SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Robust cooperative multi-agent reinforcement learning via multi-view message certification

Lei YUAN[1,2], Tao JIANG[1], Lihe LI[1], Feng CHEN[1], Zongzhang ZHANG[1] & Yang YU[1,2*]

[1]*National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210000*, China;*
[2]*Polixir Technologies, Nanjing* 211106*, China*

Appendix A Product of a Finite Number of Gaussians

$$\begin{aligned}\mu_i &= \left(\frac{\mu_{i1}}{\sigma_{i1}^2}+\frac{\mu_{i2}}{\sigma_{i2}^2}+\cdots+\frac{\mu_{iN}}{\sigma_{iN}^2}\right)\sigma_i^2,\\ \frac{1}{\sigma_i^2} &= \frac{1}{\sigma_{i1}^2}+\frac{1}{\sigma_{i2}^2}+\cdots+\frac{1}{\sigma_{iN}^2}.\end{aligned} \tag{A1}$$

It can be proved by induction.

Proof.

We want to prove Eqn. A1 is true for all $N \geqslant 2$.

- Base case: Suppose $N = 2$ and $p_1(x) = \mathcal{N}(x|\mu_1, \sigma_1), p_2(x) = \mathcal{N}(x|\mu_2, \sigma_2)$, then

$$\begin{aligned}p_1(x)p_2(x) &= \frac{1}{\sqrt{2\pi}\sigma_1}\exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right)\cdot\frac{1}{\sqrt{2\pi}\sigma_2}\exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right)\\ &= \frac{1}{2\pi\sigma_1\sigma_2}\exp\left(-\left(\frac{(x-\mu_1)^2}{2\sigma_1^2}+\frac{(x-\mu_2)^2}{2\sigma_2^2}\right)\right)\\ &= \frac{1}{2\pi\sigma_1\sigma_2}\exp\left(-\frac{x^2-2\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}x+\frac{\mu_1^2\sigma_2^2+\mu_2^2\sigma_1^2}{\sigma_1^2+\sigma_2^2}}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}\right)\\ &= \frac{1}{2\pi\sigma_1\sigma_2}\exp\left(-\frac{\left(x-\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}-\frac{(\mu_1-\mu_2)^2}{2\sigma_1^2\sigma_2^2}\right)\\ &= \frac{\exp\left(-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}\right)}{\sqrt{2\pi\left(\sigma_1^2+\sigma_2^2\right)}}\cdot\frac{1}{\sqrt{2\pi}\frac{\sigma_1\sigma_2}{\sqrt{\sigma_1^2+\sigma_2^2}}}\exp\left(-\frac{\left(x-\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}\right)\\ &= A\cdot\frac{1}{\sqrt{2\pi}\frac{\sigma_1\sigma_2}{\sqrt{\sigma_1^2+\sigma_2^2}}}\exp\left(-\frac{\left(x-\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}\right).\end{aligned} \tag{A2}$$

Eqn. A2 can be seen as PDF of $\mathcal{N}(\mu, \sigma)$ times A where $\mu = (\frac{\mu_1}{\sigma_1^2}+\frac{\mu_2}{\sigma_2^2})\sigma^2$, $\frac{1}{\sigma^2} = \frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}$.

- Induction step: Suppose it is true when $N = n$, and the product distribution of n Gaussian experts has mean $\tilde{\mu} = (\frac{\mu_1}{\sigma_1^2}+\cdots+\frac{\mu_n}{\sigma_n^2})\tilde{\sigma}^2$ and variance $\frac{1}{\tilde{\sigma}^2} = \frac{1}{\sigma_1^2}+\cdots+\frac{1}{\sigma_n^2}$, then for $n+1$ Gaussian experts:

$$\begin{aligned}\frac{1}{\sigma^2} &= \frac{1}{\tilde{\sigma}^2}+\frac{1}{\sigma_{n+1}^2} = \frac{1}{\sigma_1^2}+\cdots+\frac{1}{\sigma_n^2}+\frac{1}{\sigma_{n+1}^2},\\ \mu &= \left(\frac{\tilde{\mu}}{\tilde{\sigma}^2}+\frac{\mu_{n+1}}{\sigma_{n+1}^2}\right)\sigma^2 = \left(\frac{\mu_1}{\sigma_1^2}+\cdots+\frac{\mu_n}{\sigma_n^2}+\frac{\mu_{n+1}}{\sigma_{n+1}^2}\right)\sigma^2.\end{aligned} \tag{A3}$$

- Eqn. A1 has been proved by the above derivation.

