REDUCING COMMUNICATION ENTROPY IN MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Communication in multi-agent reinforcement learning has been drawing attention recently for its significant role in cooperation. However, multi-agent systems may suffer from limitations on communication resource and thus need efficient communication techniques in real-world scenarios. According to the Shannon-Hartley theorem, messages to be transmitted reliably in worse channels requires lower entropy. Therefore, we aim to reduce message entropy in multi-agent communication. A fundamental challenge in this is that the gradients of entropy are either 0 or ∞, disabling gradient-based methods. To handle it, we propose a pseudo gradient descent scheme, which reduces entropy by adjusting the distributions of messages wisely. We conduct experiments on six environment settings and two base communication frameworks and find that our scheme can reduce communication entropy by up to 90% with nearly no loss of performance.

1 INTRODUCTION

Over these years, multi-agent reinforcement learning (MARL) has been attracting increasing attention for its broad applications in cooperative tasks, such as robots navigation (Han et al., 2020), traffic lights control (Calvo & Dusparic, 2018) and large-scale fleet management (Lin et al., 2018). To promote the cooperation of agents, a few researchers have designed communication protocols among agents and got good results (Ahilan & Dayan, 2020; Chu et al., 2019; Kim et al., 2020).

However, many multi-agent communication frameworks use cooperation scores as the only metric and do not take communication efficiency into account, making them impractical in scenarios where communication resources are limited. (Rangwala & Williams, 2020; Serra-Gómez et al., 2020; Sun et al., 2020). Some others try to design efficient multi-agent communication protocols, whose work can be divided into two categories. The first is decreasing communication times (Ma et al., 2021; Kim et al., 2018; Vijay et al., 2021), including wisely choosing communication timing and partners. The second is reducing communication entropy. The motivation is that messages with lower entropy can be reliably transmitted over worse communication channels according to the Shannon-Hartley Theorem (Shannon, 1948). Most learning-based multi-agent communication frameworks use continuous variables to communicate, and hence works in this area aim to minimize differential entropy[1] (Wang et al., 2019; 2020; Zhang et al., 2020).

Nevertheless, these methods have two defects. Firstly, some of them rely on specially designed architectures to minimize communication entropy (Zhang et al., 2020), impairing their generalizability. Secondly, differential entropy is hard to estimate without prior information, and some methods simply treat the message distributions as single Gaussian, which may be far from reality. We also notice that reducing differential entropy is less significant than reducing discrete entropy of quantized messages. This is because continuous variables must be quantized to discrete variables before being transmitted in a modern communication system (Shannon, 1948), making discrete entropy much more important than differential entropy when considering efficient communication.

In this paper, we propose a scheme, Discrete Entropy Minimization (abbreviated as DisEM), that can be applied to common MARL communication frameworks and reduce the discrete entropy

[1] Entropy of discrete variables and continuous variables is defined differently in Shannon (1948). Following Cover (1999), we use discrete entropy to denote the entropy of discrete variables and differential entropy to denote the entropy of continuous variables for ease of reading.

of quantized messages of them with little performance decline. The core problem of doing so is that quantization truncates gradients, making all gradient-based training algorithms infeasible. To overcome this challenge, we put forward a novel pseudo gradient descent method that reduces discrete entropy by adjusting the distributions of messages according to well-designed pseudo gradients. We also theoretically prove its effectiveness. An intuitive description of how our DisEM changes the message distribution is that it makes message variables move from less popular quantization intervals to adjacent more popular ones. As a result, DisEM does not change the message distribution too much and hence manages to reduce entropy with little performance degradation. To empirically illuminate DisEM's effectiveness, we conduct experiments in three communication-critical multi-agent tasks with six settings in total. Meanwhile, we apply our scheme to two base multi-agent communication frameworks, IC3NET (Singh et al., 2018) and TARMAC (Das et al., 2019), to manifest its generalizability. Experiments show that DisEM can reduce up to 90% entropy without performance degradation in some settings. To sum up, our contributions are listed as follows:

- We propose a light-weighted yet effective scheme DisEM that can be incorporated into common learning-based multi-agent communication frameworks and reduce the message entropy of them with nearly no loss of performance.
- We overcome the problem that discrete entropy of quantized messages cannot be reduced with gradient-based methods. Specifically, we propose a novel pseudo gradient descent method and theoretically prove its ability to reduce entropy.
- We conduct adequate experiments to confirm DisEM's preponderance over previous methods. Besides, we execute several investigative experiments to further illuminate the features of our scheme, including communication simulations demonstrating how low entropy benefits multi-agent communication in noisy scenarios.

2 Related work

Communication in MARL has been a hot area of research since 2016. Foerster et al. (2016) suggest that generating differentiable messages and letting gradients flow between agents are beneficial for multi-agent cooperation, laying the foundation for learning-based multi-agent communication frameworks. Sukhbaatar et al. (2016) put forward COMMNET, where agents broadcast their hidden states to others for collaboration. Singh et al. (2018) propose IC3NET based on COMMNET with two advancements: (1) agents are trained with individualized rewards; (2) agents adopt a gating mechanism to learn when to communicate. Inspired by Transformer (Vaswani et al., 2017), Das et al. (2019) put forward TARMAC, an attention-based method. More techniques have been used to enhance communication performance in the past year or two, such as hierarchical communication (Sheng et al., 2020), relabelling history messages (Ahilan & Dayan, 2020), and intention sharing (Kim et al., 2020). Nevertheless, these frameworks have no concern for communication overhead, making them less practical in real-world applications.

In terms of efficient communication, most researchers choose to wisely select when to communicate and whom to communicate with. Work in this area can be divided into three categories. (1) The multi-agent system adopts a scheduler to decide who can communicate at each step (Kim et al., 2018; Wang et al., 2020). (2) Agents use a learnable gating mechanism to control communication (Jiang & Lu, 2018; Vijay et al., 2021). (3) Agents schedule communication according to predefined rules (Ding et al., 2020; Ma et al., 2021; Zhang et al., 2019).

