the coordination efficiency. Some methods utilize techniques like teammate modeling to generate more succinct and efficient messages [8,28,29]. Besides, VBC [28] and TMC [29] also answer the question of when to communicate by utilizing a fixed threshold to control the chance of communication. In terms of the robustness of communication in cooperative MARL, [30] filters valuable content from noisy messages by Gaussian process modeling. AME [31] utilizes an ensemble-based defense method to reach robustness but it only assumes no more than half of the message channels in the system can be attacked. [18] considers the communication robustness in situations where one agent in the cooperating group is taken over by a learned adversary, and then the policy-search response-oracle (PSRO) technique is applied.

**Robustness in cooperative MARL** Previous cooperative MARL [32] works either concentrate on improving coordination ability from diverse aspects like scalability [33], credit assignment [34], and non-stationarity [35], or applying the cooperative MARL technique to multiple domains like autonomous vehicle teams [36,37], power management [38], and dynamic algorithm configuration [39]. Those approaches ignore the robustness of the learned policy when encountering uncertainties, perturbations, or structural changes in the environment [10], hastening the robustness test in the MARL [40]. For the altering of the opponent policy, M3DDPG [41] learns a minimax variant of MADDPG [42] and trains the MARL policy in an adversarial way, showing potential in solving the problem of poor local optima compared with multiple baselines. [43] applies the social empowerment technique to avoid the MARL overfitting to their specific trained partners. As for the uncertainty caused by the inaccurate knowledge of the MARL dynamic model, R-MADDPG [44] proposes the concept of robust Nash equilibrium, treats the uncertainty of environment as a natural agent, and exhibits superiority when encountering reward uncertainty. Consider the observation perturbation, [16] learns an adversarial observation policy to attack one participant in a cooperative MARL system, demonstrating the high vulnerability of cooperative MARL facing observation perturbation. For the action robustness in cooperative MARL, ARTS [45] and RADAR [46] learn resilient MARL policies via adversarial value decomposition. [17] further designs an action regularizer to attack the cooperative MARL system efficiently.

**Population-based reinforcement learning (PBRL)** Population-Based Training (PBT) has been widely used in

machine learning and made tremendous success in different domains [47], which also reveals great potential for reinforcement learning problems [48,49]. One successful application of PBRL is to train multiple policies to generate diverse behaviors that can accelerate the learning of downstream tasks [50]. Another category focus on applying population training to facilitate reinforcement learning in aspects like efficient exploration [51], model learning [52], robustness [19], and zero-shot coordination [53,54]. Among all these works, [19] is most similar to our work, which maintains a population with different individuals by the different network initialization, without an explicit diversity constraint among individuals. However, our work differs because we further consider the robustness of multi-agent communication beyond the single-agent RL setting, and we explicitly optimize the diversity of the attacker population. Actually, if all individuals act indistinctively, the population is just multiple copies of an individual, so one key factor in population-based training is to evaluate individuals effectively and ensure their differences. One series of representative algorithms is evolutionary computation [51,55], and among them the Quality-Diversity (QD) algorithms [56,57] have been widely used to obtain high-quality solutions with diverse behaviors in a single run. High-quality refers to each individual trying to accomplish the given task, while diversity means all individuals behave differently as possible. These methods obtain great success in diverse domains such as skill learning [58], multi-object optimization [59], and skill discovering [60]. Nevertheless, multi-agent problems such as SMAC [61] are much more complex, as how to distinguish agents' cooperation patterns and measure distances between different joint policies are still open questions.

### 3 Problem setting

This paper considers the problem setting of fully cooperative MARL which can be modeled as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [62] consisting of a tuple  $\langle I, S, \mathcal{A}, P, \Omega, O, R, \gamma \rangle$ , where  $I = \{1, 2, \dots, N\}$  indicates the finite set of  $N$  agents and  $\gamma \in [0, 1)$  is the discounted factor. At each timestep, each agent  $i$  observes a local observation  $o_i \in \Omega$ , which is a projection of the true state  $s \in S$  by the observation function  $o_i = O(s, i)$ . Each agent selects an action  $a_i \in \mathcal{A}$  to execute, and all individual actions form a joint action  $\mathbf{a} \in \mathcal{A}^N$  which leads to the next state  $s' \sim P(s'|s, \mathbf{a})$  and a reward  $r = R(s, \mathbf{a})$ . Besides, a message set  $\mathcal{M}$  is introduced to model the agent communication, and the Dec-POMDP can be transformed into a Dec-POMDP with Communication (Dec-POMDP-Comm) [18]. Specifically, each agent makes decision based on an individual policy  $\pi_i(a_i | \tau_i, m_i)$ , where  $\tau_i$  represents the history  $(o_i^1, a_i^1, \dots, o_i^{t-1}, a_i^{t-1}, o_i^t)$  of agent  $i$ ,  $m_i \in \mathcal{M}$  is the message received by agent  $i$  and  $m_{ij}$  indicates the message sent from agent  $j$  to  $i$ . As each agent can behave as a message sender and receiver, this paper uses the default message generator and message processor for various methods like NDQ [20]. Specifically, the messages sent by agent  $i$  are denoted as  $m_i = \text{msg}_i(o_i)$ , where  $\text{msg}_i(\cdot)$  indicates the message generator of agent  $i$ . It focuses on obtaining a robust policy via

adversarial training for different message perturbations.

To conduct the adversarial training, we aim to learn auxiliary message adversaries  $\hat{\pi}$  which perturb the messages received by each agent, transforming  $m$  into  $\hat{m}$ . Thus, each agent  $i$  actually takes action by  $\pi_i(a_i | \tau_i, \hat{m}_i)$ . Specifically, we apply additive perturbations and  $\hat{\pi}$  is defined as a deterministic policy, which means that:

$$\xi = \hat{\pi}(\mathbf{o}), \hat{m} = m + \xi,$$

where  $\mathbf{o}$  is the joint observation of all agents, i.e.,  $\mathbf{o} = (o_1, o_2, \dots, o_n)$ . In practice, we consider decentralized attack policy, where  $\hat{\pi} = (\hat{\pi}_1, \dots, \hat{\pi}_N)$  with each sub-policy taking the local observation  $o_i$  as input. More details are described in Section 4.1. Besides, arbitrary perturbations without bounds can have a devastating impact on the communication performance, and robustness under this situation is almost impossible. Hence to consider a more realistic setting, we restrict the power of adversaries and constrain the perturbed messages to a set  $\mathcal{B}$ . For example, we can typically define  $\mathcal{B}$  as a  $p$ -norm ball centered around the original messages, i.e.,  $\mathcal{B} = \{\hat{m} \mid \|m - \hat{m}\|_p = \epsilon\}$ , where  $\epsilon$  is the given perturbation magnitude and  $p$  is the norm type.

Finally, we optimize the ego-system policy by value-based MARL method, where deep Q-learning [63] implements the action-value function with a deep neural network  $Q(\tau, \mathbf{a}; \psi)$  parameterized by  $\psi$ . This paper follows the Centralized Training and Decentralized Execution (CTDE) [64] paradigm. In the centralized training phase, it uses a replay memory  $\mathcal{D}$  to store the transition tuple  $\langle \tau, \mathbf{a}, r, \tau' \rangle$ . We use  $Q(\tau, \mathbf{a}; \psi)$  to approximate  $Q(s, \mathbf{a})$  to alleviate the partial observability. Thus, the parameters  $\psi$  are learnt by minimizing the expected Temporal Difference (TD) error:

$$\mathcal{L}(\psi) = \mathbb{E}_{(\tau, \mathbf{a}, r, \tau') \in \mathcal{D}} \left[ (r + \gamma V(\tau'; \psi^-) - Q(\tau, \mathbf{a}; \psi))^2 \right],$$

where  $V(\tau'; \psi^-) = \max_{\mathbf{a}'} Q(\tau', \mathbf{a}'; \psi^-)$  is the expected future return of the TD target and  $\psi^-$  are parameters of the target network, which will be periodically updated with  $\psi$ . Note that though we follow the CTDE paradigm, multi-agent communication is allowed in the execution process, which means that the inputs to the individual  $Q$ -networks are agents' local observations and the received information. More specific details about the design of  $Q(\tau, \mathbf{a}; \psi)$  are related to the corresponding underlying multi-agent communication methods. Such as in the Full-Comm algorithm [5] where agents broadcast their individual observations,  $Q(\tau, \mathbf{a}; \psi)$  is decomposed of a mixing network and  $N$  individual  $Q$ -networks, where each  $Q_i$  is conditioned on both agent  $i$ 's observation and received messages.

### 4 Method

In this section, we will describe the detailed design of our proposed method named MA3C. Firstly, we show how to model message adversaries as a cooperative multi-agent attack problem and how to learn a specific attacker instance. Immediately after, we introduce the concept of attacker population and our diversity mechanism that helps obtain an auxiliary attacker population with diverse and qualified attacker instances. Finally, a complete training description of our approach is provided.



#### 4.1 Message channel level attacking

To achieve robust communication for a multi-agent system, we propose to train the ego system policy under possible communication channel attacks, thus consequently obtaining an attack-robust communication-based policy. Generally, we aim to design an adversarial framework that consists of two main-body components, which are respectively the **attacker** and **ego system** (c.f. Fig. 1). Specifically, the attacker aims to produce perturbation on the communication channels to degrade the communication performance of the ego system. In contrast, the ego system can be any MARL methods with communication, aiming at maximizing coordination ability. The core idea is that adversarial training helps the ego system encounter different message perturbation situations and learn how to handle possible communication attacks to achieve robust communication. To serve this goal, one critical question is how to build a qualified attacker.

To answer the mentioned question, we claim that a qualified attacker ought to have two vital properties - **comprehensiveness** and **specificity**. Mostly, every agent plays a different role in a multi-agent system, and each communication channel is of different importance to the whole communication system. In most cases, a portion of the message channels are more important, and perturbing these message channels is the key to achieving a practical attack. The referred **comprehensiveness** means that the attacker should consider all communication message channels to avoid only perturbing some specific or even useless messages, but overlooking those vital individuals, leading to low robustness. On the other hand, since different agents process the received messages in different ways in general, a reasonable perspective is that the attacker should be distinct in how it perturbs the messages received by each agent, which we call **specificity**.

