A Meta-Gradient Approach to Learning Cooperative Multi-Agent Communication Topology

Dingyang Chen
Artificial Intelligence Institute
University of South Carolina
Columbia, SC
dingyang@email.sc.edu

Qi Zhang
Artificial Intelligence Institute
University of South Carolina
Columbia, SC
qz5@cse.sc.edu

Abstract

In cooperative multi-agent reinforcement learning (MARL), agents often can only partially observe the environment state, and thus communication is crucial to achieving coordination. Communicating agents must simultaneously learn to whom to communicate (i.e., communication topology) and how to interpret the received message for decision-making. Although agents can efficiently learn communication interpretation by end-to-end backpropagation, learning communication topology is much trickier since the binary decisions of whether to communicate impede end-to-end differentiation. As evidenced in our experiments, existing solutions, such as reparameterization tricks and reformulating topology learning as reinforcement learning, often fall short. This paper introduces a meta-learning framework that aims to discover and continually adapt the update rules for communication topology learning. Empirical results show that our meta-learning approach outperforms existing alternatives in a range of cooperative MARL tasks and demonstrates a reasonably strong ability to generalize to tasks different from meta-training. Preliminary analyses suggest that, interestingly, the discovered update rules occasionally resemble the human-designed rules such as policy gradients, yet remaining qualitatively different in most cases.

1 Introduction

There have been significant successes of reinforcement learning (RL) recently. Many RL applications involve multiple agents learning and adapting simultaneously to achieve shared goals, which naturally fall into the framework of cooperative multi-agent RL (MARL). In addition to challenges in single-agent RL, there are (at least) two challenges unique to cooperative MARL: (i) Since all agents are updating their policies, the environment becomes nonstationary from the perspective of any individual agent, violating assumptions in traditional single-agent RL and perhaps resulting in learning instability; and (ii) Compared to single-agent RL, the issue of partial observability is often more severe in MARL, since each individual agent might only observe a fraction of the information in the global state, leading to potential failure to coordinate. To cope with (i), the paradigm of centralized training and decentralized execution (CTDE) [1, 2] have been proposed, where agents are trained with access to global information to mitigate the issue of nonstationarity and choose actions during execution independently in a decentralized manner. To cope with (ii), it is crucial to enable communication among agents during execution [3, 4], where agents share their local information to others, such that each agent can form better knowledge about the global state, leading to better cooperation. This paper aims to better address multi-agent communication during execution under the CTDE paradigm.

5th Workshop on Meta-Learning at NeurIPS 2021.

For effective multi-agent communication, individual agents must simultaneously solve the problem of communication interpretation, i.e., how to utilize the received messages from other agents to make better decisions, and the preceding problem of communication topology, i.e., whom to communicate to. Recently, many research works have shown that it is effective to learn communication interpretation by end-to-end backpropagation (e.g., [5, 3]). However, the end-to-end learning of communication topology is much trickier, since the binary decisions of to whom to communicate impede end-to-end differentiation. Prior work either uses reparameterization tricks such as Gumbel-Softmax [6] to force differentiability (e.g., [7]) or reformulates the learning of communication decisions as an RL problem to apply policy gradient methods (e.g., [8]). However, these existing solutions are not completely satisfactory: Gumbel-Softmax is known to be unstable, especially for RL tasks; the formulation can dramatically slow down training due to the trial-and-error nature of RL, especially considering that the decision space for communication topology is often large.

Motivated by the aforementioned challenges, in this paper we propose to employ a meta-learning framework that formulates the problem of discovering an update rule for learning communication topology. We then develop an architecture and an algorithm for discovering and adaptively adjusting the update rule in an online fashion, with several key design choices aimed to stabilize and facilitate the meta-learning process. Experimental results show that our algorithm outperforms alternative baselines such as Gumbel-Softmax and policy gradient for learning communication topology. Our ablation study confirms the importance of our key design choices for performance. We also conduct analyses showing that the meta-learned update rule can transfer to different tasks with reasonable effectiveness. Surprisingly, while it outperforms the policy gradient baseline in most cases, we found that the update rule discovered by the meta-learning framework occasionally resembles patterns of the policy gradient update rule, yet remains qualitatively different.