Some other studies focus on minimizing communication entropy. Zhang et al. (2020) succeed in reducing the variance of sending messages without degrading performance. However, the scheme relies on specially designed communication and decision architectures and cannot be applied to other communication frameworks. Wang et al. (2019) force message generators of agents to output Gaussian variables with fixed variance. In this way, the differential entropy of messages can be decreased by minimizing the mutual information between the output values and input features of the message generators. Wang et al. (2020) extend this scheme by introducing the concept of information bottleneck (Tishby et al., 2000). Although this kind of scheme can be applied to some base communication frameworks, it has three defects. Firstly, it requires the message generators to output Gaussian variables, which may lower performance. Secondly, it only guarantees to reduce the upper bound of mutual information instead of differential entropy. Thirdly, it models the output messages as single Gaussian distributions with predefined variances, which may be far from reality.

In our experiments, we refer to this kind of scheme as Differential Entropy Minimization (DifEM) and compare it with our scheme.

3 PRELIMINARIES

3.1 DISCRETE ENTROPY AND DIFFERENTIAL ENTROPY

Discrete entropy (Shannon, 1948) is a measure of uncertainty of a discrete random variable. Let X be a discrete random variable with alphabet $\mathcal{X}$ and a probability mass function $p(x)$, the discrete entropy $H(X)$ of X is defined by

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{1}$$

In comparison, the uncertainty of a discrete random variable is measured by differential entropy (Shannon, 1948). Suppose a continuous random variable Y has a probability density function $f(y)$ with support set S, then its differential entropy $h(Y)$ is calculated as

$$h(Y) = -\int_S f(y) \log f(y) dy \tag{2}$$

There are two important differences between them. Firstly, discrete entropy is usually easier to accurately estimate than differential entropy. The reason is that given enough samples, the probability mass function of a discrete variable can be easily estimated. In contrast, the probability density function of a continuous variable is hard to estimate without knowing the prior distribution. Secondly, since modern communication systems utilize discrete symbols to carry information (Shannon, 1948), discrete entropy is much more critical than differential entropy when considering efficient communication. Therefore, we choose to minimize discrete entropy of quantized messages.

3.2 THE IMPORTANCE OF LOW ENTROPY

The Shannon-Hartley theorem (Shannon, 1948) reveals the maximum rate at which information can be transmitted over a communication channel:

Theorem 3.1. *(The Shannon-Hartley theorem) Given a noisy channel with channel capacity C and information rate R, if $R < C$, then there exists a coding technique that allows the probability of error at the receiver to be made arbitrarily small. Otherwise, an arbitrarily small probability of error is not achievable. The channel capacity C is calculated as follows:*

$$C = Blog(1 + SNR) \tag{3}$$

where B is the bandwidth and SNR is the signal-to-noise ratio.

The information rate R of a source is calculated by $R = rH$, where H is the averaged entropy of sending messages and r is the rate at which the messages are generated. It can be concluded from the theorem that a multi-agent communication protocol with lower entropy is more reliable when the communication resources are limited. We run several communication simulations in Sec.5.3 to illustrate this point.

3.3 MULTI-AGENT REINFORCEMENT LEARNING WITH COMMUNICATION

We consider a partially observable n-agent Markov game (Littman, 1994) with communication among agents. This process can be described with a tuple $\langle S, A, R, T, O, \Omega, M_S, n, \gamma \rangle$, where S denotes the state space of the environment, A denotes the set of available actions, R is the reward function $R : S \times A \to \mathbb{R}$, T is the transition function $T : S \times A \to S$, O is the observation space of agents, Ω is the observation function for agents: $\Omega : S \to O$, M_S denotes the message space, n represents the number of agents, and γ is the discount factor. In a basic MARL framework with communication, an agent i needs a policy π^i and a message generator g^i to finish tasks. At timestep t, agent i firstly obtains observation o_t^i from the environment and receives messages $\vec{m}_{t-1}^{recv,i}$ from others:

$$\vec{m}_{t-1}^{recv,i} = \{\vec{m}_{t-1}^1, ..., \vec{m}_{t-1}^{i-1}, \vec{m}_{t-1}^{i+1}, ..., \vec{m}_{t-1}^n\}$$

Secondly, it makes decisions $a_t^i = \pi^i(o_t^i, \vec{m}_{t-1}^{recv,i})$ and broadcasts message $\vec{m}_t^i = g^i(o_t^i, \vec{m}_{t-1}^{recv,i})$ to others. Finally, it gets rewards r_t^i from the environment.

In our paper, we focus on learning-based multi-agent communication frameworks where g^i is learned using gradient-based methods and generates continuous messages. In this case, we can use θ_i to denote the parameters of its policy and message generator, and the training goal is to maximize the objective function $J(\theta_i)$:

$$J(\theta_i) = \mathbb{E}_{\pi_i, g_i}\left[\sum_{t=0}^{\infty} \gamma^t r_t^i\right] \tag{4}$$

In our experiments, the action space is discrete and we choose policy gradient methods to optimize $J(\theta_i)$:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}[\pi^i(o,m,a)A(o,m,a)\nabla_{\theta_i} log\pi^i(o,m,a)] \tag{5}$$

where $\pi^i(o,m,a)$ represents the possibility of choosing action a given (o,m) and $A(o,m,a)$ is the advantage function.

4 Methods

4.1 Quantize messages during testing

Communication frameworks in MARL usually use continuous messages so that message generators can be trained using gradient-based methods. In our paper, we still use continuous messages during training, but quantize them during testing. We do this for three reasons: (1) Proper quantization does not hinder the exchange of information between agents: the performance of agents remains the same when we quantize the messages during testing. (2) Quantization is necessary in modern communication systems. (3) Compared with differential entropy, discrete entropy is easier to accurately estimate and more important in communication.

Specifically, we set the output range of agents' message generators to $[-1, 1]$ and use a uniform quantization function $f^Q(x)$ with quantization interval length Δ:

$$f^Q(x) = k\Delta - 1,\ \forall x \in [(k-0.5)\Delta - 1, (k+0.5)\Delta - 1),\ k \in \{0, 1, ..., K\} \tag{6}$$

where $K = 2/\Delta$ and $K+1$ is the number of quantization intervals.

4.2 Entropy of quantized messages

To formulate how entropy of quantized messages is calculated, we define $h_k(\cdot)$ below:

$$h_k(x) = \begin{cases} 1,\ x \in [(k-0.5)\Delta - 1, (k+0.5)\Delta - 1) \\ 0,\ x \notin [(k-0.5)\Delta - 1, (k+0.5)\Delta - 1) \end{cases} \tag{7}$$

Without loss of generality, we firstly focus on the setting where the message length is 1, which means one piece of message is a number instead of a vector. Given a message set $M = \{m_1, m_2, ..., m_N\}$, the entropy of quantized messages is calculated below:

$$H(M) = -\sum_{k=0}^{K} \frac{\epsilon + \sum_{i=1}^{N} h_k(m_i)}{N} \log \frac{\epsilon + \sum_{i=1}^{N} h_k(m_i)}{N} \tag{8}$$

where ϵ is a small number to avoid $\log 0$ in calculation. If the message length is more than 1, the entropy can be calculated by summing up the entropy of each digit.