* Corresponding author (email: yuy@nju.edu.cn)

If we write $T_{ij} = (\sigma_{ij}^2)^{-1}$, then Eqn. A1 can be written as:

$$\begin{aligned} \mu_i &= \Big(\sum_{j=1}^{N} \mu_{ij} T_{ij}\Big)\Big(\sum_{j=1}^{N} T_{ij}\Big)^{-1}, \\ \sigma_i^2 &= \Big(\sum_{j=1}^{N} T_{ij}\Big)^{-1}, \end{aligned} \tag{A4}$$

and is exactly what we're trying to prove.

Algorithm A1 CroMAC

Input: env, ϵ, κ, C_MIN, C_MAX, α_1, α_2, α_3, T, T_r
Initialize: Randomly initialize $\boldsymbol{\theta}$, $\boldsymbol{\psi}_{\text{enc}}$, $\boldsymbol{\psi}_{\text{dec}}$, $\boldsymbol{\phi}_{\text{enc}}$, and initialize empty replay buffer D
1: $t = 0$
2: **while** $t < T$ **do**
3: Collect trajectory h from env with $\boldsymbol{\theta}(\mathbf{a}|\phi_{\text{enc}}(\mathbf{s}))$ and update replay buffer D
4: **for** $j = 1, 2, \cdots$ **do**
5: Sample a batch of Episodes from D
6: Update policy network:
7: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$
8: Update VAE ψ:
9: $\boldsymbol{\psi}_{\text{enc}} \leftarrow \boldsymbol{\psi}_{\text{enc}} - \alpha_1 \nabla_{\boldsymbol{\psi}_{\text{enc}}} \left(\mathcal{L}(\boldsymbol{\psi}) + \mathcal{L}(\boldsymbol{\theta})\right)$
10: $\boldsymbol{\psi}_{\text{dec}} \leftarrow \boldsymbol{\psi}_{\text{dec}} - \alpha_1 \nabla_{\boldsymbol{\psi}_{\text{dec}}} \mathcal{L}(\boldsymbol{\psi})$
11: Update MVAE ϕ:
12: $\boldsymbol{\phi}_{\text{enc}} \leftarrow \boldsymbol{\phi}_{\text{enc}} - \alpha_2 \nabla_{\boldsymbol{\phi}_{\text{enc}}} \mathcal{L}_(\boldsymbol{\phi})$
13: Clamp: $\boldsymbol{\phi}_{\text{enc}} \leftarrow clamp(\boldsymbol{\phi}_{\text{enc}}, C\_MIN, C\_MAX)$
14: **if** $t > T_r$ **then**
15: Update policy network:
16: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_3 \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{adv}}$
17: **end if**
18: Update the target network $\boldsymbol{\theta}^-$ at regular intervals
19: **end for**
20: Update t
21: **end while**

Appendix B Algorithm

The whole optimization process is shown in Alg. A1. Where lines 6 to 7 are used to train policy network with only TD-error such that it has nothing to do with robust training; lines 8 to 10 aim at encoding the state into a latent space, while lines 11 to 13 train the MVAE with only partial observation and the received messages for decentralized execution, where $clamp(input, min, max)$ means clamping all elements in $input$ into range $[min, max]$; we train the policy network to be robust with auxiliary loss from lines 14 to 17.

Appendix C Details about Baselines and Benchmarks

We compare CroMAC against different baselines and variants on diverse multi-agent benchmarks, and we introduce more details about these baselines here.