2 Related work

Learning to learn. Meta-learning, aka learning to learn, refers to the objective of improving the learning process itself (usually an optimization process) to be effective for a variety of learning tasks. This idea has been around since the late 80s with various formulations such as improving the genetic programming process [9], learning update rules for neural networks [10], improving domain-invariant transfer of learned knowledge [11], and, more recently, identifying initial neural network parameters for fast adaptation [12] using meta-gradients, i.e., gradients obtained from backpropagation of the meta-learning objective. Recently, Xu et al. [13] introduced the framework of using meta-gradients to improve the learning of an RL agent, which has been applied to various components of an RL algorithm, including intrinsic rewards [14], auxiliary tasks [15], hyperparameters of an actor-critic loss function [16], update target [17, 18]. Meta-learning for multi-agent RL is relatively under-explored. Du et al. [19] have applied meta-gradients to the discovery of multi-agent intrinsic rewards. In contrast, our work has an orthogonal contribution of using meta-gradients to learn multi-agent communication topology.

Communication in cooperative MARL. Communications between agents play an important role in solving the nonstationarity and partial observability in MARL. A popular training framework to solve the issue of nonstationarity is CDTE [2], where global information is obtained by all-to-all communication during training (usually achieved by a centralized module) to mitigate nonstationarity, and no communication during execution to enable fast deployment. However, when agents only partially observe the environment state, the augmentation of communication during execution becomes necessary. A commonly used way is to broadcast the messages from one agent to all the others during execution if the communication is determined necessary for the current timestep [3][20][21]. However, such broadcasting of information is not only unscalable when the number of agents is large[22], but also can potentially include redundant or even harmful information [23][24]. Thus, finding topology graphs to selectively communicate is crucial. The topology can be obtained by trainable networks that make the communication decisions [25][22], or obtained naturally by the physical constraints e.g. only agents within certain distances are viewable [4]. We give a more detailed description of existing methods on finding communication topology in Section 4.1, in distinction to our approach.

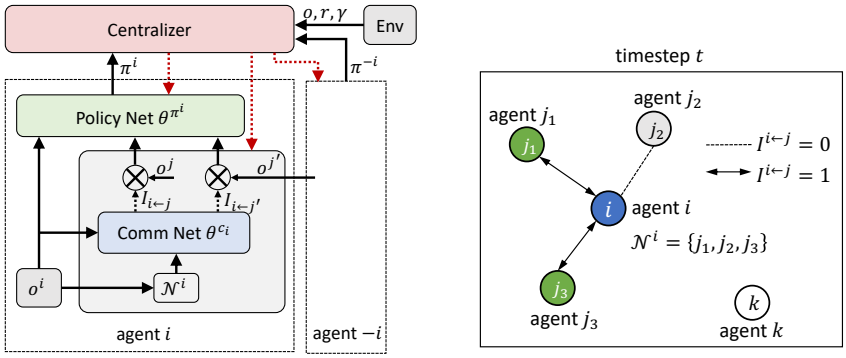

Figure 1: *Left*: Our architecture for learning communication topology of cooperative MARL tasks. *Right*: Illustration of an agent i selecting its communicating neighbors at a particular timestep.

3 Preliminaries

We consider fully cooperative multi-agent tasks that can be modeled as Markov games [26] augmented with networked communication [4, 25], in which N agents, indexed by $i \in \mathcal{N} := \{1, .., N\}$, choose sequential actions. At each timestep t, the task has a global state $s_t \in \mathcal{S}$ that is Markovian, and observation function $f^i : \mathcal{S} \to \mathcal{O}^i$ yields observation $o_t^i \in \mathcal{O}^i$, for each agent i where $\mathcal{O}^i$ is its observation space. At timestep t, each agent i chooses an action $a_t^i \in \mathcal{A}^i$ a, forming a joint action $a_t := (a_t^1, ..., a_t^N) \in \mathcal{A} := \times_i \mathcal{A}^i$ that induces a transition of the global state according to the state transition function $P(s'|s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. The N agents choose actions after selectively communicating through a time-variant network, defined by a undirected graph $\mathcal{G}_t := (\mathcal{N}, \mathcal{E}_t)$ with vertex set $\mathcal{N}$ and edge set $\mathcal{E}_t \subseteq \{(i, j) : i, j \in \mathcal{N}, i \neq j\}$. Specifically, let $\mathcal{N}_t^i := \{j : (i, j) \in \mathcal{E}_t\}$ be the neighbors of agent i at timestep t. Based on its observation o_t^i, agent i selects a subset of its neighbors, $\mathcal{C}_t^i \subseteq \mathcal{N}_t^i$, and request the observations from the selected subset, $o_t(\mathcal{C}_t^i) := \{o_t^j : j \in \mathcal{C}_t^i\}$. Then, agent i chooses action according to its policy π^i, $a_t^i \sim \pi^i(o_t^i, o_t(\mathcal{C}_t^i))$, conditioning on both its own observation and observations of the selected neighbors. We consider the fully cooperative setting in which agents optimize their policies with respect to global reward function $r(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and discount factor $\gamma \in [0, 1]$. The discounted return from timestep t is $G_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$, where $r_t := r(s_t, a_t)$ is the reward at timestep t. The agents' joint policy $\pi = (\pi^1, ..., \pi^N)$ induce a value function, which is defined as $V^\pi(s_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t:\infty}}[G_t|s_t]$, and action-value function $Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}[G_t|s_t, a_t]$.