4.3 Reduce entropy without reducing performance

We intend to reduce the message entropy of gradient-based multi-agent communication frameworks without harming performance. In specific, for agent i, firstly we use the baseline framework to train θ_i until π^i, g^i are well-trained. Suppose $J(\theta_i) = J_p$ at this time. Secondly, we try to optimize this constrained objective:

$$\min_{\theta_i} H(M_i)\ s.t.\ J(\theta_i) > J_p \tag{9}$$

where $H(M_i)$ represents the message entropy of agent i. With the introduction of a Lagrange multiplier α, we get an unconstrained optimization objective:

$$\max_{\theta_i} J'(\theta_i) = J(\theta_i) - \alpha H(M_i) \tag{10}$$

Then the gradient of θ_i with respect to this new objective becomes:

$$\nabla_{\theta_i} J'(\theta_i) = \nabla_{\theta_i} J(\theta_i) - \alpha \nabla_{\theta_i} H(M_i) \tag{11}$$

Note that the gradient of $h_k(\cdot)$ is either 0 or ∞, consequently, $\nabla_{\theta_i} H(M_i)$ is either 0 or ∞. This disables gradient-based training methods and thus makes $H(M_i)$ hard to be reduced. We propose a pseudo gradient descent method to handle this problem.

4.4 Reduce entropy with pseudo gradient descent

$H(M)$ can be treated as a multivariate function with N variables $H(m_1, m_2, ..., m_N)$, and in this part we focus on how to reduce $H(M)$ by adjusting m_i. To start with, we present the core function of gradient descent methods: given a continuously differentiable function $f(x_1, x_2, ..., x_n)$ and $\eta \to 0$,

$$\begin{gathered} x_i' = x_i - \eta \nabla_{x_i} f \\ f(x_1, ..., x_i', ..., f_n) \leq f(x_1, ..., x_i, ..., f_n) \end{gathered} \tag{12}$$

Since $\nabla_{m_i} H(M) = \sum_k \nabla_{h_k} H(M) \nabla_{m_i} h_k = 0$ or ∞, we cannot use gradient descent to minimize $H(M)$. Therefore, we try to design a pseudo gradient $\nabla^p_{m_i} H(M)$ to achieve a similar effect to Eqa. 12.

Our core idea is to replace $\nabla_{m_i} h_k$ with $s_k(m_i)$:

$$s_k(x) = \begin{cases} 1, x \in ((k-1)\Delta - 1, k\Delta - 1) \\ -1, x \in (k\Delta - 1, (k+1)\Delta - 1) \\ 0, x = k\Delta - 1 \end{cases} \tag{13}$$

and the expression for pseudo gradient $\nabla^p_{m_i} H(M)$ is :

$$\nabla^p_{m_i} H(M) = \sum_k \nabla_{h_k} H(M) s_k(m_i) \tag{14}$$

Next, we show that pseudo gradient descent has a similar property to gradient descent. Given $H(M) = H(m_1, ..., m_i, ..., m_n)$ and $\eta \to 0$,

$$\begin{gathered} m_i' = m_i - \eta \nabla^p_{m_i} H(M) \\ H(m_1, ..., m_i', ..., m_N) \leq H(m_1, ..., m_i, ..., m_n) \end{gathered} \tag{15}$$

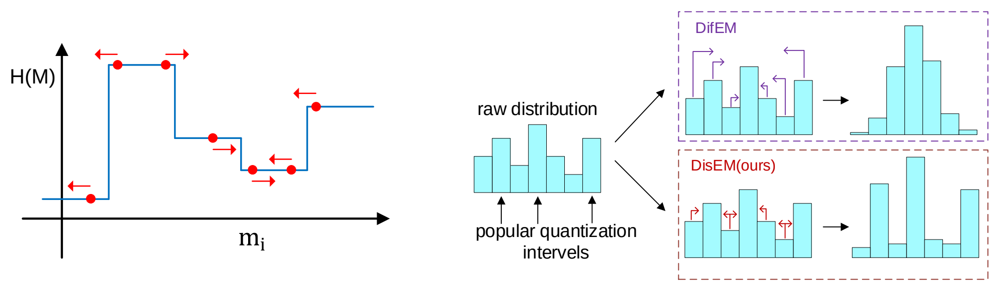
Proof see Appendix A.

We visualize how pseudo gradient descent method adjusts m_i in Fig. 1(a) and how our scheme and previous schemes (those that aim to minimize differential entropy) change the distribution of messages in Fig. 1(b). In brief, our scheme reduces entropy by moving numbers from less popular quantization intervals to adjacent more popular ones and does not change the messages too much; As a comparison, traditional schemes simply make the distribution more like a low variance Gaussian distribution regardless of what the original distribution is.

4.5 Implementation details

MARL frameworks with communication commonly use gradient-based methods to update the parameters of neural networks. To make our scheme compatible with common implementations in deep learning platforms (e.g. PyTorch), we can use the $nograd(\cdot)$ operation, which means treating the gradient of this part as 0 during derivation. And we can find u_k s.t. $\nabla_{m_i} u_k = s_k(m_i)$. From the perspective of writing codes, the expression for $H(M)$ is:

$$\begin{aligned} H(M) &= -\sum_{k=0}^{K} \frac{\epsilon + \sum_{i=1}^{N} h_k'(m_i)}{N} \log \frac{\epsilon + \sum_{i=1}^{N} h_k'(m_i)}{N} \\ h_k'(m_i) &= nograd(h_k(m_i) - u_k(m_i)) + u_k(m_i) \end{aligned} \tag{16}$$



(a) (b)

Figure 1: (a) How our pseudo gradient descent method reduces $H(M)$ by adjusting m_i. The x-coordinate of a red dot represents a possible value for m_i, and the y-coordinate of it represents corresponding value for $H(M)$. Red arrows indicate directions of pseudo gradient descent. Pseudo gradient descent makes m_i moves to a direction that might reduce $H(M)$. (b) How our scheme (Discrete Entropy Minimization, abbreviated as DisEM) and traditional scheme (Differential Entropy Minimization, abbreviated as DifEM) change the distributions of messages. DisEM reduces entropy by moving numbers from less popular quantization intervals to adjacent more popular ones, while DifEM simply make the distribution more like a low variance Gaussian distribution.