AME. Ablated Message Ensemble (AME) is a recently proposed certifiable defense method for multi-agent communication, which can guarantee agents' robustness when only part of communication messages suffer from perturbations. Specifically, AME makes a mild assumption that the attacker can only manipulate no more than half of the communication messages and trains a message-ablation policy that takes in a subset of messages from other agents and outputs a base action. In deployment, an ensemble policy is introduced by aggregating multiple base actions from multiple subsets of messages, achieving the robustness goal to some extent. The pseudocode can be seen in Alg.C1 and Alg.C2.

We introduce four types of testing environments as shown in Fig. **??** in our paper, including Hallway [**?**], Level-Based Foraging (LBF) [**?**], Traffic Junction (TJ) [**?**], and two maps named 1o2r_vs_4r and 1o10b_vs_1r requiring communication from StarCraft Multi-Agent Challenge (SMAC) [**?**]. In this part, we will describe the details of these used environments.

Hallway. We design two instances of the Hallway environment, where agent can see nothing except its own location. In the first instance, we apply three hallways with lengths of 4, 5, and 6, respectively. That means we let three agents a, b, c respectively initialized randomly at states a_1 to a_4, b_1 to b_5, and c_1 to c_6, and require them to arrive at state g simultaneously. In the second instance, 4 agents are distributed in four hallways with lengths of $3, 3, 4$, and 4. The action space is {*still*, *left*, *right*}. A reward of 1 will be given if all the agents arrive at the goal g simultaneously. However, if any agent does not reach the goal g at the same time, the game will stop immediately, and obtain 0 reward.

Level-Based Foraging (LBF). We use a variant version of the original environment, where only one agent can observe the map and others can see nothing. On this basis, we use two environment instances with different configurations, both of which are 8×8

Algorithm C1 AME in the training phase

Input: env, ablation size k
Initialize: Randomly initialize $\hat{\pi}_i$ for each agent i.

$t = 0$
while $t < T$ **do**
  for $i = 1$ to N **do**
    Receive a set of messages $\mathbf{m}_{:\to \mathrm{i}}$, and update local history τ_i
    Randomly sample a subset of messages $[\mathbf{m}_{:\to \mathrm{i}}]_{\mathrm{k}} \sim \mathrm{Uniform}(\mathcal{H}_{\mathrm{k}}(\mathbf{m}_{:\to \mathrm{i}}))$
    Take action based on τ_i and the message subset $[\mathbf{m}_{:\to \mathrm{i}}]_{\mathrm{k}}$, i.e., $a_i \leftarrow \hat{\pi}_i(\tau_i, [\mathbf{m}_{:\to \mathrm{i}}]_{\mathrm{k}})$
    Update replay buffer and policy $\hat{\pi}_i$
  end for
end while

Algorithm C2 AME in the testing phase

Input: env, ablation size k, trained message-ablation policy $\hat{\pi}_i$ for each agent i

for $i = 1$ to N **do**
  Receive a set of messages $\mathbf{m}_{:\to \mathrm{i}}$ with no more than $C < \frac{N}{2}$ malicious messages, and update local history τ_i
  Discrete action space:
    Take action $a \leftarrow \arg\max_{a \in \mathcal{A}} \sum_{[\mathbf{m}]_{\mathrm{k}} \in \mathcal{H}_{\mathrm{k}}(\mathbf{m})} \mathbb{I}[\hat{\pi}_i(\tau_i, [\mathbf{m}]_k) = a]$
  Continuous Action Space:
    Take action $a \leftarrow \mathrm{Median}\{\hat{\pi}_{\mathrm{i}}(\tau_{\mathrm{i}}, [\mathbf{m}]_{\mathrm{k}}) : [\mathbf{m}]_{\mathrm{k}} \in \mathcal{H}_{\mathrm{k}}(\mathbf{m})\}$
end for

grid world, 1 foods, and at least 3 agents are required to catch the food when they gather together next to the food concurrently. One of them contains 3 agents and they need to complete the task in 25 time-steps while the other contains 4 agents but only 15 time-steps are provided for them. The action set for each agent includes staying still and moving in one of four directions, only when at least 3 agents catch the food, they receive a constant reward $r = 1$.