Except for these two important properties, one more fundamental requirement for the attacker is that it should act sequentially to degrade the performance of the ego system. To serve this goal, we model the attacking process as a sequential decision problem. Specifically, the reward is defined as the opposite of the ego system's reward:  $\hat{R} = -R$ , and the learning objective is to maximize:

$$\mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t \hat{R} \right] = \mathbb{E} \left[ - \sum_{t=0}^{\infty} \gamma^t R(s^t, \hat{a}^t) \mid s^{t+1} \sim P(\cdot | s^t, \hat{a}^t), \right. \\ \left. \hat{a}_i^t \sim \pi_i(\cdot | o_i^t, \hat{m}_i^t), \hat{m}_i^t = m_i^t + \xi_i^t, \right. \\ \left. m_{i,j}^t = \text{msg}_i(o_i^t), \xi^t = \hat{\pi}(s^t) \right]. \quad (1)$$

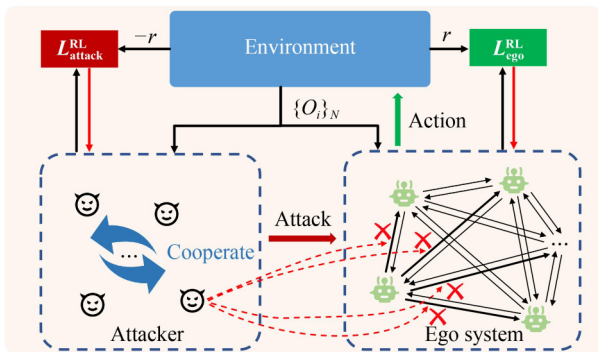
Note that to equip the attacker with the properties of **comprehensiveness** and **specificity**, we model the action space as a continuous action space with dimension of  $N \times (N-1) \times d_{\text{comm}}$ , where  $N$  denotes the number of agents,  $N \times (N-1)$  is the total number of the communication channels and  $d_{\text{comm}}$  indicates the dimension of one single message. At each time step, the *attacker* should learn to output an action with dimension of  $N \times (N-1) \times d_{\text{comm}}$  which serves as the perturbation on the communication messages:

$$\xi^t = \hat{\pi}(s^t), \xi^t \in \mathbb{R}^{N \times (N-1) \times d_{\text{comm}}}, \\ \hat{m}_{i,j}^t = m_{i,j}^t + \xi_{i,j}^t, i \neq j \in \{1, 2, \dots, N\}. \quad (2)$$

However, this design leads to a large action space, especially when there exist quite many agents in the environment since the action dimension grows squarely with the number of agents. The problem of large action space is likely to bring difficulties to the attack policy learning, which has already been discussed before [33,65]. In fact, some modern MARL algorithms like QMIX [66], decompose the joint action space and alleviate the difficulties posed by the large joint action space in policy learning. Motivated by this point, we propose to apply cooperative MARL methods to mitigate the problem of large action space in attack policy learning.

Specifically, we construct  $N$  virtual agents, divide the  $N \times (N-1)$  communication channels into  $N$  groups and let each virtual agent be responsible for the attacks on  $N-1$  communication channels. In other words, the action dimension for each virtual agent is  $(N-1) * d_{\text{comm}}$ , which grows linearly with the number of agents, and these  $N$  virtual agents together form a concept of **attacker**. As shown in Fig. 1, the  $i$ th virtual agent takes the individual observation of agent  $i$  as the input into its policy. The policy's output is the perturbation on the sent information from agent  $i$ , i.e., virtual agent  $i$  undertakes the responsibility of contaminating the messages sent by the  $i$ th agent. In this way, we can think that these  $N$  virtual agents cooperate to attack the underlying communication system effectively. The whole process of adversarial training can be seen as a confrontation between two groups of agents; one is the group of these  $N$  virtual agents, and the other is the underlying multi-agent system (the ego system). In particular, our approach can be applied to any off-the-self MARL algorithm that can handle problems with continuous action spaces.

In practice, adversary  $i$  (the virtual agent mentioned above) uses the current observation  $o_i$  of agent  $i$ , then learns an attack policy  $\hat{\pi}_i(\xi_i | o_i; \theta)$  to perturb the  $N-1$  messages sent by agent  $i$ , where  $\theta$  denotes the parameters of the total attacker model. Thus each message equals to  $m_{ij} + \xi_{ij}$ . We here can employ any actor-critic method to optimize the policy. However, in the MARL system, there exists only a shared reward; thus, the simplest way is to use a central critic similar to COMA [67] to



**Fig. 1** The overall relationship between the attacker and the ego system. The black solid arrows indicate the direction of data flow, the red solid ones indicate the direction of gradient flow and the red dotted ones mean the attack actions from the attacker onto specific communication channels

optimize it. It takes  $(s, \xi)$  as input and outputs the Q value  $Q(s, \xi)$ . Specifically, we extend TD3 [68] to multi-agent setting, named MATD3, where the centralized Q-function is learned with

$$\arg \min_{\theta} \sum_{j=1}^2 \sum_t (Q_j(s^t, \xi_1^t, \dots, \xi_N^t) - y^t)^2, \quad j \in \{1, 2\},$$

$$y^t = \mathbb{E} \left[ r^t + \gamma \min_{j=1,2} Q'_j(s^t, \hat{\pi}_1(o_i^{t+1}), \dots, \hat{\pi}_N(o_N^{t+1})) \right], \quad (3)$$

where we maintain two  $Q$ -networks  $Q_1, Q_2$ , and  $Q'_j$  is the target network for  $Q_j$ . The actors are optimized via the deterministic policy gradient:

$$\nabla_{\theta} J = \mathbb{E} \left[ \nabla_{\theta} \hat{\pi}_i(\xi_i | o_i) \nabla_{\xi_i} Q_1(s, \xi_1, \dots, \xi_N) \right]. \quad (4)$$

## 4.2 Attacker population optimization

Many previous adversarial algorithms for reinforcement learning typically build one attacker and alternatively do adversarial training via finding a Nash equilibrium solution. However, one often criticized issue is that the ego system is largely over-fitted to the corresponding attacker and may fail to obtain good performance in practice. One possible solution to this issue is constructing an attacker population instead of one single attacker. By forcing the ego system to perform well against all attackers in the population, we expect to impede the ego system from over-fitting to one specific pattern and help it achieve good performance in the average sense. We believe this practice can effectively help avoid some crash cases, which is consistent with the goal of robustness.

Besides, one natural idea is that the adversarial training population should be diverse enough to cover attackers with quite different patterns, thus avoiding the population degenerating into one single attacker. This requires that we train the ego system along with the learned attacker population as:

$$\arg \max_{\pi} J(\pi | \mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_{\hat{\pi} \in \mathcal{P}} J(\pi | \hat{\pi}),$$

$$J(\pi | \hat{\pi}) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R(s^t, \hat{a}^t) \middle| s^{t+1} \sim P(\cdot | s^t, \hat{a}^t), \right.$$

$$\hat{a}_i^t \sim \pi_i(\cdot | o_i^t, \hat{m}_i^t), \hat{m}_i^t = m_i^t + \xi_i^t,$$

$$\left. m_{:,i}^t = \text{msg}_i(o_i^t), \xi^t = \hat{\pi}(s^t) \right], \quad (5)$$

$$\text{Distance}(\hat{\pi}_i, \hat{\pi}_j) > \epsilon, \forall \hat{\pi}_i, \hat{\pi}_j \in \mathcal{P},$$

where  $\hat{\pi}_j \in \mathcal{P}$  means we sample an attacker from the learned population during the training process, and  $\text{Distance}(\hat{\pi}_i, \hat{\pi}_j)$  refers to the distance between attacker  $i$  and attacker  $j$ . Nevertheless, the remaining question is how to define the  $\text{Distance}$  function. Some previous population-based methods [51, 55] typically achieve a balance between the quality and diversity by utilizing the mean action vector as an identification for each instance. However, these practices usually require a well-defined behavioral descriptor [69], which is sometimes the prior knowledge about some critical states and has demonstrated even hard in MARL setting [70].

To reach the mentioned goal, firstly, we observe that the differences in attack patterns can be well revealed in the behaviors of the attacked ego system. The basic idea is that when under different types of communication attacks, the multi-agent system may exhibit different behaviors, e.g., generating quite different trajectories. Based on this idea, for each attacker  $j$ , we sample  $m$  trajectories of the ego system under the communication attacks of attacker  $j$ , and utilize a trajectory encoder to encode them into a representation  $z_j$ :

$$z_j = \frac{1}{m} \sum_{k=1}^m f_{\phi}^{\text{enc}}(\tau_j^k), \quad (6)$$

where  $f_{\phi}^{\text{enc}}$  denotes a trajectory encoder network which is parameterized by  $\phi$ . Then we utilize the representation  $z_j$  as the identification for attacker  $j$ , and the distance between  $z_i$  and  $z_j$  describes how different these two attackers are.

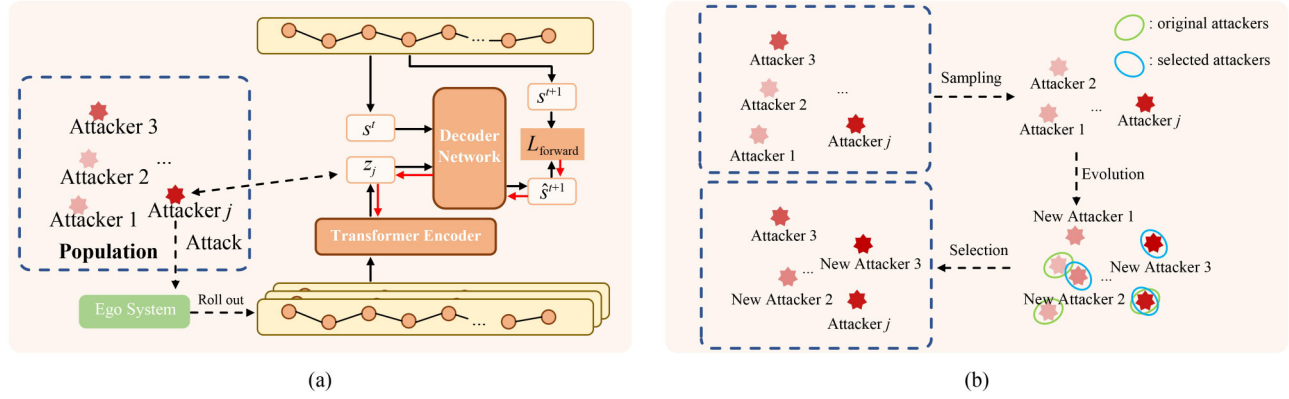
In designing the architecture and the optimization objective of the trajectory encoder network, we emphasize two vital points: (i) some specific parts in one trajectory are essential for distinguishing it from other trajectories; (ii) the representation  $z_j$  should imply the behavioral trends of the ego system when under the attacks of attacker  $j$ . To address point (i), we design the architecture of the trajectory encoder as a transformer [71] network because the attention mechanism can help the encoder focus on specific essential parts of the trajectory. In terms of point (ii), we design a forward prediction loss  $L_{\text{forward}}$  to help optimize the trajectory encoder network, as shown in Fig. 2(a). Specifically, we feed  $z_j$  and  $s_j^t$  into an extra decoder network, which outputs a prediction for  $s_j^{t+1}$ . We update the encoder and decoder networks together by minimizing the following prediction loss:

$$L_{\text{forward}}(\phi, \eta) = \sum_{k=1}^m \sum_{s_j^{t+1} \sim \tau_j^k} \|s_j^{t+1} - \hat{s}_j^{t+1}\|^2,$$

$$\hat{s}_j^{t+1} = f_{\eta}^{\text{dec}}(z_j, s_j^t), \quad t \in \{0, 1, 2, \dots, T-1\}, \quad (7)$$

where  $s_j^{t+1}$  denotes the state information at timestep  $t+1$  in one sampled trajectory, and  $f_{\eta}^{\text{dec}}$  indicates the decoder network parameterized by  $\eta$ . The key point is that by optimizing the forward prediction loss, we force the encoded representation  $z_j$  to contain behavioral information which can help predict the trend of these sampled trajectories.

