Continuous Communication with Factorized Policy Gradients in Multi-agent Deep Reinforcement Learning

Changxi Zhu
Utrecht University
Utrecht, Netherlands
c.zhu@uu.nl

Mehdi Dastani
Utrecht University
Utrecht, Netherlands
m.m.dastani@uu.nl

Shihan Wang
Utrecht University
Utrecht, Netherlands
s.wang2@uu.nl

ABSTRACT

In multi-agent deep reinforcement learning (MADRL), agents can learn to communicate to broaden their view and understanding of the environment and their teammates. Previous works on communication in MADRL mainly rely on centralized or independent value functions for the learning to communicate. In this paper, we propose to employ value decomposition methods, which provide centralized but factorized value functions for learning communication. The decomposed value functions utilize global information and help each agent differentiate how their decisions and associated communication contribute to the learning. Therefore, we propose a new policy gradient method called continuous communication with factorized policy gradients (CCFPG). The proposed method employs a dedicated factorized critic based on communication and an attention mechanism to aggregate messages. The results on continuous predator-prey and a new platform Multi-agent MuJoCo show that CCFPG can improve the performance or accelerate learning compared to other learning with communication methods.

KEYWORDS

Communication, Multi-agent Reinforcement Learning, Continuous Multi-agent Environments

1 INTRODUCTION

Multi-agent deep reinforcement learning (MADRL) holds considerable promise to help address a variety of multi-agent problems, such as autonomous driving [19], sensor networks [28], robotics [10], and game-playing [1, 20]. In many such settings, communication has shown great benefits to improve the performance of MADRL, where agents communicate their local observations, goals, or intentions to provide a better view of the environment and themselves for other agents [2, 4, 8, 23, 35]. Previous works on communication in multi-agent deep reinforcement learning (Comm-MADRL) mostly investigate learnable communication in domains with discrete action space. However, many tasks of interest, most notably physical control tasks, have continuous (real valued) and high dimensional action spaces. Whether communication can improve the learning of agents in continuous control, is underexplored.

In many multi-agent tasks, agents cooperate with each other to achieve a final goal. *Centralized training and decentralized execution* (CTDE) is one of the most popular training paradigms in MADRL and also Comm-MADRL [4, 5, 12, 17]. Many CTDE algorithms are based on actor-critic, where a value (or an action-value) function serves as a critic to guide learning optimal policies (i.e., actors) [5, 12]. In CTDE with actor-critic methods, the critic is usually centralized which gathers information from all agents and provides global feedback to decentralized actors (i.e., policies). However, the centralized critic can easily suffer from high dimensional input, which becomes even worse when agents view received messages as additional input. Moreover, training the fully centralized critic becomes impractical with the increasing number of agents, as more samples regarding agents' combinatorial behaviors and communication are required.

Previous works on Comm-MADRL do not fully address the two issues introduced above. Existing learning with communication methods are mainly developed for domain tasks with discrete action space, using either centralized or independent value functions to learn communication. At the same time, value decomposition methods, such as VDN [25] and QMIX [17] have gained performance improvements in many benchmark environments. The value decomposition methods decompose centralized value functions via a linear or non-linear combination of local Q-functions, which help agents differentiate their contributions to the learning. Inspired by this, we propose a new policy gradient method called continuous communication with factorized policy gradients (CCFPG) that enable continuous communication in MADRL tasks with continuous control. The CCFPG decomposes centralized value functions based on communication, which helps agents differentiate how their decisions and associated communication contribute to learning. In CCFPG, each agent employs a dedicated message encoder to generate real-valued messages. Then, the messages will be broadcast. In order to maintain a fixed size of the input, we employ an attention mechanism to aggregate received messages and agents will select actions based on aggregated messages. Each agent then simultaneously computes their local Q-functions, which will be combined through value decomposition methods like QMIX. By using deterministic actors and message encoders, gradients are backpropagated from factorized critics to associated communication. We evaluate CCFPG on one popular game Continuous Predator-Prey and the HalfCheetah task with continuous control in a new multi-agent platform Multi-Agent MuJoCo [16]. The results show that communication methods using value decomposition and the attention mechanism can significantly accelerate learning.

Our main contributions are:

- We propose a novel policy gradient method that employs centralized but factorized critics for the learning of communication in MADRL.
- By using deterministic actors and message encoders, the training is end-to-end, which efficiently backpropagates gradients from factorized critics to communication.
- We concentrate on the rarely studied multi-agent tasks with continuous control. The experiments on two different tasks

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 29-30, 2023, London, UK, https://alaworkshop2023.github.io/. 2023.

demonstrate that our proposed approach improves performance and/or accelerate learning.

2 RELATED WORK

Recent works on MARL have enabled agents to communicate and exchange messages during execution. Most research works on MARL with communication are built based on multi-agent tasks with continuous state and discrete action spaces. Two seminal works, DIAL [4] and CommNet [24] utilize the training of neural networks to allow backpropagation through the model of communication while having discrete action spaces in the design. Many communication methods are tested on the open-source platform StarCraft [18, 26], which provides a series of maps with different difficulty levels for continuous states and a large range of discrete actions. Due to the property of the underlying platform, those communication methods do not take policies on continuous actions into account [29, 30, 32–34]. Other communication methods are evaluated based on grid world games, such as Predator-prey and Traffic Junction, with discrete state and action spaces [2, 3, 7, 9, 14, 21]. To the best of our knowledge, there are only two research works, ATOC [8] and G2ANet [11], which are evaluated with a domain task with continuous state and action spaces. In ATOC, agents encode intended actions and communicate with nearby agents. In G2ANet, agents learn not only whether to communicate with other agents but also how to aggregate messages by weightings. Since both existing works target on a simple particle environment, it is essential to develop new learning with communication methods for a more rigorous comparison in complex environments to show generalization ability and scalability. This may require more efficient and effective usage of communication.

Centralized training and decentralized execution (CTDE) becomes popular in recent research works in multi-agent deep reinforcement learning and the field of communication. Most research works on MARL with communication utilize a centralized value (or action-value) function to guide the learning of decentralized policies [2, 3, 9, 14, 29]. However, agents can also learn a value decomposition from the shared reward signal into the individual component value functions [17, 22, 25]. Communicating agents can benefit from decomposition because individual value functions tell each agent how their behaviours and associated communication contribute to the overall learning. There is no previous work to employ factorized value functions for the learning of decentralized policies with the presence of communication among agents. Popular value decomposition methods are VDN [25], QMIX [17], and QTRAN [22]. In VDN, the value decomposition is linear and the state information is ignored during training. QMIX presents an improvement by conditioning a hypernetwork on the global state for a non-linear mixing of the individual action-values, but it is still limited by the monotonicity constraint of its mixing weights. QTRAN attempts to address these limitations with a provably more general value factorization, but it imposes computationally intractable constraints that can lead to poor empirical performance. These factorization methods are value-based, which acquire policies from learned value functions. Agents can also utilize factorized value functions to facilitate the learning of decentralized policies. DOP [31] and Facmac [16] are two policy gradient methods to employ centralized but factorized Q-functions to train stochastic and deterministic policies. DOP employs two replay buffers, an on-policy buffer and an off-policy buffer to efficiently utilize samples and reduce the variance of factorized Q-functions. Facmac is a fully differentiable method where deterministic policies receive gradient backpropagation from factorized Q-functions. Our proposed method extends Facmac to include communication and makes a few adaptions to efficiently utilize communication and accelerate learning.