Following policy gradient methods, the parameters θ_i of agent i can be trained according to the following expression:

$$\nabla_{\theta_i} J'(\theta_i) = \nabla_{\theta_i} J(\theta_i) - \alpha \nabla^p_{\theta_i} H(M_i) \tag{17}$$

Besides, we first train the agents without the entropy regularizer (i.e. set $\alpha = 0$) to make them optimize their policies and message generators. After T_N epochs, we add the regularizer (i.e. set $\alpha = \alpha_p$) and continue to train them for another $T_{max} - T_N$ epochs. T_N, α_p and T_{max} are predefined hyperparameters, whose values are present in Appendix B. Additionally, we set $\Delta = 0.25$ for the quantization function.

5 Experiments

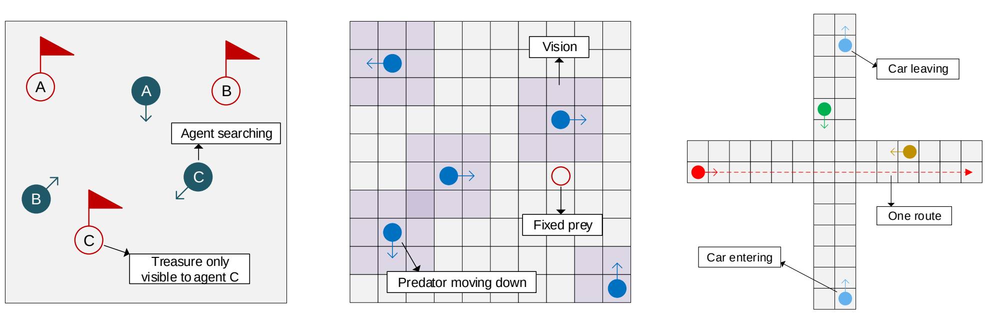


Figure 2: Visualizations of Treasure Hunt, Predator Prey, and Traffic Junction.

5.1 Tested algorithms

To test the effectiveness of our scheme, the tested algorithms are based on two multi-agent communication frameworks: IC3NET (Singh et al., 2018) and TARMAC (Das et al., 2019), which are elaborated in Appendix D.1 and Appendix D.2. In particular, we test four variants of each framework as described below. (1) **Original**: This is the original version of the framework without any modifications, which manifests baseline performance and entropy. (2) **ZeroComm**: This variant disables communication by compulsively setting all messages to zero and manifests performance without communication. (3) **DifEM**: (Wang et al., 2020) A previous method that minimizes the differential entropy with a mutual information regularizer. (4) **DisEM**: This is our proposed scheme which reduces entropy by adding a pseudo entropy regularizer to the original loss function.

5.2 ENVIRONMENTS AND RESULTS

We consider three environments, each with two settings, for demonstration purposes: Treasure Hunt (TH), Predator Prey (PP), and Traffic Junction (TJ)(Singh et al., 2018), which are visualized in Fig. 2. We also try some popular MARL environments, for example, SMAC (Samvelyan et al., 2019) and MPE (Lowe et al., 2017). However, we find that communication is not crucial in these tasks and several MARL frameworks have achieved high scores without communication in them. In comparison, these three environments we choose are all communication-critical, where reducing communication entropy is meaningful.

Table 1: Results for experiments in Treasure Hunt tasks

	Setting A		Setting B	
	timesteps↓	entropy↓	timesteps↓	entropy↓
IC3NET-Original	11.7±0.2	139±4	15.6±0.2	127±2
IC3NET-ZeroComm	0.3±0.1	0	0.1±0.0	0
IC3NET-DifEM	10.9±0.2	70±3	17.1±3.3	39±13
IC3NET-DisEM(ours)	11.0±0.1	15±1	15.9±0.7	30±1
TARMAC-Original	11.8±0.1	70±1	39.0±0.3	72±2
TARMAC-ZeroComm	0.3±0.1	0	0.1±0.0	0
TARMAC-DifEM	10.0±0.3	29±1	23.7±3.0	25±3
TARMAC-DisEM(ours)	10.8±0.1	9±1	37.2±0.2	15±1

Treasure Hunt In this task, N agents work together to hunt treasures in the field. Each agent obtains the coordinates of its treasure, which is invisible to others. Note that an agent cannot collect its treasure by itself. Instead, it should help others hunt it by broadcasting the coordinates. The field size is 1×1, and an agent can move in eight directions at speed v. One episode ends if all treasures are found or the timestep reaches the upper limit t_{max}. Therefore, smaller timesteps indicate better performance. In setting A (TH-A), $N = 3, v = 0.15$ and $t_{max} = 20$. In setting B (TH-B), $N = 6, v = 0.09$ and $t_{max} = 60$. We present the experiment results in Table 1. Our scheme successfully reduces $75\% \sim 89\%$ entropy without lowering performance. It even improves performance slightly in some settings: TARMAC-DisEM achieves lower timesteps than TARMAC-Original both in TH-A and TH-B. The reason is that reducing entropy removes redundant information, making it easier for agents to extract useful information from messages.

Table 2: Results for experiments in Predator Prey tasks

	Setting A		Setting B	
	timesteps↓	entropy↓	timesteps↓	entropy↓
IC3NET-Original	9.9±0.1	94±2	23.5±0.7	122±6
IC3NET-ZeroComm	2.6±0.3	0	6.9±0.4	0
IC3NET-DifEM	5.6±1.3	20±7	22.5±0.3	67±3
IC3NET-DisEM(ours)	9.3±0.2	9±1	23.7±0.3	27±2
TARMAC-Original	10.0±0.1	60±2	24.0±0.8	58±3
TARMAC-ZeroComm	2.5±0.1	0	6.4±0.3	0
TARMAC-DifEM	8.7±0.2	13±1	23.2±0.3	25±2
TARMAC-DisEM(ours)	9.6±0.1	6±1	24.6±0.3	14±1

Predator Prey (Singh et al., 2018) In this task, N agents with limited vision are required to reach a fixed prey in a grid world of size $D \times D$. One episode ends if all agents reach the prey or the timestep reaches the upper limit t_{max}. Therefore, smaller timesteps indicate better performance. In setting A (PP-A), $N = 3$, $D = 5$, $t_{max} = 20$ and the vision is set to 0. In setting B (PP-B), $N = 5$, $D = 10$, $t_{max} = 40$ and the vision is set to 1. Due to the severely limited perception, agents must communicate with others to finish tasks earlier. For example, the first agent to reach the prey can guide others by broadcasting its coordinates. The experiment results in Table 2 confirm that our scheme reduces $75\% \sim 90\%$ entropy with little or no performance degradation.