4 Methods

Figure 1 depicts our neural network architecture for solving the problem formulated in Section 3 with a learned communication topology. The architecture is compatible with any instantiation of the CTDE framework. Specifically, it consists of a communication network for each agent, a policy network for each agent, and a centralizer module shared by all agents, which are described next in detail. For the rest of the paper, we omit the subscript of timestep t when the timestep is clear or irrelevant to the context.

Comm Net. At each timestep, the communication network selects a subset of each agent's neighbors to communicate with based on its current observation, and thus determining the communication topology. Specifically, we assume agent i's observation o^i contains the IDs of its neighbors $\mathcal{N}^i$. Each agent i's communication network $c^i(\cdot)$ takes o^i as input and outputs $c^{i\leftarrow j}(o^i) \in [0, 1], \forall j \in \mathcal{N}^i$, which is interpreted as the probability that agent i communicates with neighboring agent j for its observation o^j. Letting $I^{i\leftarrow j}$ be the corresponding Bernoulli random variable with mean $c^{i\leftarrow j}(o^i)$, the selected subset of communicating neighbors is thus $\mathcal{C}^i = \{j \in \mathcal{N}^i : I^{i\leftarrow j} = 1\}$. The output size of the communication network is $|\mathcal{N}^i|$. When $|\mathcal{N}^i|$ is a constant (e.g., an agent has a constant number of $n(< N)$ neighbors), and we can use a simple multi-layer perceptron (MLP) as the communication network. In general, $|\mathcal{N}^i|$ is a time-variant variable, and architecture choices such as LSTMs [27],

transformers [28], and Graph Convolutional Nets (GCNs) [29] to deal with the variable output size. We use θ_c^i to denote the parameters for the communication network of agent i.

Policy Net. In this paper, we consider deterministic policies which are suitable for continuous action spaces. Our method can be straightforwardly adapted to the stochastic policy case. Specifically, the agent i's policy network $\pi^i(\cdot)$ takes as input its local observation o^i as well as the received observations $o(\mathcal{C}^i)$ and deterministically outputs an action $\pi^i(o^i, o(\mathcal{C}^i)) \in \mathcal{A}^i$. In order to scale to an arbitrary number of communicating neighbors, in practice we treat $o(\mathcal{C}^i)$ as a sequence of length $|\mathcal{C}^i|$ use an LSTM network that processes the sequence. The final hidden state of the LSTM network is concatenated with local observation o^i and then fed into a multi-layer perceptron (MLP) to produce the action. We use θ_π^i to denote the parameters for the policy network of agent i.

Centralizer. As an instantiation of the CTDE framework, our architecture involves a centralizer module that utilizes global information to guide the training of the modules for decentralized execution. As the communication and policy networks are the decentralized execution modules, the role of our centralizer is to provide learning signals for these two types of networks. We follow the prior work and consider the typical centralized critic that seeks to approximate the action-value function of the joint policy, $Q(o, a) \approx Q^\pi(s, a)$, where $o := (o^1, ..., o^N)$ is the joint observation and $a := (a^1, ..., a^N)$ is the joint action; if the global state s is conveniently available, joint observation o can be replaced by s. Given joint observation $o \in \mathcal{O}$, the communication networks and the policy networks $\{c^i, \pi^i\}_{i\in\mathcal{N}}$ together define a (stochastic) joint policy $\pi : \mathcal{O} \to \Delta(\mathcal{A})$ that maps the joint observation to a distribution over the joint action, where the stochasticity comes from the selection of communicating neighbors by the communication networks (assuming the policy network is deterministic). Letting the critic be parameterized by θ_Q, the critic can be trained by minimizing the temporal difference (TD) error:

$$\mathcal{L}(\theta_Q) = \mathbb{E}_{(o,a,r,o')\sim\mathcal{D}}\left[(Q(o, a; \theta_Q) - y)^2\right], \qquad y = r + \gamma Q'(o', a'; \theta'_Q)|_{a'\sim\pi'(o')} \tag{1}$$

where $\mathcal{D}$ is the replay buffer recording joint observations and actions; TD target y is computed by the target critic network Q' with delayed parameters θ'_Q and target joint policy π' with delayed parameters of the communication and policy networks. Following MADDPG [2], the centralized critic can be used to guide the optimization of each agent i's network parameters by the optimization of

$$\max_{\theta_\pi^i, \theta_c^i} J(\theta_\pi^i, \theta_c^i) \quad \text{with } J(\theta_\pi^i, \theta_c^i) = \mathbb{E}_{(o,a)\sim\mathcal{D}}\left[Q(o, a^i, a^{-i}; \theta_Q)|_{a^i=\pi^i(o^i, o(\mathcal{C}^i); \theta_\pi^i), \mathcal{C}^i\sim c^i(o^i;\theta_c^i)}\right]. \tag{2}$$

Using the chain rule on the deterministic policy network $a^i = \pi^i(o^i, o(\mathcal{C}^i); \theta_\pi^i)$, the gradient w.r.t. θ_π^i is derived as

$$\nabla_{\theta_\pi^i} J(\theta_\pi^i, \theta_c^i) = \mathbb{E}_{(o,a)\sim\mathcal{D}}\left[\nabla_{\theta_\pi^i}\pi^i(o^i, o(\mathcal{C}^i); \theta_\pi^i)\nabla_{a^i}Q(o, a^i, a^{-i})|_{a^i=\pi^i(o^i, o(\mathcal{C}^i); \theta_\pi^i), \mathcal{C}^i\sim c^i(o^i;\theta_c^i)}\right]. \tag{3}$$

4.1 Problem of learning communication topology and existing solutions

In contrast with the policy network, the gradient of objective (2) w.r.t. the communication network's parameters θ_c^i cannot be obtained via the chain rule, since the sampling procedure for the communicating neighbors, $\mathcal{C}^i \sim c^i(o^i; \theta_c^i)$, is non-differentiable. Before presenting our solution using the meta-gradient framework in Section 4.2, we here review existing solutions in prior work:

No communication, i.e., $\mathcal{C}_t^i = \varnothing\ \forall i, t$. At the extreme of the no communication topology, no agent communicates with any neighbor at any time, and thus the action choice is only based on the local observation, i.e., $a_t^i \sim \pi^i(o_t^i)\ \forall i, t$. Early work on CTDE, e.g., MADDPG [2], uses no communication as the default topology. No communication suffers from the issue of partial observability of individual agents.

Full communication, i.e., $\mathcal{C}_t^i = \mathcal{N}_t^i\ \forall i, t$. This solution bypasses the problem of communication topology learning by simply letting all agents always communicate with all neighbors. Prior work such as DIAL [5] and CommNet [3] employs such full communication topology. Such topology suffers from large communication overhead, especially when the number of agents is large. Moreover, later work (e.g., TarMAC [20]), as well as our empirical results in Section 5, shows that full communication can even hinder the learning performance, presumably because the agents are forced to interpret usually excessive messages received from all neighbors.