3 PRELIMINARIES

We consider a fully cooperative multi-agent task in which a team of agents interacts within the same environment to achieve some common goals. The task can be modeled as a decentralized partially observable Markov decision process (Dec-POMDP) [15]. A Dec-POMDP is defined by a tuple $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{A}_i\}, \mathcal{P}, \{\mathcal{O}_i\}, \{\mathcal{R}_i\}, \gamma \rangle$, where $\mathcal{I}$ is a set of (finite) agents indexed as $\{1, ..., N\}$, $\mathcal{S}$ is a set of environment states, $\mathcal{A}_i$ is a set of actions available to agent i, and $\mathcal{O}_i$ is a set of observations of agent i. We can denote a joint action space as $\boldsymbol{\mathcal{A}} = \times_{i \in \mathcal{I}} \mathcal{A}_i$ and a joint observation space as $\boldsymbol{\mathcal{O}} = \times_{i \in \mathcal{I}} \mathcal{O}_i$. In the current environment state $s \in \mathcal{S}$, the joint action $\boldsymbol{a} = \langle a_1, ..., a_N \rangle$ of agents, where $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$, causes the environment to move to the next state $s' \in \mathcal{S}$ and to emit a new joint observation $\boldsymbol{o'} = \langle o'_1, ..., o'_N \rangle$ for all agents, where $\boldsymbol{o'} \in \boldsymbol{\mathcal{O}}$, according to the transition and observation probabilities $\mathcal{P}(s', \boldsymbol{o'} \mid s, \boldsymbol{a})$. Each agent then receives an immediate reward according to their own reward functions $\mathcal{R}_i : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \rightarrow \mathbb{R}$. Similar to the joint action and observation, we denote $\boldsymbol{r} = \langle r_1, ..., r_N \rangle$ as a joint reward. In Dec-POMDP, agents' reward functions are the same, i.e., they have identical goals, then $r_1 = r_2 = ... = r_N$ holds for every time step. The rewards are discounted by the discount factor γ. The joint policy of agents induces a joint action-value function: $Q^{joint}(s, \boldsymbol{a}) = \mathbb{E}_D[\sum_{t=0}^{\infty} \gamma^t r_t | s, \boldsymbol{a}]$. Whenever the true state s is not accessible, we use the joint trajectory $\boldsymbol{\tau} = \{\tau_1, ..., \tau_N\}$ instead, where $\tau_i = (o_0^i, a_0^i, ..., o_t^i)$ is the observation-action history up to the current time step t.

Value Decompositions. Value decomposition methods, such as VDN [25] and QMIX [17], aim to learn centralized but factorized Q-functions. VDN factorizes Q^{joint} into a sum of the per-agent local Q-functions: $Q^{joint}(s, \boldsymbol{a}) = \sum_{i=1}^{N} Q^i(s, a_i)$, whereas QMIX combines the local Q-functions of every agent via a continuous monotonic function that is state-dependent: $f_s(Q^1(\tau_1, a_1), ..., Q^N(\tau_N, a_N)) = Q^{joint}(\boldsymbol{\tau}, \boldsymbol{a})$, where $\frac{\partial f_s}{\partial Q^i} \geq 0, \forall i \in \mathcal{I}$. A common expression of QMIX is as follows,

$$f_s(Q^1(\tau_1, a_1), ..., Q^N(\tau_N, a_N)) = W_2 \times ELU(W_1 \times \boldsymbol{Q}(\boldsymbol{\tau}, \boldsymbol{a}) + b_1) + b_2$$

where W_1 and W_1 are weights produced by separate hypernetworks, $ELU(\cdot)$ is the ELU function, $\boldsymbol{Q}(\boldsymbol{\tau}, \boldsymbol{a}) = (Q^1(\tau_1, a_1), ..., Q^N(\tau_N, a_N))$, b_1 and b_2 are bias. By adopting neural networks as Q-functions, QMIX is trained in a DQN way. The squared temporal difference error $\mathcal{L}$ is minimized by using a minibatch of b samples from a replay buffer:

$$\mathcal{L} = \frac{1}{b} \sum_{k=1}^{b} (Q^{joint}(\boldsymbol{\tau}, \boldsymbol{a}; \boldsymbol{\lambda}) - y_k) \tag{1}$$

where $\boldsymbol{\lambda}$ is the parameters, $y_k = r + \gamma \max_{\boldsymbol{a}'} Q^{joint}(\boldsymbol{\tau}', \boldsymbol{a}'; \boldsymbol{\lambda}^-)$ is the target values and $\boldsymbol{\lambda}^-$ is the target parameters. The target parameters $\boldsymbol{\lambda}^-$ will be periodically synchronized with the parameters $\boldsymbol{\lambda}$: $\boldsymbol{\lambda}^- = (1-\tau) * \boldsymbol{\lambda}^- + \tau * \boldsymbol{\lambda}$, where $\tau \in [0, 1]$ is a hyperparameter.

4 METHOD

Previous works on MARL with communication either consider a centralized Q-function or decentralized Q-functions. Our proposed method employs a centralized but factorized Q-function, which enjoys the benefit of credit assignment to each agent and their communication. In order to derive the factorized policy gradients with communication step-by-step, we start with presenting the centralized policy gradients and combine it with communication. Based on the centralized policy gradients with communication, we further derive the proposed centralized but factorized policy gradients with communication (i.e., CCFPG). Later, we illustrate the forward pass of CCFPG in Figure 1 and essential updates of CCFPG in Algorithm 1.