Table 3: Results for experiments in Traffic Junction tasks

	Setting A		Setting B	
	success rates↑	entropy↓	success rates↑	entropy↓
IC3NET-Original	0.866±0.063	141±5	0.946±0.008	75±15
IC3NET-ZeroComm	0.291±0.010	0	0.739±0.018	0
IC3NET-DifEM	0.655±0.050	96±13	0.730±0.012	77±16
IC3NET-DisEM(ours)	0.846±0.055	83±5	0.925±0.007	38±15
TARMAC-Original	0.774±0.092	27±2	0.946±0.009	44±1
TARMAC-ZeroComm	0.279±0.012	0	0.723±0.011	0
TARMAC-DifEM	0.712±0.021	27±7	0.904±0.043	22±5
TARMAC-DisEM(ours)	0.738±0.106	27±2	0.952±0.008	14±3

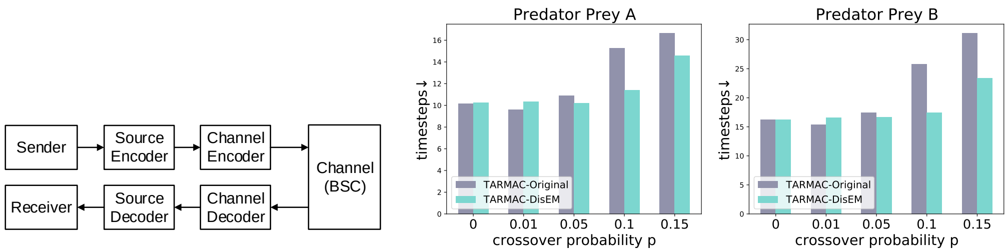
Traffic Junction (Sukhbaatar et al., 2016) In this task, cars enter a junction from all entry points with a probability p_{arr}. The maximum number of cars present is set to N. Each car is assigned a fixed route, and they have two options at each step: move along the route or stay. Cars are required to finish routes quickly without collisions. If no collision happens after t_{max}, this episode is counted as a success, or else a failure. It is worth noticing that a car only observes its location without knowing whether there is any car in front of it. Therefore, agents must communicate to learn the locations of other cars, thereby avoiding collisions. In setting A (TJ-A), $N = 5$, $p_{arr} = 0.3$ and $t_{max} = 20$. In setting B (TJ-B), $N = 10$, $p_{arr} = 0.05$ and $t_{max} = 40$. Results are shown in Table 3. We notice that both DifEM and DisEM do not reduce entropy as much as in the previous two environments. TARMAC-DifEM and TARMAC-DisEM even fail to reduce entropy in TJ-A compared with TARMAC-Original. It is because the message distribution in this setting is hard to optimize.

We conclude two facts from the experiments above. (1) ZeroComm misbehaves in almost all settings, which reflects the importance of communication in these experiments. (2) DifEM indeed reduces communication entropy at the cost of degrading performance more or less, while our scheme DisEM reduces entropy more with little or no performance degradation.

5.3 Investigative experiments

This subsection aims to answer the following questions that may aid in illuminating the features of our scheme.

How does low entropy benefit multi-agent communication? To better demonstrate the importance of low entropy communication, we build a basic digital communication system (John Proakis, 2007) and test the performance of TARMAC-Original and TARMAC-DisEM in Predator Prey environments. The overall design of the communication system is shown in Fig. 3(a), where the purpose of source coding is compressing information, and the purpose of channel coding is to achieve reliable communication over a noisy channel by adding redundancy to the codes. Besides, a binary symmetric channel (BSC) with crossover probability p is a basic channel (Shannon, 1948) with binary input and binary output. The input x and output y of BSC satisfy $p(y = x) = 1 - p$ and $p(y \neq x) = p$. Results demonstrated in Fig. 3 confirm that TARMAC-DisEM works better in bad channel conditions than TARMAC-Original with the help of proper source coding and channel coding. See Appendix C for additional details. Note that these are just simple demonstration experiments, and we have not fully optimized our encoding algorithms.



(a) (b) (c)

Figure 3: (a) The overall design of the communication system, where BSC is short for Binary Symmetric Channel. (b) Performance of TARMAC-Original and TARMAC-DisEM in Predator Prey A with different channel conditions, where larger crossover probability indicates worse channels. (c) Performance of TARMAC-Original and TARMAC-DisEM in Predator Prey B with different channel conditions.

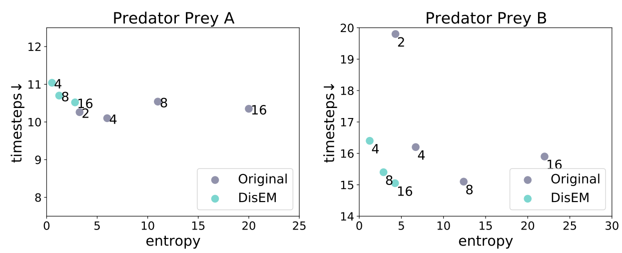


Figure 4: Performance of IC3NET-Original and IC3NET-DisEM in Predator Prey environments. The number near a data point represents its message length, and the color of a data point indicates whether it is an IC3NET-Original model or an IC3NET-DisEM model.

What if we reduce the length of messages? Reducing layer size is an efficient and common way to compress information, as is used in autoencoders (Kramer, 1991). We test IC3NET-Original and IC3NET-DisEM with different message lengths in Predator Prey environments and exhibit results in Fig.4. Each point represents a model's performance with its x-coordinate representing entropy and y-coordinate representing timesteps. Therefore, points located in the bottom left signify good performance and low entropy. In PP-A, DisEM significantly reduces communication entropy with slight performance degradation. In PP-B, DisEM reduces communication entropy massively while maintaining baseline performance. In conclusion, even though reducing the message lengths of IC3NET-Original can reduce entropy, combining it with our scheme is better.