RL for communication topology. The communication decisions $\{I_t^{i\leftarrow j}\}_{j\in\mathcal{N}_t^i}$ made by agent i at timestep t can be viewed as a $|\mathcal{N}_t^i|$-bit action in the RL sense, with reward r_t simply being the original task reward. Therefore, any (multi-agent) RL algorithm can in principle be applied to obtain the update for the communication network c^i that makes the decisions, e.g., REINFORCE [30]:

$$\Delta\theta_c^i \propto \mathbb{E}\left[\nabla_{\theta_c^i}\sum_t\sum_{j\in\mathcal{N}_t^i}\left(\log c^{i\leftarrow j}(I_t^{i\leftarrow j}|o_t^i;\theta_c^i)\right)G_t\right] \quad \forall i\in\mathcal{N} \tag{4}$$

where the expectation is w.r.t. trajectories generated by on-policy parameters $\{\theta_c^i,\theta_\pi^i\}_{i\in\mathcal{N}}$. This idea of reformulating topology learning as an RL problem has been explored in prior work. For example, I3CNet [8] uses REINFORCE to train a single gating mechanism for each agent to decide whether to broadcast its observation to all of its neighbors. However, no prior work has attempted to use RL for training agent-to-agent communication decision-making like our communication network. We include REINFORCE as a baseline in our experiments, yet we do not claim it as our main contribution.

Reparameterization tricks. Reparameterization tricks are widely used to address the non-differentiability caused by sampling. Specifically, Straight Through Gumbel-Softmax [6] can be used for categorical samplings like $I_t^{i\leftarrow j}\sim c^{i\leftarrow j}(o_t^i)$. While convenient to use, Gumbel-Softmax can be very unstable. Without carefully chosen regularizers, Gumbel-Softmax can easily degenerate to full communication, as evidenced in prior work [7, 31, 32] and our experiments.

4.2 Learning communication networks with meta-gradients

Inspired by the success of single-agent meta-RL [13, 14, 17, 18], we here develop a meta-gradient framework to train the communication networks. Our main hypothesis of this work is that *the communication topology learning rule obtained from our meta-gradient based optimization outperforms the traditional methods based on the policy gradient and Gumbel-Softmax* . The meta-gradient approach will yield a learning rule with independent advantage estimates for each (i,j) pair of agents (cf. $\hat{A}_t^{i\leftarrow j}$ in (5)), which potentially reduces variance and speeds up learning compared with the policy gradient method (4) that shares the same advantage estimate G_t for all pairs of agents. Moreover, the meta-learned update rule does not involve any reparameterization tricks such as Gumbel-Softmax that can result in unstable learning.

As for the architecture, the centralizer module is augmented with a backward LSTM network that takes as input the reverse trajectory of all the agents and produces the update direction for each agent's communication network. Specifically, the backward LSTM takes as input $x_t := [r_t, d_t, \gamma, \{c^{i\leftarrow j}(o_t^i)\}_{(i,j)\in\mathcal{N}\times\mathcal{N}_t^i}]$ at each timestep t of a trajectory, where r_t is the shared reward, d_t is the binary value indicating episode termination, γ is the discount factor, and $c^{i\leftarrow j}(o_t^i)$ is the probability of agent i communicating with neighbor j at timestep t that is computed by communication network $c^i(\cdot)$. Since $|\mathcal{N}_t^i|$ is in general a time-variant variable, the size of input x_t is time-variant. We assume $|\mathcal{N}_t^i|$ to be bounded by d (i.e., an agent has at most d neighbors), which is always possible since d can be set to $N-1$. Thus, the size of input x_t is time-invariant, and x_t can be flattened for the LSTM input. The LSTM outputs, for each agent-neighbor pair of $(i,j)\in\mathcal{N}\times\mathcal{N}_t^i$ at each timestep t, a scalar $\hat{A}_t^{i\leftarrow j}\in\mathbb{R}$ that defines the update of the communication network:

$$\Delta\theta_c^i \propto \mathbb{E}\left[\nabla_{\theta_c^i}\sum_t\sum_{j\in\mathcal{N}_t^i}\left(\log c^{i\leftarrow j}(I_t^{i\leftarrow j}|o_t^i;\theta_c^i)\right)\hat{A}_t^{i\leftarrow j}\right] \quad \forall i\in\mathcal{N}. \tag{5}$$

Compared with the REINFORCE update in Equation (4), the term of return G_t is replaced by $\hat{A}_t^{i\leftarrow j}$ that is computed by the backward LSTM.