Centralized Policy Gradients with Communication. Centralized Policy Gradients learn a joint policy with a centralized critic, such as the joint Q-function $Q^{joint}(\boldsymbol{\tau}, \boldsymbol{a})$ or the joint value function $V^{joint}(\boldsymbol{\tau})$, where $\boldsymbol{\tau}$ and $\boldsymbol{a}$ are the joint trajectory and the joint action of N agents. In this paper, we consider the joint Q-function to facilitate the factorization among actions. Then, a big Q-function is learned through Q-learning and enables gradient backpropagation for the joint action policy $\boldsymbol{\mu}(\boldsymbol{\tau}; \boldsymbol{\theta^{\mu}})$, parameterized with $\boldsymbol{\theta^{\mu}} = \{\theta_1^{\mu}, ..., \theta_N^{\mu}\}$. The centralized policy gradients with deterministic policies are therefore defined as follows,

$$g = \mathbb{E}_{\mathcal{D}}[\nabla_{\boldsymbol{\theta^{\mu}}} \boldsymbol{\mu}(\boldsymbol{\tau}; \boldsymbol{\theta^{\mu}}) \nabla_{\boldsymbol{a}} Q^{joint}(\boldsymbol{\tau}, \boldsymbol{a})|_{\boldsymbol{a}=\boldsymbol{\mu}(\boldsymbol{\tau}; \boldsymbol{\theta^{\mu}})}]$$

where D is the replay buffer. The gradient g is used to update the joint policy so as to maximize the corresponding Q-values. In CTDE, agents aim at learning decentralized policies. With communication, they can share information among individual policies to promote learning. We define a deterministic message encoder for each agent i as $f(\tau_i; \theta_i^{enc})$, parameterized with θ_i^{enc} (abbreviated as $f_i^{enc}(\tau_i)$). The message encoder produces real-valued messages $m^i = f_i^{enc}(\tau_i)$ based on agent i's own trajectory, where $m^i \in \mathcal{R}^D$ and D is the degree of freedom. Then we can denote the messages sent from other agents (except agent i) as $\boldsymbol{m}^{-i} = \{..., m^{i-1}, m^{i+1}, ...\}$. Each agent will select an action based on received messages: $a_i = \pi_i(\tau_i, \boldsymbol{m}^{-i}; \theta^{\pi_i})$, parameterized with θ^{π_i}. Note that we omit the parameters θ^{π_i} in policy π_i for the sake of convenience. Therefore, in CTDE, the centralized policy gradients with communication for each agent i is defined as follows,

$$g_i = \mathbb{E}_{\mathcal{D}}[\nabla_{\theta^{\pi_i}} \pi_i(\tau_i, \boldsymbol{m}^{-i}) \nabla_{a_i} Q^{joint}(\boldsymbol{\tau}, \boldsymbol{a})|_{a_i=\pi_i(\tau_i, \boldsymbol{m}^{-i})}]$$

The gradient g_i is used to update the policy of agent i. Compare to the centralized policy gradients without communication, each agent individually performs the above gradient update and assumes other agents to be fixed. At the same time, there is a dedicated communication part to coordinate the action selection of agents before making decisions in the environment. The centralized critic is known to be unbiased but has high variance when performing policy gradients [13], which can get worse in high stochasticity environments. We propose to factorize the centralized critic to neglect the effect of other agents' behaviours when updating each policy and associated communication, therefore, achieving more stable policy updates.

Continuous Communication with Factorized Policy Gradients. In the following section, we present a new policy gradient method called Continuous Communication with Factorized Policy Gradients (CCFPG) that aims to learn deterministic policies and communication in fully cooperative environments. Our key motivation is that the action policies and associated communication of agents can be factorized to differentiate their contribution to the overall learning. We propose to decompose the joint Q-function and therefore decentralized action policies and communication can benefit from gradient backpropagation from local Q-functions. In practice, we follow QMIX for the value decomposition in the critics. Instead of changing the Q-function unilaterally as in the centralized policy gradients, we allow all agents to change the Q-functions simultaneously to optimize the entire joint action space based on communication. Then, a deterministic action policy takes as input individual observations and continuous messages from other agents to select (continuous) actions. As the input messages from other agents can be large when the number of agents and the dimension of messages grows, we further utilize an attention mechanism to aggregate received messages. The attention mechanism provides a weighted combination of messages, which can help receiver agents differentiate the individual contribution of each received message.

We illustrate CCFPG in Figure 1. As shown in the figure, in a multi-agent system with N agents, individual trajectories τ_i of each agent i are first encoded as messages (denoted as m^i). Then, all agents broadcast their messages to other agents. Each agent will aggregate received messages and compute an action a_i based on aggregated messages (denoted as m_i^{agg}) and its own observed trajectories. With the chosen actions, agents combine their local Q-values by a mixing network, followed by QMIX. Finally, the combined (joint) Q-values will be trained in a DQN way, as defined in Equation 1.

We derive the policy gradient for each agent i according to factorized Q-functions as in the following equations. Note that we first do not consider the presence of an aggregator and later we show how to utilize the attention mechanism to aggregate messages. Then, agents update their policies according to the gradient g as follows,

$$g = \mathbb{E}_{\mathcal{D}}\left[\nabla_{\boldsymbol{\theta^{\pi}}} \boldsymbol{\pi} \nabla_{\boldsymbol{\pi}} Q^{joint}(\boldsymbol{\tau}, \pi_1(\tau_1, \boldsymbol{m}^{-1}), ..., \pi_N(\tau_N, \boldsymbol{m}^{-N}))\right] \quad (2)$$

where $\boldsymbol{\pi} = \{\pi_1(\tau_1, \boldsymbol{m}^{-1}; \theta^{\pi_1}), ..., \pi_N(\tau_N, \boldsymbol{m}^{-N}; \theta^{\pi_N})\}$ and $\boldsymbol{\theta^{\pi}} = \{\theta^{\pi_1}, ..., \theta^{\pi_N}\}$. By factorizing the joint Q-function, we can obtain the gradient g_i for updating each agent i's policy as follows,

$$\begin{aligned} g_i &= \mathbb{E}_{\mathcal{D}}\left[\nabla_{\theta^{\pi_i}} \pi_i(\tau_i, \boldsymbol{m}^{-i}) \nabla_{\pi_i} Q^{joint}(\boldsymbol{\tau}, ..., \pi_i(\tau_i, \boldsymbol{m}^{-i}), ...)\right] \\ &= \mathbb{E}_{\mathcal{D}}\left[\nabla_{\theta^{\pi_i}} \pi_i(\tau_i, \boldsymbol{m}^{-i}) \nabla_{\pi_i} f_s\left(..., Q^i(\tau_i, \pi_i(\tau_i, \boldsymbol{m}^{-i})), ...\right)\right] \\ &= \mathbb{E}_{\mathcal{D}}\left[\nabla_{\theta^{\pi_i}} \pi_i(\tau_i, \boldsymbol{m}^{-i}) \nabla_{\pi_i} Q^i(\tau_i, \pi_i(\tau_i, \boldsymbol{m}^{-i})) \nabla_{Q^i} f_s\right] \end{aligned} \quad (3)$$

where Q^{joint} and f_s take all agent's actions and Q-values into account. In line 2, we replace the joint Q-function with factorized Q-functions by QMIX based on communication. Line 3 is according to the Chain rule. In line 3, $\nabla_{Q^i} f_s$ is always non-negative by the definition of QMIX, which can scale the gradient with respect to the local