6 Conclusions

In this paper, we propose a simple yet effective scheme, DisEM, to reduce communication entropy for common learning-based multi-agent communication frameworks. Firstly, we point out the necessity of quantization in multi-agent communication and suggest minimizing the entropy of quantized messages. Secondly, to counter the problem that entropy cannot be optimized with gradient descent, we design pseudo gradient descent that reduces entropy by moving message variables from less popular quantization intervals to adjacent more popular ones. Thirdly, we prove the effectiveness of pseudo gradient descent theoretically. Fourthly, we conduct plenty of experiments to test our scheme. Concretely speaking, we test 8 variants of 2 base multi-agent communication frameworks, IC3NET (Singh et al., 2018) and TARMAC (Das et al., 2019), in six environment settings. The results confirm the superiority of our scheme over the existing ones. Fifthly, we conduct some investigative experiments to further manifest our ideas: (1) we run several communication simulations illuminating the importance of low entropy multi-agent communication; (2) we find out that combining our framework with reducing message lengths leads to higher efficiency. We hope our work can provide a foundation for more advanced research in efficient communication.

References

Sanjeevan Ahilan and Peter Dayan. Correcting experience replay for multi-agent communication. In *International Conference on Learning Representations*, 2020.

Jeancarlo Arguello Calvo and Ivana Dusparic. Heterogeneous multi-agent deep reinforcement learning for traffic lights control. In *AICS*, pp. 2–13, 2018.

Tianshu Chu, Sandeep Chinchali, and Sachin Katti. Multi-agent reinforcement learning for networked system control. In *International Conference on Learning Representations*, 2019.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pp. 1538–1546. PMLR, 2019.

Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning individually inferred communication for multi-agent cooperation. *arXiv preprint arXiv:2006.06455*, 2020.

Peter Elias. Coding for noisy channels. *IRE Conv. Rec.*, 3:37–46, 1955.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 29:2137–2145, 2016.

Ruihua Han, Shengduo Chen, and Qi Hao. Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 448–454. IEEE, 2020.

David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *Advances in Neural Information Processing Systems*, 31:7254–7264, 2018.

Masoud Salehi John Proakis. *Digital Communications*. McGraw-Hill Education, 2007.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2018.

Woojun Kim, Jongeui Park, and Youngchul Sung. Communication in multi-agent reinforcement learning: Intention sharing. In *International Conference on Learning Representations*, 2020.

Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1774–1783, 2018.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems*, 30:6379–6390, 2017.

Ziyuan Ma, Yudong Luo, and Jia Pan. Learning selective communication for multi-agent path finding. *arXiv preprint arXiv:2109.05413*, 2021.

Murtaza Rangwala and Ryan Williams. Learning multi-agent communication through structured attentive reasoning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 10088–10098. Curran Associates, Inc., 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2186–2188, 2019.

Alvaro Serra-Gómez, Bruno Brito, Hai Zhu, Jen Jen Chung, and Javier Alonso-Mora. With whom to communicate: learning efficient communication for multi-robot collision avoidance. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11770–11776. IEEE, 2020.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Junjie Sheng, Xiangfeng Wang, Bo Jin, Junchi Yan, Wenhao Li, Tsung-Hui Chang, Jun Wang, and Hongyuan Zha. Learning structured communication for multi-agent reinforcement learning. *arXiv preprint arXiv:2002.04235*, 2020.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *International Conference on Learning Representations*, 2018.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29:2244–2252, 2016.

Chuangchuang Sun, Macheng Shen, and Jonathan P How. Scaling up multiagent reinforcement learning for robotic systems: Learn an adaptive sparse communication graph. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11755–11762. IEEE, 2020.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Varun Kumar Vijay, Hassam Sheikh, Somdeb Majumdar, and Mariano Phielipp. Minimizing communication while maximizing performance in multi-agent reinforcement learning. *arXiv preprint arXiv:2106.08482*, 2021.

Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich. Learning efficient multi-agent communication: An information bottleneck approach. In *International Conference on Machine Learning*, pp. 9908–9918. PMLR, 2020.

Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. In *International Conference on Learning Representations*, 2019.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Efficient communication in multi-agent reinforcement learning via variance based control. *Advances in Neural Information Processing Systems*, 32: 3235–3244, 2019.

Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Succinct and robust multi-agent communication with temporal message control. *Advances in Neural Information Processing Systems*, 33, 2020.

A Proof Eqa. 15

We set $N_k = \epsilon + \sum_{i=1}^{N} h_k(m_i)$ for brevity. Note that N_k represents the number of message variables that fall into the quantization interval $[(k-0.5)\Delta - 1, (k+0.5)\Delta - 1)$. To prove Eqa. 15, we present three lemmas:

Lemma A.1. *If* $m_i \in (u\Delta - 1, (u+1)\Delta - 1)$*, then*

$$\begin{aligned} &\nabla^p_{m_i} H(M) > 0,\ if\ N_u > N_{u+1} \\ &\nabla^p_{m_i} H(M) < 0,\ if\ N_u < N_{u+1} \\ &\nabla^p_{m_i} H(M) = 0,\ if\ N_u = N_{u+1} \end{aligned} \tag{18}$$

Remark This lemma shows that the for two adjacent quantization intervals, pseudo gradient descent method will make message variables move from the less popular one to the more popular one.

Proof.

$$\begin{aligned}\nabla^p_{m_i} H(M) &= \sum_{k=0}^{K} \nabla_{h_k} H(M) s_k(m_i) \\ &= -\frac{1}{N}\sum_{k=0}^{K}(1+\log\frac{N_k}{N}) s_k(m_i)\end{aligned} \tag{19}$$

Note that $\forall m_i \in (u\Delta - 1, (u+1)\Delta - 1)$, $s_u(m_i) = -1$, $s_{u+1}(m_i) = 1$ and $s_k(m_i) = 0\ \forall k \in \{0, 1, ..., K\} \setminus \{u, u+1\}$. Therefore, we get

$$\nabla^p_{m_i} H(M) = -\frac{1}{N} log \frac{N_{u+1}}{N_u} \tag{20}$$

Then we can lead to the equations in Lemma A.1 □

Lemma A.2. *If $N_u > N_{u+1}$ and $m_i \in (u\Delta - 1, (u+1)\Delta - 1)$ is updated to m'_i with $m'_i = m_i - \eta\nabla^p_{m_i} H(M), \eta \to 0$, this update leads to only two possible results:*