After defining the identification for attackers, we optimize the attacker population by alternatively conducting the processes of evolution and selection. For evolution, we update the selected instances via the method described in Section 4.1, leading to some new instances. For selection, we always choose those more distant new instances to add to the population based on the definition of  $z_j$ . In specific, each time we want to update the population, we randomly select a fixed proportion of attacker models from the population. We apply MATD3 (c.f. Section 4.1) for each selected instance to update its attack policy with a fixed number of samples, resulting in a group of new attacker instances. Then for each new attacker, we find the closest instance to it in the population and compare their distance based on the trajectory representation.



**Fig. 2** The overall framework for the attacker population optimization. (a) We utilize the representation of the attacked ego system's trajectories to identify different attacker instances. Specifically, we apply an encoder-decoder architecture to learn the trajectory representation. The black solid arrows indicate the direction of data flow and the red solid ones imply the direction of gradient flow. (b) This is a simple visualization case for one time population updating. The locations of points imply the distances of representations and the color shades indicate the attack ability, i.e., the attackers corresponding to deeper points are stronger attackers. For example, new Attacker 3 is accepted as it is distant enough with other attackers, and the oldest Attacker 1 is removed; new Attacker 2 is accepted and the closest Attacker 2 is removed as it is weaker

If their distance is over a fixed threshold, we retain the new attacker and throw out the current *oldest instance* in the population; otherwise, we retain the better one by comparing their attack performance and throw out the other. Note that we utilize a First-In-First-Out (FIFO) queue to implement the attacker population and the *oldest instance* here means the first element of the queue. The whole process is shown in Algorithm 1.

### 4.3 Robust communication and training

Based on the proposed attacker training algorithm and the

---

#### Algorithm 1 Population update

---

**Input:** Population  $\mathcal{P}$ , Distance threshold  $\zeta$ , Number of evolution  $T_{\text{evolution}}$ , Ego system model  $\pi_{\text{ego}}$ , Trajectory encoder  $f_{\theta}^{\text{enc}}$ .

- 1: **for**  $t = 1, 2, \dots, T$  **do**
- 2:   Select a subset  $\mathcal{P}_{\text{sub}}$  of a fixed size from  $\mathcal{P}$  randomly.
- 3:   **for** attacker instance  $I_{\text{attacker}}$  in  $\mathcal{P}_{\text{sub}}$  **do**
- 4:     Update  $I_{\text{attacker}}$  with the MATD3 algorithm and obtain a new attacker instance  $I'_{\text{attacker}}$ .
- 5:     Roll out  $m$  trajectories  $\{\tau\}_m$  of the ego system  $\pi_{\text{ego}}$  under the attack of  $I'_{\text{attacker}}$ .
- 6:     Encode the trajectories  $\{\tau\}_m$  with  $f_{\theta}^{\text{enc}}$  and get identification  $z'$  for the new attacker instance  $I'_{\text{attacker}}$ .
- 7:     Select the attacker instance  $\bar{I}_{\text{attacker}}$  from  $\mathcal{P}$ , which has the closest identification  $\bar{z}$  to  $z'$ .
- 8:     **if**  $\|\bar{z} - z'\|_2 > \zeta$  **then**
- 9:       Remove the oldest attacker instance in  $\mathcal{P}$ .
- 10:      Add  $I'_{\text{attacker}}$  to  $\mathcal{P}$ .
- 11:     **else**
- 12:       **if**  $I'_{\text{attacker}}$  is stronger than  $\bar{I}_{\text{attacker}}$  **then**
- 13:          Remove  $\bar{I}_{\text{attacker}}$  from  $\mathcal{P}$ .
- 14:          Add  $I'_{\text{attacker}}$  to  $\mathcal{P}$ .
- 15:       **else**
- 16:          Discard  $I'_{\text{attacker}}$ .
- 17:       **end if**
- 18:     **end if**
- 19:   **end for**
- 20: **end for**

---

population optimization method, we further design a whole training framework where we alternatively train the ego system and update the population. In fact, throughout the whole process of adversarial training, we maintain an ego system and a fixed-size population. In the phase of ego-system training, we uniformly select an attacker from the population at the start of each episode. We let the ego system roll out with this attacker, and the roll-outed trajectory is added to a training buffer. We update the ego system with data sampled from the buffer, which equates to adversarially training the ego system against the whole attacker population. On the other hand, in the phase of population updating, we load the latest ego system model and apply the population updating mechanism against the loaded ego system, as described in Algorithm 1. The whole training process consists of multiple repetitions of these two phases, and the ego system and the population are iteratively enhanced in the whole process. The whole adversarial training procedure is described in Algorithm 2.

## 5 Experimental results

Note that our approach is orthogonal to the underlying multi-agent communication algorithm, thus to validate the effectiveness of our approach in this section, we apply our approach to different communication methods and conduct experiments on various benchmarks. In specific, we aim to answer the following questions based on the experimental results in this section: 1) Can MA3C facilitate the robustness of multi-agent communication, and does each part in our approach make effect (Section 5.2)? 2) What kind of diverse

---

#### Algorithm 2 Adversarial training

---

**Input:** Initialized population  $\mathcal{P}$ , Initialized ego system policy  $\pi_{\text{ego}}$ .

- 1: Pre-train the ego system without any adversaries.
- 2: **for** iter = 1, 2, ..., max\_iter **do**
- 3:   Call Population Update to update the attacker population  $\mathcal{P}$ .
- 4:   Update the ego system policy  $\pi_{\text{ego}}$  with a multi-agent communication method against the whole attacker population  $\mathcal{P}$ .
- 5: **end for**

---



attacker population has been obtained by MA3C (Section 5.3)? 3) How does MA3C perform in the face of communication attacks with unseen perturbation ranges (Section 5.4)? Besides, we offer descriptions about the benchmarks in Section 5.1 and do parameter sensitivity studies in Section 5.5.

Specifically, in our experiments, we apply our approach to three different communication algorithms: Full-Comm [5], NDQ [20], and TarMAC [21], which are of different features. Full-Comm is a popular communication paradigm, where each agent directly broadcasts their individual observations and updates the communication networks with an end-to-end scheme to minimize the Q-value estimation loss, showing competitive communication ability in multiple scenarios [4,5]. NDQ aims to generate meaningful messages and does message minimization to achieve nearly decomposable Q-value functions. TarMAC applies an attention mechanism in the receiving end to help agents focus on the specific part of the received messages. The details about NDQ and TarMAC are shown in Appendixes A.1.

### 5.1 Environments

In general, we select four multi-agent environments (see Fig. 3), respectively Hallway [20], StarCraft Multi-Agent Challenge (SMAC) [61], a task environment we designed for multi-agent communication named Gold Panner (GP), and Traffic Junction (TJ) [21].

#### • Hallway

Hallway is a multi-agent task with partial observability, where multiple agents are randomly spawned at different locations and required to reach the target simultaneously. In experiments, we design two instances of the Hallway task, where the first instance (Hallway-6×6) has two hallways with a length of 6, and the second instance (Hallway-4×5×9) has three hallways that have lengths of 4, 5 and 9, respectively.

#### • StarCraft Multi-Agent Challenge (SMAC)

StarCraft Multi-Agent Challenge (SMAC) is a popular benchmark for multi-agent cooperation, where there are agent units from two camps, and the goal of the multi-agent algorithm is to control one of the camps to defeat the other. In particular, we select two maps named 1o\_2r\_vs\_4r and 1o\_10b\_vs\_1r from SMAC. In 1o\_2r\_vs\_4r, an Overseer is spawned around four enemy Reapers, and two ally Roaches are expected to reach the enemies and defeat them. Alike, in

1o\_10b\_vs\_1r, one Overseer detects a Roach, and 10 Banelings are required to reach and kill the enemies.

#### • Gold Panner (GP)

To further validate our approach’s effectiveness, we also design a task named Gold Panner (GP), which is a grid world with partial observability. This task divides the whole map into several parts, and the agents are randomly spawned at different regions. There exist one grid containing gold that is initialized within sight of one agent, and agents are expected to load the gold at the same time. The core idea is that the agent nearby the gold ought to communicate the messages about the gold’s location to the other agents to help all agents gather near the gold and load the gold together. We design two instances, which are respectively GP-4r and GP-9r. In GP-4r, there are 4 (2×2) regions, of which each region is a field with size of [3, 3], and three agents existing in the map, while, in GP-9r, there exist 9 (3×3) regions and 3 agents.

#### • Traffic Junction (TJ)

Traffic Junction (TJ) is a familiar environment for testing the communication performance of multi-agent systems. In the task of Traffic Junction, multiple vehicles move on the two-way roads with several junctions and consistently follow a fixed route. We test on the medium version map of Traffic Junction, where the road dimension is 14, and there exist two junctions on each road when applying our approach to the communication method TarMAC.

### 5.2 Robustness comparison

To testify whether our approach can facilitate robust communication when applied to different communication algorithms and scenarios, we apply our approach to three communication methods and select four tasks requiring agent communication. Specifically, we employ Full-Comm in Hallway, SMAC, and GP tasks, and NDQ and TarMAC in SMAC and TJ, respectively. The test results are listed in Table 1, and more details about the experiments are provided in Appendixes A.3.