Traffic Junction (TJ). The simulated traffic junction consists of several cars driving on the predefined road, and they need to avoid collisions and be as fast as possible. We use the *easy* version of the Traffic Junction environments as we mainly focus on the robustness of the algorithm. The *slow* version has an agent number limit to 5 and the max rate at which to add cars is 0.3 while the *fast* version has an agent number limit to 4 and the max rate at which to add cars is 0.4. In both of these two instances, the road dimension is 7, and the sight of the agent is limited to 0, which means each agent can only observe a 1×1 field of view around it. The action space is $\{gas, brake\}$, and each active car driving on the road needs to pay a time penalty $r_t = -0.01$ at every time-step. When a collision occurs, there will be a collision penalty $r_{\mathrm{collision}} = -10$.

StarCraft Multi-Agent Challenge (SMAC). We use two maps named 1o2r_vs_4r and 1o10b_vs_1r in SMAC, which are introduced in NDQ [**?**]. In 1o2r_vs_4r, an Overseer finds 4 Reapers, and the ally units, 2 Roaches, need to reach enemies and kill them. Similarly, 1o10b_vs_1r is a map full of cliffs, where an Overseer detects a Roach, and the randomly spawned ally units, 10 Banelings, are required to reach and kill the enemy. The action set consists of moving in one of four directions, attacking one of the enemies, and staying still, and agents receive sharing rewards when some enemy units' hit-point drop or win the battle.

Appendix D Implemention Detail and Hyperparameters

We train our CroMAC agents based on PYMARL with its default network structure and hyperparameters setting, except that different environments have different RNN hidden sizes. We use Adam optimizer with learning rate 0.0005 and other default hyperparameters. For the state encoder and decoder, we use an MLP with one hidden layer and ReLU activation. For the message encoder, we use the same structure as the state encoder with shared parameters. All the prior distributions in the experiment are set to standard normal distribution, i.e., $\mathcal{N}(\mathbf{0}, \mathbf{1})$. The other hyperparameters of our proposed CroMAC for different benchmarks are summarized in Tab. D1.

Fast Gradient Sign Method (FGSM) [**?**] is a popular white-box method of generating adversarial examples, for MA-Dec-POMDP-Com, agent i receives multiple messages m_{ij}^t from teammate $j \in \{1, \cdots, i-1, i+1, \cdots, N\}$ at time t, we can compute perturbations for each m_{ij}^t with individual Q-network θ_i:

$$\eta_{ij}^t = \boldsymbol{\epsilon} \cdot \mathrm{sign}(\nabla_{m_{ij}^t} J(\theta_i, m_{ij}^t, y)),$$

where y is the originally selected action and the perturbed example is:

$$\hat{m}_{ij}^t = m_{ij}^t + \eta_{ij}^t.$$

Projected Gradient Descent (PGD) [**?**] can be regarded as an advanced version of FGSM where we implement it by adding perturbations within budget $\frac{\boldsymbol{\epsilon}}{3}$ to original message 3 times.

Table D1 Hyperparameters in experiments.

Hyperparameter	Hallway: 4x5x6 Hallway: 3x3x4x4	LBF: 3p-1f LBF: 4p-1f	TJ: slow TJ: fast	1o10b_vs_1r 1o2r_vs_4r
RNN Hidden Dim	16	32	32	64
Z Dim	16	32	32	64
VAE Hidden Dim	32	64	64	128
α_1	0.1	0.01	0.01	0.01
α_2	0.001	0.001	0.001	0.01
α_3	0.3	0.3	0.3	0.3 0.1
κ	5 10	5 10	10	10
C_MAX	0.1 0.2	0.3	0.3 0.6	0.3 0.2
Robust Start Time T_r	0.7M	0.8M	1.0M	1.0M
ϵ of FGSM (1)	0.3 \	0.02 \	0.0003 \	0.0055 \
ϵ of FGSM (2)	0.4 \	0.25 \	0.0004 \	0.0065 \
ϵ of FGSM	0.5	0.03 0.05	0.0005 0.001	0.0075 0.015
ϵ of FGSM (3)	0.6 \	0.35 \	0.0006 \	0.00875 \
ϵ of FGSM (4)	0.7 \	0.4 \	0.0007 \	0.01 \