(1) $N'_k = N_k,\ \forall k \in \{0, 1, ..., K\}$

(2) $N'_u = N_u + 1$, $N'_{u+1} = N_{u+1} - 1$ and $N'_k = N_k,\ \forall k \in \{0, 1, ..., K\} \setminus \{u, u+1\}$

If $N_u < N_{u+1}$, similarly, this update leads to only two possible results:

(1) $N'_k = N_k,\ \forall k \in \{0, 1, ..., K\}$

(2) $N'_u = N_u - 1$, $N'_{u+1} = N_{u+1} + 1$ and $N'_k = N_k,\ \forall k \in \{0, 1, ..., K\} \setminus \{u, u+1\}$

Proof. We firstly focus on the first case (i.e. $N_u > N_{u+1}$). Since $\nabla^p_{m_i} H(M) > 0$,

$$m'_i = m_i - \eta\nabla^p_{m_i} H(M) \in (m_i - \eta \max(\nabla^p_{m_i} H(M)),\ m_i) \tag{21}$$

where

$$\max(\nabla^p_{m_i} H(M)) = \frac{1}{N}\log\frac{N_s+\epsilon}{\epsilon} \tag{22}$$

Since $\eta \to 0$, $\eta\max(\nabla^p_{m_i} H(M)) < 0.5\Delta$. Note that $m_i \in (u\Delta - 1, (u+1)\Delta - 1)$, so $m'_i \in ((u-0.5)\Delta - 1, (u+1)\Delta - 1)$. Consequently, there are only two possibilities for the values of m_i and m'_i:

(1) If $m_i \in ((u+0.5)\Delta - 1, (u+1)\Delta - 1)$ and $m'_i \in ((u-0.5)\Delta - 1, (u+0.5)\Delta - 1)$, then $N'_u = N_u + 1$, $N'_{u+1} = N_{u+1} - 1$ and $N'_k = N_k,\ \forall k \in \{0, 1, ..., K\} \setminus \{u, u+1\}$

(2) Otherwise, $N'_k = N_k,\ \forall k \in \{0, 1, ..., K\}$

The proof is similar in second case (i.e. $N_u < N_{u+1}$) □

Using $H(M)$ to denote $H(m_1, ..., m_i, ..., m_N)$ and $H(M')$ to denote $(m_1, ..., m'_i, ..., m_N)$, we propose our third lemma:

Lemma A.3. *$H(M') < H(M)$ if $N_u > N_{u+1}$, $N'_u = N_u + 1$, $N'_{u+1} = N_{u+1} - 1$ and $N'_k = N_k,\ \forall k \in \{0, 1, ..., K\} \setminus \{u, u+1\}$.*

$H(M') < H(M)$ if $N_u < N_{u+1}$, $N'_u = N_u - 1$, $N'_{u+1} = N_{u+1} + 1$ and $N'_k = N_k,\ \forall k \in \{0, 1, ..., K\} \setminus \{u, u+1\}$.

Proof. We firstly focus on the first case (i.e. $N_u > N_{u+1}$).

$$H(M') - H(M) = \frac{N_u}{N}\log\frac{N_u}{N} + \frac{N_{u+1}}{N}\log\frac{N_{u+1}}{N} - \frac{N_u+1}{N}\log\frac{N_u+1}{N} - \frac{N_{u+1}-1}{N}\log\frac{N_{u+1}-1}{N} \tag{23}$$

For brevity, we set $x_1 = \frac{N_u}{N}$, $x_2 = \frac{N_{u+1}}{N}$, $\delta = \frac{1}{N}$ and $f(x) = x \log x$. Then we get:

$$H(M') - H(M) = f(x_2) - f(x_2 - \delta) - (f(x_1 + \delta) - f(x_1)) \tag{24}$$

According to the mean value theorem, there exists $x_{1m} \in (x_1, x_1 + \delta)$ and $x_{2m} \in (x_2 - \delta, x_2)$ such that $f(x_2) - f(x_2 - \delta) = \delta f'(x_{2m})$ and $f(x_1 + \delta) - f(x_1) = \delta f'(x_{1m})$. Since $f''(x) > 0$ and $x_{2m} < x_2 < x_1 < x_{1m}$, $H(M') - H(M) < 0$. The proof is similar in the second case. □

From Lemma A.2 and Lemma A.3 we conclude that, if $H(M') \neq H(M)$, $H(M') < H(M)$. Therefore, we obtain $H(M') \leq H(M)$.

B Training details

We set the hidden layer size to 128 units in TARMAC and IC3NET. We use ReLU as the activation function in our fully-connected layers. We use REINFORCE to train our models, and the batch size is 6000. We perform ten weight updates in one epoch. The message size of IC3NET is 128. For TARMAC, the sizes of query vector, signature vector, and value vector are 16, 16, and 32. A message in TARMAC is composed of a query vector and a value vector therefore its length is 48. We train Original and ZeroComm models for T_{max} epochs. For DifEM and DisEM models, we train them without the regularizer for T_N epochs, then replace messages m with $\hat{m}$, add the regularizer and train for another $T_{max} - T_N$ epochs. The total training epochs for all models under one specific setting are the same, T_{max}, for ease of comparison. The weight of entropy regularizer for DisEM is α_p. Specific training details for each environment are present in Table 4.

Table 4: Specific training details for each environment

	Treasure Hunt	Predator Prey	Traffic Junction
optimizer	Adam	Adam	RMSProp
learning rate	0.0003	0.0003	0.001
T_{max}	200	200	1250
T_N	100	150	1000
α_p	0.2	0.05	0.05

C Communication simulations details

In this section, we build a basic digital communication system and test the performance of TARMAC-Original and TARMAC-DisEM in Predator Prey environment. The overall design of our communication system is demonstrated in Figure 6, and we introduce each module below.

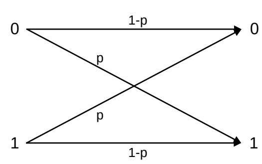


Figure 5: Binary symmetric channel.

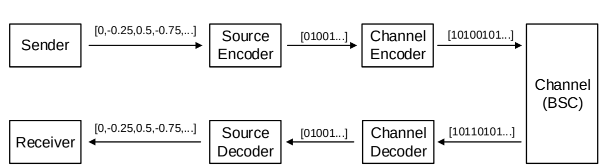


Figure 6: The overall design of the communication system.