For the compared algorithms, we design three straightforward baselines, including Vanilla, Noise Adv. and Instance Adv. respectively. Vanilla does not apply any adversarial training technique and learns the ego system policy in scenarios without communication attacks. Noise Adv. applies adversarial training with random noise attacks. While Instance Adv. builds one single communication attacker,

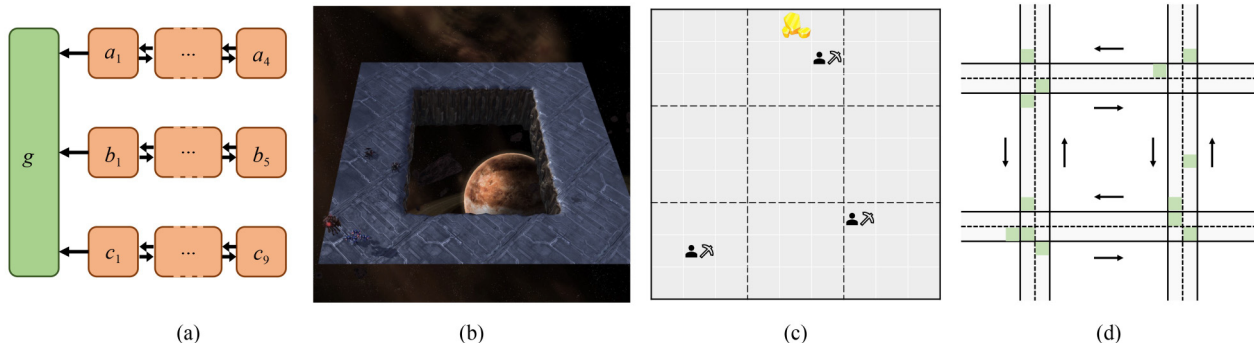


Fig. 3 Experimental Environments used in this paper. (a) Hallway; (b) SMAC; (c) Gold Panner (GP); (d) Traffic Junction (TJ)

**Table 1** Performance comparison under different attack modes

		Hallway-6x6	Hallway-4x5x9	SMAC-1o_2r_vs_4r	SMAC-1o_10b_vs_1r	GP-4r	GP-9r
Normal	MA3C	0.94±0.05	0.97±0.05	0.86±0.02	0.62±0.01	0.87±0.02	0.82±0.01
	Vanilla	<b>1.00±0.00</b>	<b>1.00±0.00</b>	0.81±0.06	<b>0.63±0.04</b>	<b>0.88±0.03</b>	0.82±0.02
	Noise Adv.	<b>1.00±0.00</b>	0.99±0.01	<b>0.88±0.04</b>	0.6±0.05	<b>0.88±0.03</b>	<b>0.85±0.02</b>
	MA3C w/o div.	0.98±0.02	0.66±0.46	0.86±0.02	0.62±0.03	0.86±0.09	0.81±0.03
	Instance Adv.	0.52±0.48	0.67±0.47	0.84±0.02	0.57±0.04	0.86±0.03	0.82±0.03
	AME	<b>1.00±0.00</b>	0.98±0.02	0.81±0.05	0.60±0.01	0.23±0.37	0.00±0.00
Random noise	MA3C	0.91±0.07	0.79±0.18	<b>0.87±0.01</b>	<b>0.67±0.03</b>	0.88±0.01	0.80±0.07
	Vanilla	0.58±0.03	0.53±0.06	0.73±0.07	0.60±0.02	0.86±0.03	0.79±0.02
	Noise Adv.	<b>0.97±0.02</b>	<b>1.00±0.00</b>	0.82±0.02	0.56±0.02	0.88±0.01	<b>0.82±0.01</b>
	MA3C w/o div.	0.68±0.07	0.68±0.29	0.73±0.07	0.53±0.01	0.82±0.06	0.80±0.07
	Instance Adv.	0.56±0.34	0.67±0.47	0.79±0.07	0.60±0.08	<b>0.90±0.03</b>	0.81±0.02
	AME	0.61±0.06	0.79±0.03	0.71±0.13	0.59±0.08	0.22±0.37	0.00±0.00
Aggressive attackers	MA3C	<b>0.91±0.22</b>	<b>0.98±0.01</b>	<b>0.67±0.03</b>	<b>0.62±0.03</b>	<b>0.81±0.02</b>	<b>0.76±0.03</b>
	Vanilla	0.09±0.19	0.00±0.00	0.26±0.12	0.57±0.03	0.38±0.02	0.30±0.05
	Noise Adv.	0.61±0.37	0.13±0.14	0.51±0.02	0.54±0.03	0.41±0.13	0.48±0.11
	MA3C w/o div.	0.57±0.39	0.96±0.03	0.54±0.05	0.61±0.02	0.68±0.06	0.71±0.01
	Instance Adv.	0.63±0.42	0.88±0.14	0.28±0.01	0.61±0.04	<b>0.81±0.02</b>	<b>0.76±0.03</b>
	AME	0.13±0.03	0.00±0.00	0.39±0.05	0.59±0.07	0.10±0.16	0.00±0.00

training the communication system and the attacker alternatively. Actually, Instance Adv. can be seen as an ablation that does not use population, thus used to verify the influence of attacker population. One special note is that to alleviate the overfitting problem of adversarial training with one single attacker and thus construct a stronger baseline, we enhance Instance Adv. by maintaining a pool of historical attacker models and doing adversarial training against the whole pool. Besides, to make our work more solid, we additionally compare our approach with two existing methods for robustness. The first one is the variant of RAP in our experiment setting, which also adopts the adversary population for training robust policies. Its main difference from our approach is that it does not explicitly optimize the diversity of the population. Thus, we denote this baseline as MA3C w/o div., which can be used to verify the effectiveness of our diversity mechanism. The other compared method is a recently proposed work called AME [31], which builds a message-ensemble policy by aggregating the decision results of utilizing different ablated message subsets.

For each method, we employ three different test modes: 1) **Normal** means no communication attacks, which is to show the communication performance of the multi-agent system in clean scenarios; 2) **Random noise** tests the communication robustness under random noise attacks; 3) **Aggressive Attackers** additionally trains a set of unseen communication attackers and utilizes them to do robustness test.

As we can see from Table 1, our approach MA3C exhibits comparable or more better performance than other baselines when applied to Full-Comm. Concretely, the Vanilla baseline, trained in a natural manner, performs excellently in the Normal setting where no noise exists but suffers from performance drop when tested in a noisy communication environment. It even fails when encountering aggressive attackers (e.g., it obtains zero success rate on Hallway-4×5×6). This phenomenon reveals the vulnerability of communication-based policy and calls for algorithm design to enhance communication robustness. As for Noise Adv., it works well when tested with random noise attacks since this corresponds

to its training situation. However, it struggles when encountered with unseen aggressive attackers, e.g., a performance drop of 0.87 is found in the task of Hallway-4x5x9. We hypothesize that the reason for this is that the randomly generated noise attacks can hardly cover some specific attack patterns, and the obtained policy fails in the face of such communication attacks. Furthermore, for the Instance Adv., though we strengthen it by maintaining a pool of historical attacker models, it still suffers from coordination degradation in the Normal situation in different environments, which shows that adversarial training may damage the communication ability, and has been discussed in some other RL domains [19]. The performance advantage of MA3C and MA3C w/o div. over Instance Adv. demonstrates the effectiveness of population.

Besides, the comparison to the baseline MA3C w/o div., which conducts a similar approach to RAP in our setting, proves that our diversity mechanism has good gains in the robustness of obtained ego system policy. The essential reason behind it is that it can avoid homogeneity of instances in the population and ensure that the communication attackers encountered during the training phase can cover more attack patterns. Again, when compared to the baseline AME, our approach still exhibits good performance advantages. By aggregating the decision results of utilizing multiple ablated message subsets, AME can somewhat alleviate the influence of communication attacks and shows relatively good performance when tested with random noise on the environments of Hallway and SMAC. However, when we apply aggressive attackers to test its robustness, its performance faces a significant drop, which shows the limitation of this kind of approach. Besides, we find that AME performs terribly on the environment of GP. We hypothesize the reason is that some specific channels are vital for the agents to complete this task, so the practice of utilizing ablated message subsets may miss these critical messages. Considering that the AME approach is more suited to the setting where a portion of agents are attacked, we additionally test MA3C and AME in this setting, and the results are



reported in Appendixes A.4. It can be seen that, even in this test setup, MA3C still shows better communication robustness, justifying the superiority of our approach.

### 5.3 Attacking behavior analysis

To reveal what kinds of attackers have been obtained by our approach, we conduct visualization analysis to check whether our attacker population optimization method can help obtain a population with diverse and qualified attackers in the task of SMAC-1o\_2r\_vs\_4r. Specifically, we pre-train a multi-agent communication system and apply our population mechanism to train an attacker population to conquer the communication system. Each attacker instance in the population has an identification vector, as mentioned before. We take out attacker instances from the population at various stages throughout the training process, downscale their feature vectors to one dimension, and visualize them in Fig. 4. We expect the attackers to cover more regions along the horizontal coordinate, which means diversity, and to be located as high as possible for the vertical coordinate, indicating good attack ability.

From the results shown in Fig. 4(a), we can see that scatters on the top tend to be darker, implying that the population optimization process can help obtain stronger attackers. We also compute the whole attack performance, which equals to the average attack performance of all attackers in the population, and plot the variant curve in Fig. 4(a), from which we can see the upward trend of the population’s attack ability. Besides, we mark the final population in the last iteration as yellow stars, and we can see that the final 20 attackers are diverse along the attacker identification feature axis. To further check what attack patterns have been learned for the attackers in the population, we render the trajectories for specific attacker instances and find: (1) in the picture at the bottom left, one Roach wanders around, leaving its teammate alone to battle with the enemies; (2) in the picture at the top right, the messages of the Overseer are attacked, and fake enemy location information is transmitted to the two teammates, leading to the two ally Roaches moving towards

the fake locations; (3) in the picture at the bottom right, attacker achieves effective attacks by tricking the allies into remaining silent during battling through message perturbations.

Furthermore, since our approach and other baselines are all agnostic to specific communication methods, we also implement them on other typical communication methods such as NDQ and TarMAC. We can see from Fig. 5, the superiority over other baselines demonstrates the generality of MA3C.