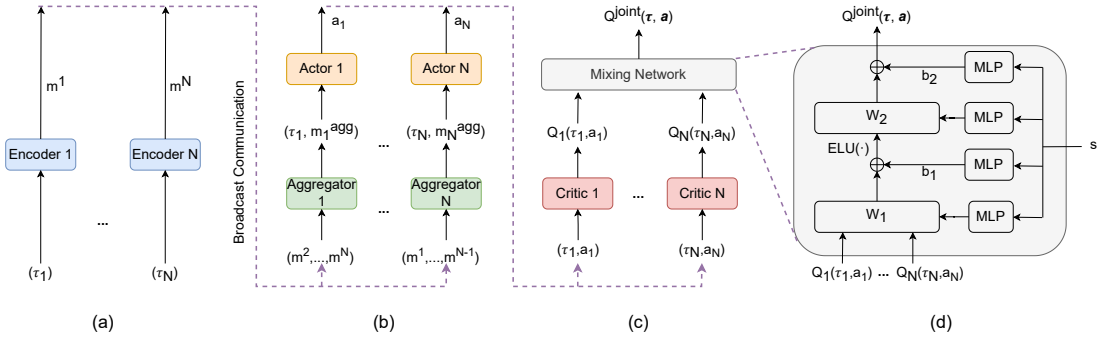

Figure 1: Overview of CCFPG. (a) Architecture for the message encoders. (b) Architecture for the decentralized policy networks. (c) Architecture for the centralized but factorized critic. (d) Architecture for the centralized mixing network.

Q-function Q^i. Then, each agent simultaneously uses the above gradient to update their policies. Similarly, for other agents' message encoders, we can continue to perform gradient backpropagation to update their parameters $\boldsymbol{\theta}^{enc}_{-i} = \{\theta^{enc}_1, ..., \theta^{enc}_{i-1}, \theta^{enc}_{i+1}, ..., \theta^{enc}_N\}$. Therefore, we define the gradients of all agents' message encoders except for agent i as follows,

$$g^{enc}_{-i} = \mathbb{E}_{\mathcal{D}}[\nabla_{\boldsymbol{\theta}^{enc}_{-i}} \boldsymbol{f^{enc}_{-i}} \nabla_{\boldsymbol{f^{enc}_{-i}}} \pi_i(\tau_i, \boldsymbol{f^{enc}_{-i}}) \nabla_{\pi_i} Q^{joint}(\boldsymbol{\tau}, ..., \pi_i(\tau_i, \boldsymbol{f^{enc}_{-i}}), ...)] \quad (4)$$

where $\boldsymbol{f^{enc}_{-i}} = \{f^{enc}_1(\tau_1), ..., f^{enc}_{i-1}(\tau_{i-1}), f^{enc}_{i+1}(\tau_{i+1}), ..., f^{enc}_N(\tau_N)\}$, including all senders' messages. The gradient g^{enc}_{-i} is used to update the message encoders of all the agents except for agent i. Due to the value decomposition, the gradient is also scaled by the non-negative term $\nabla_{Q^i} f_s$. The gradient update will be performed for every receiver agent.

Attentional Continuous Communication. The action policy requires messages from all the other agents, which may introduce high dimensional input with the growing number of agents. Moreover, during learning, some agents may be more important than the others, for example, agents who are closer to the goal. Therefore, agents should impose importance on received messages and maintain a fixed size of the input of receivers' action policies. In CCFPG, we utilize an attention mechanism to combine received messages by learnable weights. We adopt the form of Dot-Product Attention [27]. Then, for agent i, message aggregation function f^{agg}_i is defined as follows,

$$f^{agg}_i(\boldsymbol{m}^{-i}) = W_P \left(\sum_{j \neq i}^{N} \alpha_{ij} ReLU(W_V m^j) \right)$$

where $W_V \in \mathbb{R}^{D' \times D}$ is a (learnable) weight matrix to project messages to a value matrix, $ReLU(\cdot)$ is the ReLU function, $\alpha_{ij} \in \mathbb{R}^{D'}$ is a (learnable) weighting vector to combine values from the value matrix, and $W_P \in \mathbb{R}^{D \times D'}$ is used to project from the value matrix back to message space, where D' refers to the dimension of values and weighting vector and D refers to message cardinality. The weighting vector α_{ij} is defined as follows,

$$\alpha_{ij} = softmax(W_Q m^i \otimes W_K m^j)$$

where $softmax(\cdot)$ is the softmax function, $W_Q \in \mathbb{R}^{D' \times D}$ is the weight matrix to project messages from agent i to a query matrix, $W_K \in \mathbb{R}^{D' \times D}$ is the weight matrix to project messages from other agent j to a key matrix, and $\otimes$ is an element-wise multiplication operator. Therefore, we denote the set of parameters of aggregation function f^{agg}_i as $\theta^{agg}_i = \{W_Q, W_K, W_V, W_P\}$, which is learned through backpropagation. By aggregating messages, for any other agent j who sent messages to agent i, we can rewrite the gradient update of sender agent j's message encoder as follows,

$$g^{enc}_{i,j} = \mathbb{E}_{\mathcal{D}}[\nabla_{\theta^{enc}_j} f^{enc}_j(\tau_j) \nabla_{f^{enc}_j} f^{agg}_i(\boldsymbol{f^{enc}_{-i}}) \nabla_{f^{agg}_i} \pi_i(\tau_i, f^{agg}_i(\boldsymbol{f^{enc}_{-i}})) \nabla_{\pi_i} Q^{joint}(\boldsymbol{\tau}, ..., \pi_i(\tau_i, f^{agg}_i(\boldsymbol{f^{enc}_{-i}})), ...)] \quad (5)$$

The gradient $g^{enc}_{i,j}$ is computed for each pair of receiver-sender agents (i, j), where $i \neq j$. Compare to Equation 4, Equation 5 utilizes the attention mechanism as an aggregation function, which exerts importance on each received message and scales the gradients through weighting vector α_{ij}. Therefore, gradients in Equation 5 are firstly scaled by the term $\nabla_{Q^i} f_s$ from the value decomposition and then the attention mechanism, leading to more fine-grained feedback for each pair of sender and receiver agents. Because of the usage of the aggregation function, the policy in Equation 3 will take aggregated messages into account instead of all the received messages. Equations 3 and 5 constitute the final update rules of action policy and communication, which will update the Encoder and Actor in Figure 1.