C.1 Binary symmetric channel

A binary symmetric channel (BSC) with crossover probability p is a basic channel (Shannon, 1948) with binary input and binary output as shown in Figure 5. The input x and output y of BSC satisfies $p(y = x) = 1 - p$ and $p(y \neq x) = p$.

C.2 SOURCE CODING

The goal of source coding is to represent symbols output by the source with 0 and 1 (John Proakis, 2007). This step usually contains data compression, whose purpose is to minimize the expected length of codes. Besides, the entropy of the source indicates the limits of source coding (Shannon, 1948):

Theorem C.1. *(Source Coding Theorem) The expected length L of the optimal D-ary code for a random variable X satisfies the following inequalities:*

$$H_D(X) \leq L < H_D(X) + 1 \tag{25}$$

The messages of TARMAC-Original and TARMAC-DisEM agents are vectors of length 48: $\vec{m} = [m_0, m_1, ..., m_{47}]$ where each digit m_i has a certain distribution. In our implementation, we assume that all the digits follow the same discrete distribution for convenience and apply Huffman Coding (Huffman, 1952). We calculate the frequency of each value in the history message set and obtain a code table shown in Table 5. The expected lengths of TARMAC-DisEM codes are shorter than those of TARMAC-Original codes because messages of TARMAC-DisEM have lower entropy.

Table 5: Huffman Coding table, where **LEN** refers to the expected lengths of codes

			-1	**-0.75**	**-0.5**	**-0.25**	**0**	**0.25**	**0.5**	**0.75**	**1**	**len**
PP-A	*Original*	*Probability*	*0.006*	*0.015*	*0.055*	*0.241*	*0.398*	*0.208*	*0.056*	*0.015*	*0.006*	
		Code	*11010100*	*1101011*	*11011*	*10*	*0.000*	*111*	*1100*	*110100*	*11010101*	2.30
	DisEM	*Probability*	*0.000*	*0.000*	*0.001*	*0.082*	*0.873*	*0.043*	*0.001*	*0.001*	*0.000*	
		Code	*00000001*	*0000001*	*00001*	*01*	*1*	*001*	*0001*	*000001*	*00000000*	*1.32*
PP-B	*Original*	*Probability*	*0.005*	*0.015*	*0.036*	*0.18*	*0.395*	*0.262*	*0.059*	*0.033*	*0.016*	
		Code	*1101010*	*1101011*	*11001*	*111*	*0.000*	*10*	*11011*	*11000*	*110100*	2.27
	DisEM	*Probability*	*0.000*	*0.000*	*0.003*	*0.067*	*0.689*	*0.240*	*0.001*	*0.000*	*0.000*	
		Code	*00000000*	*000001*	*0001*	*001*	*1*	*01*	*00001*	*0000001*	*00000001*	*1.47*

C.3 CHANNEL CODING

The purpose of channel coding is to achieve reliable communication over a noisy channel by adding redundancy to the codes (John Proakis, 2007). In our simulation, we use convolutional codes (Elias, 1955). A convolutional encoder with code rate n/k transforms the input sequences of data rate n to encoded sequences of data rate k. Smaller n/k leads to more redundancy and therefore grants more robustness against noises. Compared with TARMAC-Original, TARMAC-DisEM has shorter source codes, thus leaving more space for channel coding. We use convolutional encoders of data rate $1/3$ for TARMAC-DisEM, and convolutional encoders of data rate $1/2$ for TARMAC-Original.

D DETAILS OF BASELINE MULTI-AGENT COMMUNICATION FRAMEWORKS

D.1 IC3NET

IC3NET is short for **Individualized Controlled Continuous Communication Model**, a framework put forward by Singh et al. (2018). It is an improved version of COMMNET (Sukhbaatar et al., 2016) with two modifications: (1) each agent is trained with individualized rewards; (2) the model can learn when to communicate with the help of the gate mechanism. Our implementation is slightly different from the original version and is stated below.

The j-th agent is individually controlled by a GRU, a gating network f^g, a policy network π, an observation encoder network e, a linear transformation matrix C, and a message generator f^e. At timestep t, the hidden state for the j-th agent is h_j^{t-1} and it receives o_j^t from the environment. Then the decision process is described as follows:

$$\begin{aligned}
m_j^t &= f^g(h_j^{t-1}) \cdot f^e(h_j^{t-1}) \\
c_j^t &= \frac{1}{J-1} C \sum_{j' \neq j} m_{j'}^t \\
h_j^t &= GRU(e(o_j^t) + c_j^t, h_j^{t-1}) \\
a_j^t &= \pi(h_j^t)
\end{aligned}$$

where J is the number of alive agents in the system. In addition, all agents' networks share the same parameters for faster convergence and are trained with REINFORCE (Williams, 1992).

D.2 TARMAC

TARMAC is short for Targeted Multi-Agent Communication (Das et al., 2019), a framework where agents utilize an attention mechanism to determine the weights of receiving messages. Our implementation for TARMAC is stated below.

The j-th agent is individually controlled by a GRU, a policy network π, an observation encoder network e, a query predictor f^q, and a message generator f^e. At timestep t, the hidden state for the j-th agent is h_j^{t-1} and it receives o_j^t from the environment. Besides, it generates a query vector $q_j^t = f^q(h_j^{t-1}) \in \mathbb{R}^{d_k}$ to determine the weights of receiving messages, and a value vector v_j^t as well as a signature k_j^t as messages: $m_j^t = [k_j^t, v_j^t] = f^e(h_j^{t-1})$. Additionally, the attention weights α_{ji} is calculated using a softmax operation, and the decision process is described as follows:

$$\begin{aligned}
q_j^t &= f^q(h_j^{t-1}) \\
m_j^t &= [k_j^t, v_j^t] = f^e(h_j^{t-1}) \\
\alpha_j &= softmax\left[\frac{(q_j^t)^T k_1^t}{\sqrt{d_k}} \dots \frac{(q_j^t)^T k_i^t}{\sqrt{d_k}} \dots \frac{(q_j^t)^T k_N^t}{\sqrt{d_k}}\right] \\
c_j^t &= \sum_{i=1}^{N} \alpha_{ji} v_i^t \\
h_j^t &= GRU(e(o_j^t) + c_j^t, h_j^{t-1}) \\
a_j^t &= \pi(h_j^t)
\end{aligned}$$

where α_{ji} is the i-th number of vector α_j. All agents' networks share the same parameters for faster convergence and are trained based on REINFORCE (Williams, 1992).