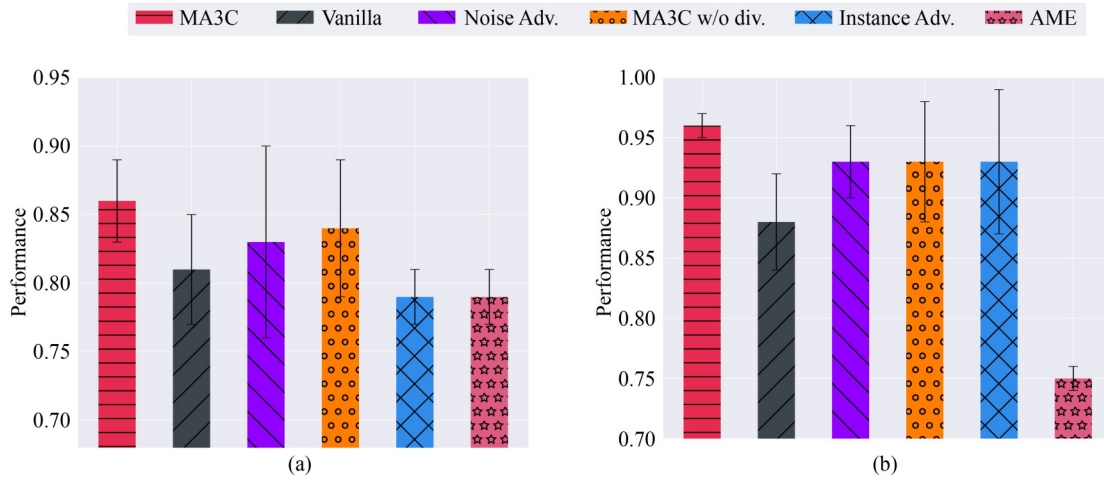
After that, we conduct experiment to investigate whether our approach can help obtain effective attackers. The concern is that the adversarial training will be meaningless if the obtained attackers can not generate practical communication attacks. Thus to confirm the validity of the attacker learning, we independently train multiple attackers with our approach in tasks of SMAC-1o\_2r\_vs\_4r and GP-4r. In both tasks, we restrict the learning process to be within 500K samples, and the vertical coordinate indicates the test performance of the attacked ego system. To further justify our practice of learning the attacker via the MARL method, we additionally compare with a baseline that learns the attacker with the TD3 [68] algorithm, which means treating it as a single-agent learning problem. A baseline named Random Noise is also added to show the performance under random noise attacks. From the learning curves in Fig. 6, we can see that the TD3 baseline and our approach can converge to much lower performance compared with Random Noise, which means that they have found the vulnerability of the communication system and learned effective attack patterns. Also, our approach exhibits better attack performance than the TD3 baseline, which shows the effectiveness of modeling the attacker learning as a multi-agent learning problem, and the virtual agents actually learn to cooperate to attack the communication system, obtaining better attack performance.

### 5.4 Policy transfer

In fact, we suppose that the allowed perturbations are always in a restricted set  $B(m)$ , and all the experiments above assume



**Fig. 4** (a) This curve traces the average attacker performance of the attacker population as evolution rounds increase; (b) population visualization. In specific, each scatter corresponds to an attacker instance, and we use the color depth to represent the training stage of the attackers, i.e., the lighter the color, the earlier the attacker model. The horizontal coordinate indicates the identification feature after dimension reduction, and the vertical coordinate indicates the attack performance of the attacker model

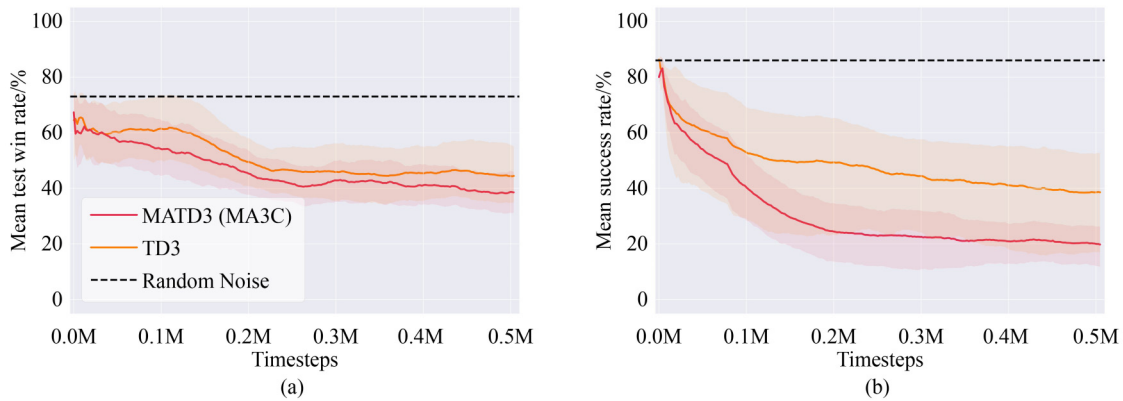


**Fig. 5** Robustness comparison when employing NDQ on SMAC-1o\_2r\_vs\_4r and TarMAC on TJ, respectively. (a) NDQ:SMAC-1o2rvs4r; (b) TarMAC:TJ

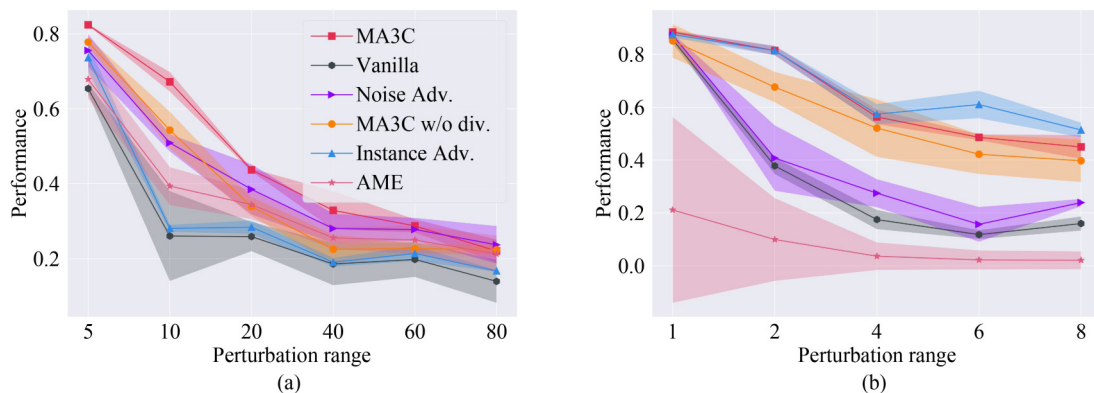
that the test perturbation range is the same as that utilized in the adversarial training, of which the details are introduced in Appendixes A.3. However, in many practical scenarios, we can not suppose the real communication attacks encountered in the execution phase are always within the perturbation range designed in the training phase. Thus, we wonder how our approach performs when generalizing to attacks with different perturbation ranges.

Firstly, we collect multiple aggressive attackedrds using the same method in Section 5.2, but with different perturbation

ranges when testing, which can be seen as a direct zero-shot generalization test. We conduct the experiment in tasks of SMAC-1o\_2r\_vs\_4r and GP-4r, and the results are recorded as plots in Fig. 7. From the results, we can see that as the perturbation range increases, all algorithms face a consistent performance degradation trend. This phenomenon is expectable because when the perturbation range is larger, greater changes in the input values tend to have a greater impact on the network's output, thus causing larger harm to the communication performance. Besides, we find our



**Fig. 6** Comparison of the attack ability of different methods. (a) SMAC-1o 2r vs 4r; (b) GP-4r



**Fig. 7** Generalization test to different perturbation ranges. (a) SMAC-1o 2r vs 4r; (b) GP-4r

approach MA3C exhibits good generalization performance under different unseen perturbation ranges. For example, in the task of SMAC-1o\_2r\_vs\_4r, MA3C still obtains the highest win rate compared to baselines and ablations under different perturbation ranges. This demonstrates the generalization ability of our approach, which is of significance for the deployment in some unknown scenarios.

Furthermore, we also conduct experiments to look into whether our approach MA3C can help achieve a quick adaptation to new environment. We here conduct transfer experiments to quite large perturbation ranges to test this property. Specifically, we select perturbation ranges of 5 and 15 in the tasks of Hallway-4×5×9 and GP-4r, respectively, compared to choices of 1.0 and 2 during adversarial training. To validate the effectiveness of the our approach, we compare with the baseline which learns from scratch in the noisy scenarios. Besides, to certify that the transfer gain is not from the extra pre-training, we add a baseline that only pre-trains the ego system policy in a clean scenario. The results in Fig. 8 demonstrate that MA3C is equipped with a good property for transferring to larger perturbation ranges, and we claim that it is of great value for some real-world applications.

### 5.5 Parameter sensitivity studies

To investigate how the parameters introduced in our work influence the final robust performance of our approach, we selectively conduct parameter sensitivity studies in the task of GP-4r. Specifically, we choose three core hyper-parameters: (1) Population size: the size of the population; (2) Reproduction Ratio: the ratio of attackers each time we select to do evolution; (3) Distance Threshold: a hyper-parameter to determine whether a new attacker instance is novel enough for the current population. We here report the results under Aggressive Attackers and Random Noise. Note that the default hyper-parameter selection in the experiments above is listed in Appendixes A.2, where the default value for Population, Reproduction Ratio and Distance Threshold are respectively 20, 0.25, and 0.25.

As we can see from the results in Fig. 9, overall the performance of the communication system under Aggressive Attackers is much lower than that under Random Noise, and the performance under Random Noise is not sensitive to the hyper-parameters, which confirms that Aggressive Attackers

have higher attack ability than Random Noise. Among these three hyper-parameters, we find that the Population Size has the least impact on the algorithm performance, implying that a population with size of five attacker instances works well in this task. For Reproduction Ratio, an interesting phenomenon is that the approach works poorly when Reproduction Ratio is zero. This phenomenon is because when Reproduction Ratio is zero, the population is always a set of random initialized attacker instances, which can provide limited information for adversarial training. Besides, for Distance Threshold, we find a selection of value 0.25 works best in this task. A too small Distance Threshold like zero ignores the process of diverse selection, and this makes the algorithm degenerate to the baseline of MA3C w/o div., while a too-large Distance Threshold causes our algorithm to pay less attention to the attack ability of the attacker instances.

### 5.6 Trajectory encoder studies

Finally, in our approach, we adopt the transformer architecture as the trajectory encoder that helps distinguish the attack patterns of different attacker instances. The superiority of the transformer architecture is that its attention mechanism can help capture the critical points in the trajectory, and its network expressiveness can help handle complex scenarios. To further demonstrate the advantages of the transformer architecture, we additionally consider a variant of MA3C that uses Multi-Layer Perception (MLP) and Mean-Pooling technology to obtain the trajectory representation as shown in Fig. 10. In this variant, the transformer architecture of the trajectory encoder is removed. We use this experiment to study how much influence the transformer architecture has.

We apply this variant to the GP environment, of which the results are shown in Table 2. Note that this variant differs from MA3C only in the trajectory encoder network and all other training details are the same. In fact, we can see from the results that MA3C with ordinary trajectory encoder achieves comparable performance to MA3C when tested with random noise attacks. However, when tested with aggressive attackers, this variant suffers a greater drop in performance compared to MA3C, but still shows better robustness than MA3C w/o div. slightly. For example, when tested with aggressive attackers, MA3C, MA3C w/ordinary encoder, and MA3C w/o div.