Algorithm 1 describes the learning procedures of our proposed method CCFPG. First, agents interact with the environment and obtain experience. They use random noises to allow exploration for message encoders and action policies, i.e., generated by random processes $\mathcal{N}^a$ and $\mathcal{N}^m$. Whenever training is enabled, each agent will sample a minibatch from the replay buffer and update the critic's parameters $\boldsymbol{\lambda}$ with learning rate α_{critic}. Agents then perform gradient updates according to Equations 3 and 5 simultaneously. The parameters of the actors and the message encoder (i.e., $\boldsymbol{\theta^{enc}}$ and $\boldsymbol{\theta^{\pi}}$) are updated sequentially (with learning rates α_{actor} and α_{comm}). Note that the sampled gradients of actors are calculated based on aggregated messages and therefore we have to record aggregated messages in the replay buffer. In practice, agents will use LSTM neural networks [6] to learn a representation of trajectories

Algorithm 1 Continuous Communication with Factorized Policy Gradients

for episode = 1 to M **do**
  Initialize random processes $\mathcal{N}^a$ and $\mathcal{N}^m$ for action exploration and message exploration
  Receive initial observations $\boldsymbol{o}$
  for time step t = 1 to max-episode-length **do**
    for each agent i, select message $\boldsymbol{m}^i = f_{\theta^{enc}}(\tau_i) + \mathcal{N}_t^m$
    for each agent i, aggregate messages $\boldsymbol{m}_i^{agg} = f_{\theta_i^{agg}}(\boldsymbol{m}^{-i})$
    for each agent i, select action $a_i = \pi_i(\tau_i, m_i^{agg}; \theta^{\pi_i}) + \mathcal{N}_t^a$
    Execute action $\boldsymbol{a}$, observe reward r and new observations $\boldsymbol{o}'$
    Store $(\boldsymbol{o}, \boldsymbol{m^{agg}}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{o}')$ in replay buffer D
    Move to the next state
  end for
  Sample a random minibatch of b samples from D
  Set $y^k = r^k + \gamma \max_{\boldsymbol{a}'} Q^{joint}(\boldsymbol{\tau}'^k, \boldsymbol{a}'^k; \boldsymbol{\lambda}^-)$ for the k-th sample
  Calculate sampled gradients by the loss in Equation 1 w.r.t. the critic's parameters $\boldsymbol{\lambda}$
  Update $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \alpha_{critic} \nabla_{\boldsymbol{\lambda}} \mathcal{L}$
  for agent i = 1 to N **do**
    Calculate sampled gradient $\hat{g}_i$ by Equation 3 w.r.t. the actor's parameters θ^{π_i}.
  end for
  for each agent i: update $\theta^{\pi_i} \leftarrow \theta^{\pi_i} + \alpha_{actor} \hat{g}_i$
  for each pair of sender and receiver (i, j) **do**
    Calculate sampled gradient $\hat{g}_{i,j}^{enc}$ by Equation 5 w.r.t. each encoder's parameters θ_j^{enc}.
  end for
  Update the attention mechanism along with encoders.
  for each receiver agent i **do**
    for each sender agent j: update $\theta_j^{enc} \leftarrow \theta_j^{enc} + \alpha_{comm} \hat{g}_{i,j}^{enc}$
  end for
  Update target critic's parameters: $\boldsymbol{\lambda}^- \leftarrow (1 - \tau) * \boldsymbol{\lambda}^- + \tau * \boldsymbol{\lambda}$
end for

which only require recording observations and actions at every time step.

5 EXPERIMENTS

We present our experimental results on two practical domains, Continuous Predator-Prey and Multi-Agent MuJoCo (MAMuJoCo) [16]. Continuous Predator-Prey can be customized with different sizes of environments with various numbers of learning agents. MAMuJoCo has a variety of tasks with different agent partitionings. Both platforms are popular in multi-agent reinforcement learning with continuous state and action spaces. In order to show the advantage of our proposed methods, we compare with several MARL baseline models as follows,

- **Facmac [16]**. This is a fully factorized method which factorizes both critic and actor. There is no communication among agents.
- **G2ANet [11]**. G2ANet is a baseline method with communication that is tested on continuous environments (with continuous actions). The critics of agents are communicated through a graph neural network, where agents have to decide when to communicate and employ an attention mechanism to aggregate messages.
- **FCMNet [32]**. FCMNet is one of the state-of-the-art communication methods, which is developed for discrete environments. We adapt this method to our continuous environments by using a Gaussian policy.
- **CCFPG**. This is our proposed method which is equipped with an attention mechanism. We also compare the method **CCFPG w/o attention**, where the attention mechanism is removed, to show the strength of aggregating messages through a weighted combination [1].

We evaluate the performance of each method using the following procedure: for each run of a method, we pause training every fixed number of timesteps (2000 timesteps for Continuous Predator-Prey and 4000 timesteps for MAMuJoCo) and run a fixed number of evaluation episodes (10 independent episodes for Continuous Predator-Prey and MAMuJoCo) with each agent performing action selection greedily in a decentralized fashion. On both Continuous Predator-Prey and MAMuJoCo, the mean value of these episode returns is used to evaluate the performance of the learned policies. All results are averaged over 9 independent runs. The independent runs use the same hyperparameter configuration, only varying the random seed. During training and testing, we restrict each episode to have a length of 25 time steps for Continuous Predator-Prey and 1000 time steps for MAMuJoCo. In total, we run 3 million training steps for Continuous Predator-Prey and 1.5 million training steps for MAMuJoCo.

Facmac, CCFPG, and CCFPG w/o attention share the same critic, which has 64 hidden units in Continuous Predator-Prey and 400 hidden units in MAMuJoCo. For all tasks, a mixing network with 64 hidden units is used for the value decomposition. Facmac, CCFPG, and CCFPG w/o attention have the same number of hidden units in the actor while CCFPG and CCFPG w/o attention have an additional input for messages. We use an attention mechanism with 32 hidden units for CCFPG, i.e., $D' = 32$. For the dimension of messages, we choose $D = 2$ in Predator-Prey with 6 and 9 agents, $D = 6$ for Predator-Prey with 12 agents, and $D = 3$ in MAMuJoCo tasks, which are fine-tuned. Gaussian noises with mean $\mu = 0$ and standard deviation $\sigma = 0.1$ are added to actions and messages during execution to allow exploration. In G2ANet, we use a similar size of neural networks as in Facmac. Facmac, G2ANet, CCFPG, and CCFPG w/o attention are trained by the Adam optimizer. The learning rate and number of updates per epoch are chosen either empirically or fine-tuned. Then, we choose a learning rate of 0.001 for the learning of critics, actors, and communication. FCMNet is trained by using the parameters introduced in the original paper. For the soft target network updates we use $\tau = 0.001$. During learning, a minibatch of 32 for Predator-Prey and 100 for MAMuJoCo are used to train the parameters. We summarize critical hyperparameters of our proposed method CCFPG in Table 1. CCFPG w/o attention removes the attention mechanism from CCFPG while keeping other networks and parameters unchanged.