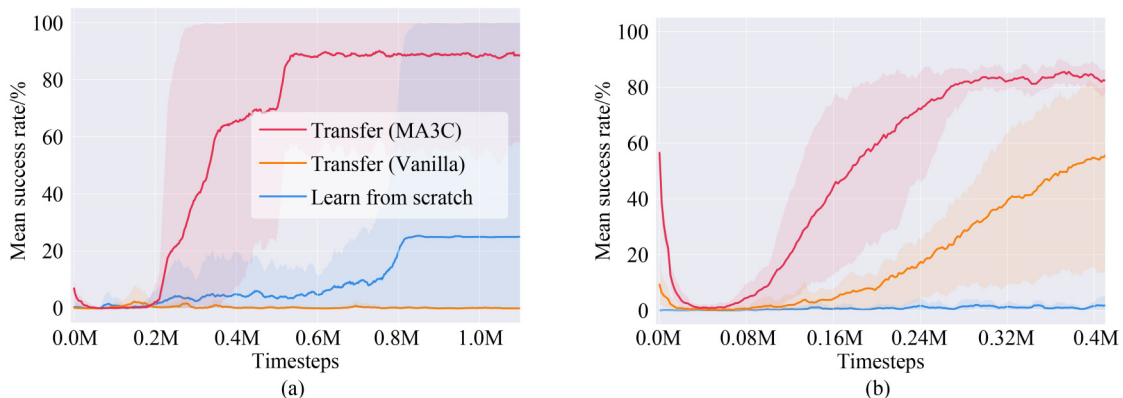
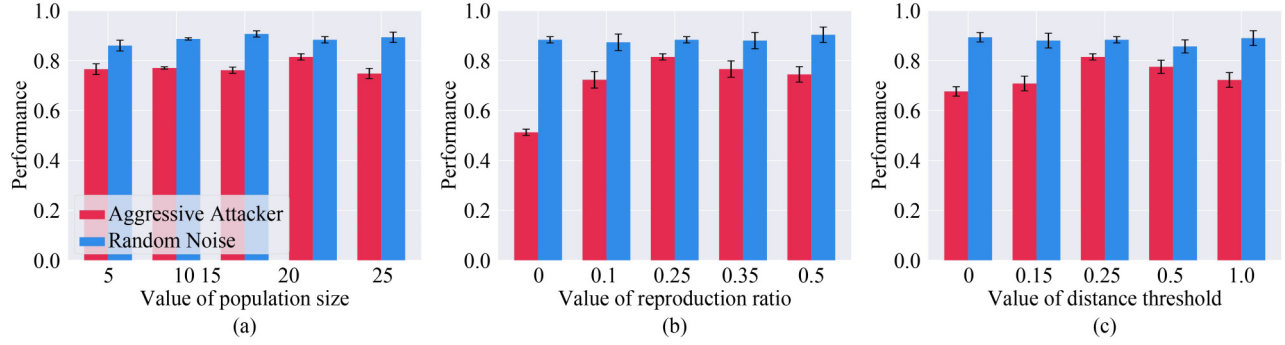
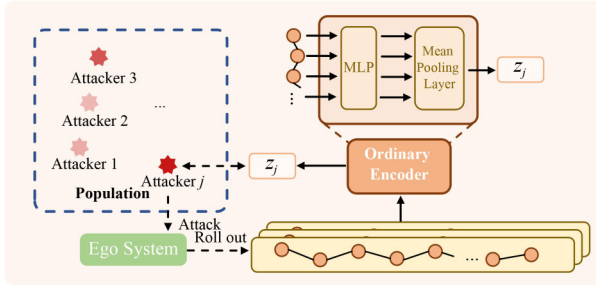


Fig. 8 Transfer to larger perturbation range. (a) Hallway-4×5×9; (b) GP-4r





**Fig. 9** Test results of parameter sensitivity studies. (a) Studies of population size; (b) studies of reproduction ratio; (c) studies of distance threshold



**Fig. 10** The architecture of the ordinary trajectory encoder. We feed the state information at each time step into a shared MLP network, and then perform mean pooling on the outputs to obtain the embedding vector of the trajectory.

**Table 2** Test results for the trajectory encoder studies

		MA3C	MA3C w/ordinary encoder	MA3C w/o div.
GP-4r	Normal	0.87±0.02	0.90±0.02	0.86±0.09
	Random Noise	0.88±0.01	0.90±0.03	0.82±0.06
	Aggressive Attackers	0.81±0.02	0.73±0.04	0.68±0.06
GP-9r	Normal	0.82±0.01	0.82±0.03	0.81±0.03
	Random Noise	0.80±0.07	0.80±0.05	0.80±0.07
	Aggressive Attackers	0.76±0.03	0.70±0.06	0.71±0.01

suffer performance drops<sup>2)</sup> of 7%, 19%, and 21%, respectively. We hypothesize that this is because the ordinary encoder fails to capture the differences between certain trajectories, thus interfering with MA3C’s diversity mechanism. This results in the variant failing to cover some specific attack patterns and causing a greater performance drop.

## 6 Final remarks

How to obtain a robust communication-based policy recently became an emergent for policy deployment. Instead of employing existing techniques to get a robust policy under some constraints, we take a further step for this issue by learning adaptable multi-agent auxiliary adversaries to promote robustness for communication-based policy. Sufficient experiments conducted on multiple cooperative

benchmarks demonstrate the high robustness ability of our proposed method, other results also show its high generalization ability for various perturbation ranges, and the learned policy can transfer learned robustness ability to new tasks after fine-tuning with a few samples. As we consider a limited perturbation set, how we can develop an autonomous paradigm like curriculum learning to find the communication ability boundary is an invaluable direction, and developing efficient and effective MARL communication methods under the open-environment scenarios [72] is challenging but of great value in the future. Also, as discussed in Appendixes A.5, the inclusion of auxiliary adversaries also brings more training overhead. In the future, we hope to further optimize our approach to improve its training efficiency.

**Acknowledgements** This work was supported by the National Key R&D Program of China (2020AAA0107200), the National Natural Science Foundation of China (Grant Nos. 61921006, 61876119, 62276126), the Natural Science Foundation of Jiangsu (BK20221442), and the Program B for Outstanding PhD Candidate of Nanjing University. We thank Ziqian Zhang and Lihe Li for their useful suggestions.

**Competing interests** The authors declare that they have no competing interests or financial conflicts to disclose.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made.

The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## Appendixes

### A.1 Introduction to the selected baselines

**NDQ** [20] considers that many multi-agent tasks in the real world are not fully decomposable and then achieve nearly decomposable Q-functions via communication minimization.

<sup>2)</sup> The performance drop rate here is calculated as the percentage of drop in performance when moving from the Normal mode to being attacked by Aggressive Attackers, i.e.,  $\frac{p(\text{Normal}) - p(\text{Aggressive Attackers})}{p(\text{Normal})}$ , where  $p(X)$  is the performance in the test mode of  $X$ .

Specifically, it trains the communication-based policy by maximizing the mutual information between the agents’ action selection and the message sent to the corresponding teammate. It also minimizes the entropy of messages between agents to avoid distribution collapse. Each agent broadcasts messages to all other agents and uses the received message to augment the local policy. As NDQ minimizes the message entropy, it can extract the most useful part for decision-making and shows great performance on many tasks like SMAC.

**TarMAC** [21] is a widely used MARL communication approach, which applies an attention mechanism to extract the most valuable information from multiple received messages. Concretely, each agent generates signature and query vectors as the message, and the message is then broadcasted to all teammates. In the message-receiving phase, the attention weights of incoming messages are obtained by calculating the similarity between the query vector and the signature vector of each incoming message. Then a weighted sum over all incoming messages is performed to determine the message an agent. The extracted message is finally used to augment the local observation for decentralized execution.

### A.2 Implementation details

Our implementation of MA3C is based on PyMAREL<sup>3</sup> with StarCraft 2.4.6.2.69232. We adopt its default settings for some common hyper-parameters like learning rate. The choices of other hyper-parameters in our experiments are listed in Table A1.

### A.3 Experimental details

In this section, we provide more experimental details to help

the reader better understand or reproduce our experimental results. Specifically, we discuss the details about robustness comparison (Section 5.2), attacking behavior analysis (Section 5.3), policy transfer (Section 5.4), and parameter sensitivity studies (Section 5.6), respectively. Note that all experiments are conducted with five independent runs and we report the mean and standard deviation.

#### • Robustness comparison

As we have mentioned in the section of Problem Setting, if perturbation without bounds is allowed, the attacker can be arbitrarily strong and effective defence may be impossible. Thus, to consider a more realistic setting, we specify that the perturbed messages are restricted in a specific set, e.g., the perturbed messages  $\hat{m}$  ought to satisfy  $\|m - \hat{m}\|_p \leq \epsilon$  with respect to the original messages  $m$ . Actually, we adopted a practice that the perturbed messages are constrained to a 1-norm ball centered on the original communication messages, which means the set  $\mathcal{B} = \{\hat{m} \mid \|m - \hat{m}\|_1 = \epsilon\}$  for most experiments, except for the experiments on Hallway. The observation dimension on Hallway equals to 1; thus if we apply the previous practice, the action space will degenerate to a discrete action space of  $\{\epsilon, -\epsilon\}$  when attacking the communication system learned with Full-Comm. For experiments on Hallway, we constrain  $\hat{m}$  to satisfy  $\|m - \hat{m}\|_1 \leq \epsilon$ . Besides, the adopted magnitude  $\epsilon$  for each experiment is listed in Table A2.

Besides, for the test mode of **Aggressive Attackers**, we additionally train a set of unseen communication attackers and utilize them to test the robustness of different methods. Specifically, we extraly train a diverse attacker population

**Table A1** Selected hyper-parameters in our experiments

Hyper-parameter name	Other Experiments	Hallway-4x5x9	TarMAC: TJ	GP-4r, GP-9r
Population Size (The number of attackers one population contains)			20	
Reproduction Ratio (The proportion of updated agents in the population during each evolution)			0.25	
Distance Threshold (The threshold to determine whether the new attacker is novel enough)			0.25	
Critic Update Times (The number of times the critic is updated for each actor update in MATD3)			5	
Num of Sampled Trajectories (The number of sampled trajectories used to encode attacker identification)			10	
Alternate Update Times (The number of iterations of alternate updates between the ego system and the attacker population)	15	30	20	6
Num of Samples for Ego System in One Loop (The number of samples used to update the ego system in each iteration)		205000		505000
Evolution Times in One Loop (The number of times the population conduct evolution in each iteration)			10	
Num of Samples for Attacker in One Evolution (The number of samples used to update the attacker in each evolution)			10000	

**Table A2** Adopted magnitude  $\epsilon$  for each experiment. Comm. Alg. is short for Communication Algorithm

Comm. Alg.	Full-Comm			
Task	Hallway-6x6	Hallway-4x5x9	SMAC-1o_2r_vs_4r	SMAC-1o_10b_vs_1r
$\epsilon$	1.5	1.0	10	25
Comm. Alg.	Full-Comm		NDQ	TarMAC
Task	GP-4r	GP-9r	SMAC-1o_2r_vs_4r	TJ
$\epsilon$	2	2	6	16

<sup>3</sup> We use PyMAREL with SC2.4.6.2.6923 for the experiments on SMAC-1o\_2r\_vs\_4r and SMAC-1o\_10b\_vs\_1b. Note that performance is not always comparable among versions

with the technique of MA3C. Then we evenly split the training process of the population into five stages, and randomly sample four attacker models from the stored models at each stage, finally resulting in a total of 20 attacker models. We adopt the average performance of the ego system policy under the attacks of these 20 attacker models to represent the robustness performance.

- Attacking behavior analysis

In the experiments for Attacking Behavior Analysis, we studied the attack ability of our approach and the obtained attacker population. Specifically, for both the experiments of attack ability comparison and population visualization, we firstly pre-train a communication-based policy for the ego system, then we apply MATD3 and TD3 to learn attacker models to this ego system for the experiments of attack ability comparison, and apply MA3C to optimize an attacker population for population visualization. Besides, the adopted magnitude  $\epsilon$  is consistent with that in the section of Robustness Comparison, 10 for SMAC-1o\_2r\_vs\_4r and 2 for GP-4r.

- Policy transfer

In this part, we investigated the generalization ability of our approach to different perturbation ranges. Actually, we utilize the same setting as before, which means that we constrain the perturbed messages to be on a 1-norm ball centered on the original messages on task of SMAC and GP. However, we modify the magnitude  $\epsilon$  to test the generalization ability and transfer ability of our approach. For the zero-shot generalization test, we select a magnitude set of {5, 10, 20, 40, 60, 80} for SMAC-1o\_2r\_vs\_4r, where 10 is the magnitude for adversarial training, and select a magnitude set of {1, 2, 4, 6, 8} for GP-4r, where 2 is the magnitude for adversarial training. For the fine-tuning transfer test, the magnitudes for training are 1 and 2 respectively for Hallway-4x5x9 and GP-4r, while the transfer perturbation ranges are respectively 5 and 15.

- Parameter sensitivity studies

In the section of Parameter Sensitivity Studies, we selectively experiment on the task of GP-4r to investigate the parameter sensitivity of our approach. In specific, we consider three main hyper-parameters, which are Population Size, Reproduction Ratio, and Distance Threshold, respectively. When studying the influence of one specific hyper-parameters, the other hyper-parameters are set as the default values utilized in the main experiments. For example, when we investigate the hyper-parameters Population Size, the Reproduction Ratio and

Distance Threshold are both set as 0.25. The other settings like perturbation range are the same as those in the experiments of Robustness Comparison.

#### A.4 Additional experiments for the AME baseline

Considering that the AME approach is more suited to the test setting where a part of communication channels are attacked, we further conduct the test in this setup to justify the effectiveness of our approach. Specifically, we conduct experiments in the task of SMAC-1o\_2r\_vs\_4r and GP-4r, and the results are reported in Table A3. From the results, we can see that the AME approach shows better robustness when only partial communication channels are attacked compared with the results in Table 1. However, our approach MA3C still exhibits good performance advantages over the AME baseline, which validates the effectiveness of our approach.

#### A.5 Discussion about training efficiency of our approach

In our proposed method, we maintain an additional attacker population and conduct centralized adversarial training to enhance the communication policy's resilience to communication attacks and maintain robust communication performance. This naturally results in extra training overhead. To gain a more intuitive and in-depth understanding of our algorithm's training efficiency, we present an analysis experiment in this section, comparing our approach's time and memory costs with those of other algorithms in two experimental settings: GP-4r and GP-9r. To provide a more comprehensive comparison that covers both space and time consumption, we take into consideration the runtime, memory usage and graphics memory usage of different algorithms, of which the results are recorded in Table A4. In specific, the metric "Time" measures the actual time spent by different algorithms for one whole training process, where we keep the number of training samples the same for each algorithm. The other two metrics, "Memory" and "Graphics Memory", represent the average CPU memory usage and GPU memory usage during the algorithm process, respectively. For the computational resources used in this analysis experiment, we utilized a server equipped with an 80-core Intel(R) Xeon(R) Gold 5218R CPU @ 2.10 GHz and four NVIDIA GeForce RTX 3090 GPUs. In order to make a fair comparison between different algorithms, we controlled the experiments of each algorithm to be run under the same machine load pressure. As can be seen from the results, the CPU memory usage of different algorithms is quite similar, as the CPU memory usage is primarily determined by the sample replay buffer. However, compared to the baseline Instance Adv. and Full-Comm algorithm, our algorithm clearly requires more time and GPU memory usage. The main reason for this is that our

**Table A3** Additional test results for the AME baseline

		MA3C	AME
SMAC-1o_2r_vs_4r	Normal	0.86±0.02	0.81±0.05
	Random Noise	0.84±0.02	0.76±0.07
	Aggressive Attackers	0.81±0.01	0.60±0.06
GP-4r	Normal	0.87±0.02	0.23±0.37
	Random Noise	0.87±0.02	0.24±0.40
	Aggressive Attackers	0.86±0.01	0.17±0.29



**Table A4** Training efficiency comparison of algorithms

Environment	Algorithm	Time	Memory /GB	Graphics memory /MB
GP-4r	MA3C	5d 2h 56m	2.56	1178
	MA3C w/ ordinary encoder	5d 2h 13m	2.46	1178
	Instance Adv.	1d 5h 36m	2.41	1163
	Full-Comm	9h 9m	2.28	1148
GP-9r	MA3C	4d 13h 16m	2.51	1182
	MA3C w/ ordinary encoder	4d 9h 18m	2.43	1182
	Instance Adv.	1d 5h 18m	2.42	1168
	Full-Comm	8h 31m	2.28	1148

algorithm needs to maintain an additional population of attackers, where the attacker training and the evolution process of the population brings extra time and GPU memory consumption. Nevertheless, the training time and GPU memory usage of MA3C and MA3C w/ ordinary encoder are similar, indicating that the transformer encoder network structure does not have a significant impact on the training efficiency of the algorithm. To some extent, this is reasonable, as we only used a transformer-like network structure without adopting a very large network size. In the future, we will further optimize the algorithm in terms of underlying code and algorithm design to improve the overall training efficiency of our method.

## References

- Zhu C, Dastani M, Wang S. A survey of multi-agent reinforcement learning with communication. 2022, arXiv preprint arXiv: 2203.08975
- Ding Z, Huang T, Lu Z. Learning individually inferred communication for multi-agent cooperation. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 1851
- Wang R, He X, Yu R, Qiu W, An B, Rabinovich Z. Learning efficient multi-agent communication: an information bottleneck approach. In: Proceedings of the 37th International Conference on Machine Learning. 2020, 919
- Xue D, Yuan L, Zhang Z, Yu Y. Efficient multi-agent communication via Shapley message value. In: Proceedings of the 31st International Joint Conference on Artificial Intelligence. 2022, 578–584
- Guan C, Chen F, Yuan L, Wang C, Yin H, Zhang Z, Yu Y. Efficient multi-agent communication via self-supervised information aggregation. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022
- Foerster J N, Assael Y M, de Freitas N, Whiteson S. Learning to communicate with deep multi-agent reinforcement learning. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. 2016, 2145–2153
- Kim W, Park J, Sung Y. Communication in multi-agent reinforcement learning: intention sharing. In: Proceedings of the 9th International Conference on Learning Representations. 2021
- Yuan L, Wang J, Zhang F, Wang C, Zhang Z, Yu Y, Zhang C. Multi-agent incentive communication via decentralized teammate modeling. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence. 2022, 9466–9474
- Chakraborty A, Alam M, Dey V, Chattopadhyay A, Mukhopadhyay D. Adversarial attacks and defences: a survey. 2018, arXiv preprint arXiv: 1810.00069
- Moos J, Hansel K, Abdulsamad H, Stark S, Clever D, Peters J. Robust reinforcement learning: a review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 2022, 4(1): 276–315
- Zhang H, Chen H, Xiao C, Li B, Liu M, Boning D S, Hsieh C J. Robust deep reinforcement learning against adversarial perturbations on state observations. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 1765
- Oikarinen T, Zhang W, Megretski A, Daniel L, Weng T W. Robust deep reinforcement learning through adversarial loss. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2021, 26156–26167
- Xu M, Liu Z, Huang P, Ding W, Cen Z, Li B, Zhao D. Trustworthy reinforcement learning against intrinsic vulnerabilities: robustness, safety, and generalizability. 2022, arXiv preprint arXiv: 2209.08025
- Pan X, Seita D, Gao Y, Canny J. Risk averse robust adversarial reinforcement learning. In: Proceedings of 2019 International Conference on Robotics and Automation. 2019, 8522–8528
- Zhang H, Chen H, Boning D S, Hsieh C J. Robust reinforcement learning on state observations with learned optimal adversary. In: Proceedings of the 9th International Conference on Learning Representations. 2021
- Lin J, Dzevaroska K, Zhang S Q, Leon-Garcia A, Papernot N. On the robustness of cooperative multi-agent reinforcement learning. In: Proceedings of 2020 IEEE Security and Privacy Workshops. 2020, 62–68
- Hu Y, Zhang Z. Sparse adversarial attack in multi-agent reinforcement learning. 2022, arXiv preprint arXiv: 2205.09362
- Xue W, Qiu W, An B, Rabinovich Z, Obraztsova S, Yeo C K. Mis-spoke or mis-lead: achieving robustness in multi-agent communicative reinforcement learning. In: Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems. 2022, 1418–1426
- Vinitzky E, Du Y, Parvate K, Jang K, Abbeel P, Bayen A. Robust reinforcement learning using adversarial populations. 2020, arXiv preprint arXiv: 2008.01825
- Wang T, Wang J, Zheng C, Zhang C. Learning nearly decomposable value functions via communication minimization. In: Proceedings of the 8th International Conference on Learning Representations. 2020
- Das A, Gervet T, Romoff J, Batra D, Parikh D, Rabbat M, Pineau J. TarMAC: targeted multi-agent communication. In: Proceedings of the 36th International Conference on Machine Learning. 2019, 1538–1546
- Sukhbaatar S, Szlam A, Fergus R. Learning multiagent communication with backpropagation. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. 2016, 2252–2260
- Lowe R, Foerster J N, Boureau Y, Pineau J, Dauphin Y N. On the pitfalls of measuring emergent communication. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. 2019, 693–701
- Eccles T, Bachrach Y, Lever G, Lazaridou A, Graepel T. Biases for emergent communication in multi-agent reinforcement learning. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019, 1176
- Mao H, Zhang Z, Xiao Z, Gong Z, Ni Y. Learning agent communication