5.1 Continuous Predator-Prey

Continuous Predator-Prey is a variant of the classic predator-prey game. We employ the same implementation developed by Peng et al. [16]. In the environment, several slower cooperating circular

[1]The code is available at https://github.com/chauncyzhu/CADDPG

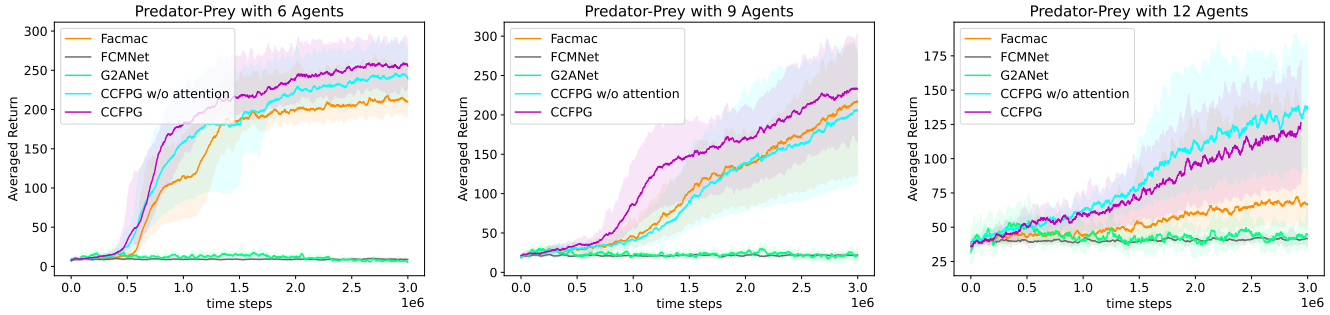

Figure 2: Averaged return of algorithms in Predator-Prey with 6 (left), 9 (medium), and 12 (right) agents.

Table 1: Important hyperparameters of CCFPG. $NN(x)$ indicates the size (i.e., x) of hidden layers in the corresponding neural networks (NN).

Hyperparameters	Predator-Prey	MAMujoco
Actor	$RNN(64)$	$MLP(400)$
Critic	$MLP(64)$	$MLP(400)$
Mixing	$HyperNet(64)$	$HyperNet(64)$
Encoder	$MLP(32)$	$MLP(100)$
Aggregator	$Attention(32)$	$Attention(32)$
α_{actor}	0.01	0.001
α_{critic}	0.01	0.001
α_{comm}	0.01	0.001
γ	0.85	0.99
τ	0.001	0.001
Exploration Noise	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 0.1)$
Batch Size	32	100
Buffer Size	5000	1000000

agents, each with continuous movement action spaces, must catch a faster circular prey on a randomly generated two-dimensional toroidal plane with large landmarks blocking the way. An example with 6 (learning) predators is shown in Figure 3. The prey's policy is a hard-coded heuristic, which moves the prey to the position with the largest distance to the closest predator. The predators are reinforcement learning agents, learning and/or communicating in order to catch the prey to get higher rewards. If one of the cooperative agents collides with the prey, a team reward of +10 is emitted; otherwise, no reward is given. Each agent has a view radius, which restricts the agents from receiving information about other entities (including landmarks, other predators, and the prey) that are out of range.

We explore Continuous Predator-Prey with a varying number of agents to show the generalization ability of our proposed methods. The results are shown in Figure 2[2]. As we can see, in all settings, our proposed methods surpass the fully factorized method Facmac, which shows the benefit of adding a communication mechanism

[2]The shadow area in a figure is the min-max range of performance across 9 runs for a particular method. We only show half of the range to see a clear difference between methods.

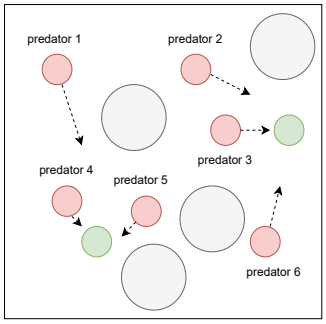

Figure 3: Continuous Predator-Prey. Top-down view of toroidal plane, with predators (red), prey (green), and obstacles (grey).

to accelerate learning. As the number of agents grows, our methods can achieve more competitive performance than Facmac. In all cases, our proposed methods are significantly better than the two communication baselines, G2ANet and FCMNet. This indicates the benefit of factorization in MADRL with communication. When the number of learning predators is 6 and 9, CCFPG outperforms CCFPG w/o attention, which shows the strength of using the attention mechanism to aggregate messages. When the number of agents grows further (i.e., 12 agents), we can see that CCFPG w/o attention is slightly better than CCFPG. We think this is because the attention mechanism has additional parameters to train which slows down learning in complex settings. Nevertheless, CCFPG still obtains higher (average) returns than G2ANet and FCMNet. We report the mean and variance of the average return of the last 1000 evaluation episodes in predator-prey in Table 2. As we can see, in predator-prey with 6 and 9 agents, CCFPG achieves a higher mean and lower variance compared to Facmac. In the 12-agent case, Facmac has a lower variance while achieving a much lower mean return than our proposed methods.

5.2 Multi-Agent MuJoCo

Multi-Agent MuJoCo (MAMuJoCo) is a novel benchmark for continuous cooperative multi-agent robotic control. MAMuJoCo extends from the popular fully observable single-agent robotic MuJoCo by creating a wide variety of novel scenarios in which multiple agents

Table 2: Results for algorithms across different settings in Predator-Prey. Final average ± standard error across 9 trials of returns across the last 1000 evaluation trajectories. We highlight the maximum mean value for each setting in bold.

Algorithm	PP 6 Agents	PP 9 Agents	PP 12 Agents
Facmac	211.58±40.79	201.6±83.67	67.98±52.27
FCMNet	9.4±0.65	21.69±1.27	41.38±2.97
G2ANet	7.27±2.64	21.48±5.8	42.48±5.17
CCFPG w/o attention	241.56±46.47	190.54±104.62	**131.47±65.72**
CCFPG	**256.17±40.29**	**225.17±64.87**	117.15±67.69

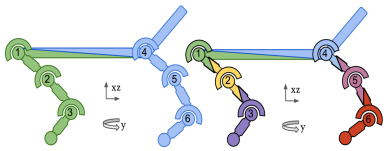

Figure 4: Agent partitionings for HalfCheetah tasks. Colors indicate agent partitionings. Each index corresponds to a single controllable joint. Left: 2-Agent HalfCheetah, where each agent controls 3 joints. Right: 6-Agent HalfCheetah, where each agent controls 1 joint.

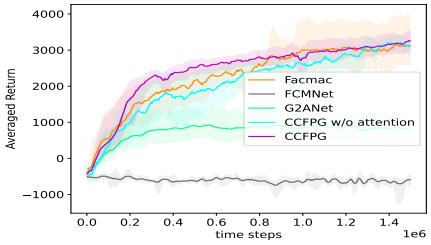

Figure 5: The testing return of 2-Agent HalfCheetah.

within a single robot have to solve a task cooperatively. We test all methods on HalfCheetah tasks with 2- and 6-Agent partitionings. The configurations are shown in Figure 4. In HalfCheetah tasks, each agent can observe only the positions of its own body parts. The rewards of agents depending on the speed of the robot and the contact forces of the body. The faster the cheetah, the more rewards the agents can get. Without access to the exact full state, each agent has to develop local decision rules for the corresponding physical components of the robot. With communication, agents can more easily coordinate their behaviours to avoid failures as well as achieve better movements.

We test all methods on 2-Agent HalfCheetah and 6-Agent HalfCheetah. As we can see in Figure 5, CCFPG learns much faster than all the other methods before 1 million time steps and then achieves similar performance as Facmac and CCFPG w/o attention. In Figure 6 6-Agent HalfCheetah task, Facmac converges very fast while CCFPG achieves significantly higher returns than other methods. Compared to CCFPG w/o attention, CCFPG benefits from the attention mechanism and obtain better performance. The communication

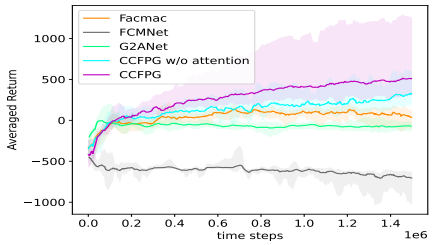

Figure 6: The testing return of 6-Agent HalfCheetah.

methods, G2ANet and FCMNet can not utilize communication effectively and are even worse than Facmac. The results of HalfCheetah tasks show that communication is essential for learning and our proposed method can efficiently and effectively use communication. We report the mean and variance of the average return of the last 1000 evaluation episodes in HalfCheetah in Table 3. As we can see, in 2-Agent HalfCheetah, CCFPG achieves a higher mean and lower variance compared to Facmac. In the 6-Agent HalfCheetah, CCFPG and CCFPG w/o attention have a much higher mean return than other methods while the variance is high.

Table 3: Results for algorithms across different settings in HalfCheetah. Final average ± standard error across 9 trials of returns across the last 1000 evaluation trajectories. We highlight the maximum mean value for each setting in bold.

Algorithm	2-Agent HalfCheetah	6-Agent HalfCheetah
Facmac	3049.05±728.11	72.37±129.53
FCMNet	-630.77±87.24	-681.02±171.68
G2ANet	831.64±536.04	-73.65±79.58
CCFPG w/o attention	3122.09±186.62	275.97±228.49
CCFPG	**3147.42±313.99**	**486.46±558.32**

5.3 The Effect of Dimensionality

The dimensionality of messages may affect the learning of agents. Higher dimensionality brings difficulties to learning as agents have to search in a larger message space. Thus, we investigate the effect of different dimensionalities of messages on our proposed methods in Predator-prey with 6 agents and 2-Agent HalfCheetah. The learning curves are shown in Figures 7 and 8. As we can see, CCFPG w/o attention can achieve better performance in a high dimension of messages in Predator-prey, as shown in Figure 7. However, CCFPG can obtain similar performance using a relatively low dimension of messages. This is especially useful when the size of messages is considered as a cost. In HalfCheetah (Figure 8), CCFPG w/o attention achieves lower and unstable performance with a large dimension of messages. This is similar in CCFPG with attention. We think this is because the observations have low dimensionality in the HalfCheetah task (3 dimensions in 2-Agent HalfCheetah and 1 dimension in 6-Agent HalfCheetah). Projecting low-dimension observations into high-dimension messages can be redundant and not necessary. In summary, for both CCPFG and CCFPG w/o attention, a reasonable

Table 4: Bootstrap mean and 95% confidence bounds in predator-prey and 90% confidence bounds in HalfCheetah. (a, b) is the upper and lower bounds.

Algorithm	PP 6 Agents	PP 9 Agents	PP 12 Agents	2-Agent HalfCheetah	6-Agent HalfCheetah
Facmac	140 (136, 144)	101 (98, 104)	53 (53, 53)	2274 (2203, 2346)	45 (39, 52)
FCMNet	9 (9, 9)	21 (21, 21)	40 (40, 40)	-605 (-609, -602)	-611 (-614, -608)
G2ANet	11 (11, 11)	23 (23, 23)	43 (43, 43)	749 (725, 774)	-64 (-66, -62)
CCFPG w/o attention	160 (156, 164)	95 (92, 98)	86 (84, 87)	2082 (2007, 2158)	135 (125, 146)
CCFPG	176 (172, 181)	129 (125, 132)	76 (75, 78)	2433 (2361, 2505)	281 (265, 298)

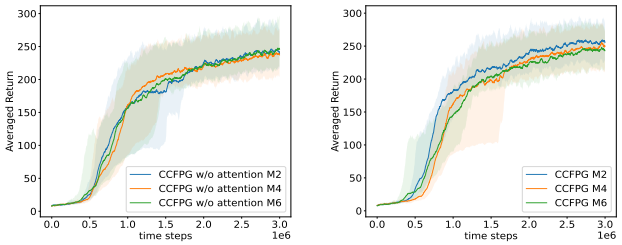

Figure 7: The learning curves of our methods in Predator-prey with 6 agents with different dimensionalities of messages (M). Left: CCFPG w/o attention. Right: CCFPG.

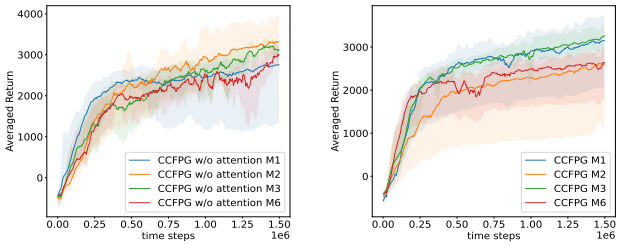

Figure 8: The learning curves of our methods in 2-Agent HalfCheetah with different dimensionalities of messages (M). Left: CCFPG w/o attention. Right: CCFPG.

number of dimensions of messages can lead to a stable and better improvement.

5.4 Discussion

To answer whether the performance improvement of our proposed methods is indeed significant, we conduct statistical tests for all methods. Table 4 shows the bootstrap mean and confidence bounds in predator-prey and HalfCheetah. Confidence bounds can vary wildly between algorithms and environments. We find that in predator-prey CCFPG surpasses Facmac and other baseline communication methods with 95% confidence. In HalfCheetah, CCFPG surpasses Facmac and other baseline communication methods with 90% confidence.