- under limited bandwidth by message pruning. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence. 2020, 5142–5149
26. Mao H, Zhang Z, Xiao Z, Gong Z, Ni Y. Learning multi-agent communication with double attentional deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 2020, 34(1): 32
  27. Wang Y, Zhong F, Xu J, Wang Y. ToM2C: target-oriented multi-agent communication and cooperation with theory of mind. In: Proceedings of the 10th International Conference on Learning Representations. 2021
  28. Zhang S Q, Zhang Q, Lin J. Efficient communication in multi-agent reinforcement learning via variance based control. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019, 291
  29. Zhang S Q, Zhang Q, Lin J. Succinct and robust multi-agent communication with temporal message control. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 1449
  30. Mitchell R, Blumenkamp J, Prorok A. Gaussian process based message filtering for robust multi-agent cooperation in the presence of adversarial communication. 2020, arXiv preprint arXiv: 2012.00508
  31. Sun Y, Zheng R, Hassanzadeh P, Liang Y, Feizi S, Ganesh S, Huang F. Certifiably robust policy learning against adversarial multi-agent communication. In: Proceedings of the 11th International Conference on Learning Representations. 2023
  32. OroojlooyJadid A, Hajinezhad D. A review of cooperative multi-agent deep reinforcement learning. 2019, arXiv preprint arXiv: 1908.03963
  33. Christianos F, Papoudakis G, Rahman M A, Albrecht S V. Scaling multi-agent reinforcement learning with selective parameter sharing. In: Proceedings of the 38th International Conference on Machine Learning. 2021, 1989–1998
  34. Wang J, Ren Z, Han B, Ye J, Zhang C. Towards understanding cooperative multi-agent Q-learning with value factorization. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2021, 29142–29155
  35. Papoudakis G, Christianos F, Rahman A, Albrecht S V. Dealing with non-stationarity in multi-agent deep reinforcement learning. 2019, arXiv preprint arXiv: 1906.04737
  36. Peng Z, Li Q, Hui K M, Liu C, Zhou B. Learning to simulate self-driven particles system with coordinated policy optimization. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2021, 10784–10797
  37. Kouzehgar M, Meghjeni M, Bouffanais R. Multi-agent reinforcement learning for dynamic ocean monitoring by a swarm of buoys. In: Proceedings of Global Oceans 2020. 2020, 1–8
  38. Wang J, Xu W, Gu Y, Song W, Green T C. Multi-agent reinforcement learning for active voltage control on power distribution networks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2021, 3271–3284
  39. Xue K, Xu J, Yuan L, Li M, Qian C, Zhang Z, Yu Y. Multi-agent dynamic algorithm configuration. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022
  40. Guo J, Chen Y, Hao Y, Yin Z, Yu Y, Li S. Towards comprehensive testing on the robustness of cooperative multi-agent reinforcement learning. In: Proceedings of 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2022, 114–121
  41. Li S, Wu Y, Cui X, Dong H, Fang F, Russell S. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence. 2019, 4213–4220
  42. Lowe R, Wu Y, Tamar A, Harb J, Abbeel P, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 6382–6393
  43. T. van der Heiden T, Salge C, Gavves E, van Hoof H. Robust multi-agent reinforcement learning with social empowerment for coordination and communication. 2020, arXiv preprint arXiv: 2012.08255
  44. Zhang K, Sun T, Tao Y, Gene S, Mallya S, Başar T. Robust multi-agent reinforcement learning with model uncertainty. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 887
  45. Phan T, Gabor T, Sedlmeier A, Ritz F, Kempter B, Klein C, Sauer H, Schmid R, Wieghardt J, Zeller M, Linnhoff-Popien C. Learning and testing resilience in cooperative multi-agent systems. In: Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems. 2020, 1055–1063
  46. Phan T, Belzner L, Gabor T, Sedlmeier A, Ritz F, Linnhoff-Popien C. Resilient multi-agent reinforcement learning with adversarial value decomposition. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence. 2021, 11308–11316
  47. Jaderberg M, Dalibard V, Osindero S, Czarnecki W M, Donahue J, Razavi A, Vinyals O, Green T, Dunning I, Simonyan K, Fernando C, Kavukcuoglu K. Population based training of neural networks. 2017, arXiv preprint arXiv: 1711.09846
  48. Jaderberg M, Czarnecki W M, Dunning I, Marris L, Lever G, Castañeda A G, Beattie C, Rabinowitz N C, Morcos A S, Ruderman A, Sonnerat N, Green T, Deason L, Leibo J Z, Silver D, Hassabis D, Kavukcuoglu K, Graepel T. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 2019, 364(6443): 859–865
  49. Qian H, Yu Y. Derivative-free reinforcement learning: a review. *Frontiers of Computer Science*, 2021, 15(6): 156336
  50. Derek K, Isola P. Adaptable agent populations via a generative model of policies. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2021, 3902–3913
  51. Parker-Holder J, Pacchiano A, Choromanski K, Roberts S. Effective diversity in population based reinforcement learning. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 1515
  52. Luo F M, Xu T, Lai H, Chen X H, Zhang W, Yu Y. A survey on model-based reinforcement learning. 2022, arXiv preprint arXiv: 2206.09328
  53. Zhao R, Song J, Haifeng H, Gao Y, Wu Y, Sun Z, Wei Y. Maximum entropy population based training for zero-shot human-AI coordination. 2021, arXiv preprint arXiv: 2112.11701
  54. Xue K, Wang Y, Yuan L, Guan C, Qian C, Yu Y. Heterogeneous multi-agent zero-shot coordination by coevolution. 2022, arXiv preprint arXiv: 2208.04957
  55. Wang Y, Xue K, Qian C. Evolutionary diversity optimization with clustering-based selection for reinforcement learning. In: Proceedings of the 10th International Conference on Learning Representations. 2021
  56. Cully A, Demiris Y. Quality and diversity optimization: a unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 2018, 22(2): 245–259
  57. Chatzilygeroudis K, Cully A, Vassiliades V, Mouret J B. Quality-diversity optimization: a novel branch of stochastic optimization. In: Pardalos P M, Rasskazova V, Vrahatis M N, eds. *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*. Cham: Springer, 2021, 109–135
  58. Lim B, Grillotti L, Bernasconi L, Cully A. Dynamics-aware quality-diversity for efficient learning of skill repertoires. In: Proceedings of 2022 International Conference on Robotics and Automation. 2022,

5360–5366

59. Pierrot T, Richard G, Beguir K, Cully A. Multi-objective quality diversity optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference. 2022, 139–147
60. Chalumeau F, Boige R, Lim B, Macé V, Allard M, Flajolet A, Cully A, Pierrot T. Neuroevolution is a competitive alternative to reinforcement learning for skill discovery. 2022, arXiv preprint arXiv: 2210.03516
61. Samvelyan M, Rashid T, de Witt C S, Farquhar G, Nardelli N, Rudner T G J, Hung C M, Torr P H S, Foerster J, Whiteson S. The StarCraft multi-agent challenge. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. 2019, 2186–2188
62. Oliehoek F A, Amato C. A Concise Introduction to Decentralized POMDPs. Cham: Springer, 2016
63. Mnih V, Kavukcuoglu K, Silver D, Rusu A A, Veness J, Bellemare M G, Graves A, Riedmiller M, Fidjeland A K, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. *Nature*, 2015, 518(7540): 529–533
64. Gronauer S, Diepold K. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 2022, 55(2): 895–943
65. Zhang K, Yang Z, Başar T. Multi-agent reinforcement learning: a selective overview of theories and algorithms. In: Vamvoudakis K G, Wan Y, Lewis F L, Cansever D, eds. *Handbook of Reinforcement Learning and Control*. Cham: Springer, 2021, 321–384
66. Rashid T, Samvelyan M, Schroeder C, Farquhar G, Foerster J, Whiteson S. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In: Proceedings of the 35th International Conference on Machine Learning. 2018, 4295–4304
67. Foerster J, Farquhar G, Afouras T, Nardelli N, Whiteson S. Counterfactual multi-agent policy gradients. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence. 2018, 2974–2982
68. Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: Proceedings of the 35th International Conference on Machine Learning. 2018, 1587–1596
69. Cully A. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In: Proceedings of the Genetic and Evolutionary Computation Conference. 2019, 81–89
70. Zhou Z, Fu W, Zhang B, Wu Y. Continuously discovering novel strategies via reward-switching policy optimization. In: Proceedings of the 10th International Conference on Learning Representations. 2022
71. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser Ł, Polosukhin I. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 6000–6010
72. Zhou Z H. Open-environment machine learning. *National Science Review*, 2022, 9(8): nwac123



Lei Yuan received the BSc degree in Department of Electronic Engineering in 2016 from Tsinghua University, China and the MSc degree from Chinese Aeronautical Establishment, China in 2019. He is currently pursuing the PhD degree with the Department of Computer Science and Technology, Nanjing University, China. His current research interests mainly include machine learning, reinforcement learning, and multi-agent reinforcement learning.



Feng Chen received his BSc degree in Automation from School of Artificial Intelligence, Nanjing University, China in 2022. He is currently pursuing the MSc degree with the School of Artificial Intelligence, Nanjing University, China. His research interests include multi-agent reinforcement learning, multiagent system.



Zongzhang Zhang received his PhD degree in computer science from University of Science and Technology of China, China in 2012. He was a research fellow at the School of Computing, National University of Singapore, Singapore from 2012 to 2014, and a visiting scholar at the Department of Aeronautics and Astronautics, Stanford University, USA from 2018 to 2019. He is currently an associate professor at the National Key Laboratory for Novel Software Technology, Nanjing University, China. He has co-authored more than 50 research papers. His research interests include reinforcement learning, intelligent planning, and multi-agent learning.



Yang Yu received the PhD degree in computer science from Nanjing University, China in 2011, and is currently a professor at the School of Artificial Intelligence, Nanjing University, China. His research interests include machine learning, mainly reinforcement learning and derivative-free optimization for learning. Prof. Yu was granted the CCF-IEEE CS Young Scientist Award in 2020, recognized as one of the AI's 10 to Watch by IEEE Intelligent Systems, and received the PAKDD Early Career Award in 2018. His team won the Champion of the 2018 OpenAI Retro Contest on transfer reinforcement learning and the 2021 ICAPS Learning to Run a Power Network Challenge with Trust. He served as Area Chairs for NeurIPS, ICML, IJCAI, AAAI, etc.