We also investigate how comparing methods behave in a longer time step. Due to the time limit, we only run predator-prey with 6 agents and 2-Agent HalfCheetah tasks, as shown in Figure 9. As

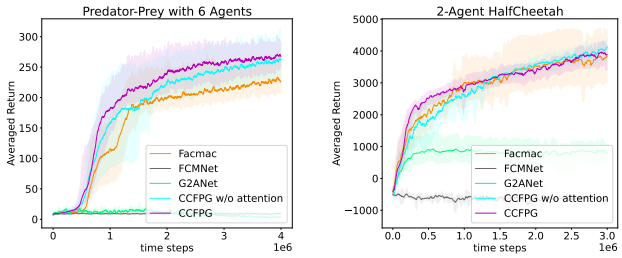

Figure 9: The learning curves of all methods with extended time steps in Predator-Prey with 6 agents (left) and 2-Agent HalfCheetah (right).

we can see, in predator-prey with 6 agents, CCFPG and CCFPG w/o attention gradually achieve similar performance while the average return of other methods are still much lower. In 2-Agent HalfCheetah tasks, Facmac is slightly better than CCFPG between 1.8 million and 2.5 million time steps. After 2.5 million time steps, CCFPG and CCFPG w/o attention achieve a (slightly) higher return than Facmac.

6 CONCLUSION

This paper presents a new policy gradient MADRL algorithm, Continuous Communication with Factorized Policy Gradients (CCFPG), which utilizes a centralized but factorized Q-function for the learning of communication. Due to the deterministic design of actors and communication, agents and their dedicated communication architecture can be trained in an end-to-end fashion. We show the advantage of using such a factorized critic and the attention mechanism to aggregate real-valued messages in two multi-agent tasks with continuous control. Our results demonstrate CCFPG's superior performance over existing learning with communication methods. Future works will explore more forms of value decomposition methods and the effect of communication graphs on performance. Surprisingly, CCFPG does not suffer from vanishing gradient problems in predator-prey and HalfCheetah. In the future, we could investigate how CFFPG performs in high complex and stochastic environments.

ACKNOWLEDGEMENT

This work made use of the Dutch national e-infrastructure with the support of the SURF Cooperative.

REFERENCES

[1] Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890.

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

[3] Ziluo Ding, Tiejun Huang, and Zongqing Lu. 2020. Learning Individually Inferred Communication for Multi-Agent Cooperation. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/fb2fcd534b0ff3bbed73cc51df620323-Abstract.html

[4] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems 29 (NIPS)*. 2137–2145.

[5] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. [n.d.]. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). 2974–2982.

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[7] Guangzheng Hu, Yuanheng Zhu, Dongbin Zhao, Mengchen Zhao, and Jianye Hao. 2020. Event-Triggered Multi-agent Reinforcement Learning with Communication under Limited-bandwidth Constraint. *CoRR* abs/2010.04978 (2020). arXiv:2010.04978 https://arxiv.org/abs/2010.04978

[8] Jiechuan Jiang and Zongqing Lu. 2018. Learning Attentional Communication for Multi-Agent Cooperation. In *Advances in Neural Information Processing Systems 31 (NIPS)*. 7265–7275.

[9] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. [n.d.]. Learning to Schedule Communication in Multi-agent Reinforcement Learning. In *7th International Conference on Learning Representations (ICLR)*.

[10] Jens Kober, J. Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *Int. J. Robotics Res.* 32, 11 (2013), 1238–1274. https://doi.org/10.1177/0278364913495721

[11] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. 2020. Multi-Agent Game Abstraction via Graph Attention Neural Network. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 7211–7218.

[12] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 6379–6390. https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html

[13] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. 2021. Contrasting Centralized and Decentralized Critics in Multi-Agent Reinforcement Learning. In *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, Frank Dignum, Alessio Lomuscio, Ulle Endriss, and Ann Nowé (Eds.). ACM, 844–852. https://doi.org/10.5555/3463952.3464053

[14] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, Zhibo Gong, and Yan Ni. [n.d.]. Learning Agent Communication under Limited Bandwidth by Message Pruning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. 5142–5149.

[15] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer. https://doi.org/10.1007/978-3-319-28929-8

[16] Bei Peng, Tabish Rashid, Christian Schröder de Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Boehmer, and Shimon Whiteson. 2021. FACMAC: Factored Multi-Agent Centralised Policy Gradients. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 12208–12221. https://proceedings.neurips.cc/paper/2021/hash/65b9eea6e1cc6bb9f0cd2a47751a186f-Abstract.html

[17] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 4292–4301. http://proceedings.mlr.press/v80/rashid18a.html

[18] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, 2186–2188. http://dl.acm.org/citation.cfm?id=3332052

[19] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. *CoRR* abs/1610.03295 (2016). arXiv:1610.03295 http://arxiv.org/abs/1610.03295

[20] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nat.* 550, 7676 (2017), 354–359. https://doi.org/10.1038/nature24270

[21] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. 2019. Learning when to Communicate at Scale in Multiagent Cooperative and Competitive Tasks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. https://openreview.net/forum?id=rye7knCqK7

[22] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5887–5896. http://proceedings.mlr.press/v97/son19a.html

[23] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems 29 (NIPS)*. 2244–2252.

[24] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 2244–2252. https://proceedings.neurips.cc/paper/2016/hash/55b1927fdafef39c48e5b73b5d61ea60-Abstract.html

[25] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar (Eds.). International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2085–2087. http://dl.acm.org/citation.cfm?id=3238080

[26] Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier. 2016. TorchCraft: a Library for Machine Learning Research on Real-Time Strategy Games. *CoRR* abs/1611.00625 (2016). arXiv:1611.00625 http://arxiv.org/abs/1611.00625

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[28] Meritxell Vinyals, Juan A. Rodríguez-Aguilar, and Jesús Cerquides. 2011. A Survey on Sensor Networks from a Multiagent Perspective. *Comput. J.* 54, 3 (2011), 455–470. https://doi.org/10.1093/comjnl/bxq018

[29] Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich. [n.d.]. Learning Efficient Multi-agent Communication: An Information Bottleneck Approach. In *Proceedings of the 37th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 119)*. 9908–9918.

[30] Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. [n.d.]. Learning Nearly Decomposable Value Functions Via Communication Minimization. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

[31] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. 2021. DOP: Off-Policy Multi-Agent Decomposed Policy Gradients. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=6FqKiVAdI3Y

[32] Yutong Wang and Guillaume Sartoretti. 2022. FCMNet: Full Communication Memory Net for Team-Level Cooperation in Multi-Agent Systems. *CoRR* abs/2201.11994 (2022). arXiv:2201.11994 https://arxiv.org/abs/2201.11994

[33] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. 2019. Efficient Communication in Multi-Agent Reinforcement Learning via Variance Based Control. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman

Garnett (Eds.). 3230–3239. https://proceedings.neurips.cc/paper/2019/hash/14cfdb59b5bda1fc245aadae15b1984a-Abstract.html

[34] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. 2020. Succinct and Robust Multi-Agent Communication With Temporal Message Control. In *Advances in Neural Information Processing Systems 33 (NIPS)*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/c82b013313066e0702d58dc70db033ca-Abstract.html

[35] Changxi Zhu, Mehdi Dastani, and Shihan Wang. 2022. A Survey of Multi-Agent Reinforcement Learning with Communication. *CoRR* abs/2203.08975 (2022). https://doi.org/10.48550/arXiv.2203.08975 arXiv:2203.08975