We use η to denote the meta-parameters for the backward LSTM network that defines the update of Equation (5). The objective of our meta-learning framework is to find the meta-parameters η that leads to high expected return after a number of K updates for the communication network parameters $\theta_c := \{\theta_c^i\}_{i\in\mathcal{N}}$, given some fixed policy network parameters $\theta_\pi := \{\theta_\pi^i\}_{i\in\mathcal{N}}$

$$\arg\max_\eta J_K(\eta) := \mathbb{E}_{\theta_c(0)}[G(\theta_c(K))] \tag{6}$$

where the expectation is w.r.t. random initializations of $\theta_c(0)$; the initial $\theta_c(0)$ is updated K times to $\theta_c(K)$ using the fixed η, $(\theta_c(0)\to\ldots\to\theta_c(K)\mid\eta)$; $G(\theta_c(K)) := \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t r_t|\theta_c(K),\theta_\pi]$ is the expected return for the communication network after the K updates $\theta_c(K)$ and the fixed policy network parameters θ_π.

Since the updates of θ_c are differentiable with respect to η, we can consider optimizing objective (6) with the meta-gradient derived using the chain rule $\nabla_\eta J_K(\eta) = \nabla_{\theta_c(K)} J_K(\eta) \nabla_\eta \theta_c(K)$, with $\nabla_{\theta_c(K)} J_K(\eta)$ obtained from the policy gradient, e.g., REINFORCE:

$$\nabla_\eta J_K(\eta) = \mathbb{E}_{\theta_c(0)} \left[\nabla_\eta \sum_t \sum_{(i,j)\in\mathcal{N}\times\mathcal{N}_t^i} \left(\log c^{i\leftarrow j} \left(I_t^{i\leftarrow j} | o_t^i; \theta_c^i(K) \right) \right) G_t \left(\theta_c(K) \right) \right] \tag{7}$$

where $G_t(\theta_c(K)) := \mathbb{E}[\sum_{t'=t}^{\infty} \gamma^{t'} r_{t'} | \theta_c(K), \theta_\pi]$ is the expected return starting from timestep t. The true meta-objective (6) and its meta-gradient (7) require backpropagation through the entire K updates of θ_c, which is infeasible due to memory constraint if K is large. In practice, we perform meta-gradient optimization every a small number of K updates while continually updating θ_c to the end of learning.

Concurrent parameter updates. The derivation of meta-gradient (7) assumes fixed policy network parameters, which makes the environment stationary from the perspective of the communication networks and thus the policy gradient can be used to calculate $\nabla_{\theta_c(K)} J_K(\eta)$. In practice, to speed up learning we consider concurrent updates for all parameters, including the communication networks θ_c, the policy networks θ_π, and the centralized critic θ_Q.

Learned baseline. To reduce variance, we separately use a baseline $V(o_t; \phi) \approx G_t$ parameterized by ϕ to approximate the expected return under the current communication and policy networks, which is learned using n-step TD:

$$\Delta\phi \propto \left(G_t^{\phi,n} - V(o_t;\phi) \right) \nabla_\phi V(o_t;\phi), \qquad G_t^{\phi,n} = \sum_{k=0}^{n-1} \gamma^k r_{t+k+1} + \gamma^n V(o_{t+n};\phi).$$

The $G_t(\theta_c(K))$ term in meta-gradient (7) is replaced by $G_t^{\phi,n} - V(o_t;\phi)$, where the trajectory for computing $G_t^{\phi,n}$ is sampled from the on-policy parameters $(\theta_c(K), \theta_\pi(K))$ that have been concurrently updated. In our empirical work, we also use a learned baseline for the original REINFORCE (4) for fair comparison.

Entropy regularization. We also propose to add entropy regularization on the communication network outputs $c^{i\leftarrow j}$ to encourage exploration and facilitate learning. The regularized meta-gradient becomes

$$\nabla_\eta(K) := \nabla_\eta J_K(\eta) + \beta \nabla_\eta \sum_t \sum_{(i,j)\in\mathcal{N}\times\mathcal{N}_t^i} H \left(c^{i\leftarrow j} \left(o_t^i; \theta_c^i(K) \right) \right) \tag{8}$$

where $H(\cdot)$ is the entropy of the Bernoulli distribution $c^{i\leftarrow j}$ that is determined by the post-update $\theta_c^i(K)$, and $\beta \in \mathbb{R}^+$ is the regularization coefficient.

Blending intermediate meta-gradients. The regularized meta-gradient $\nabla_\eta(K)$ is derived through the parameters after the K-th update, i.e., $\theta_c(K)$. In practice, we find it effective to blend it with meta-gradients derived through the intermediate parameters, resulting in our final update for meta-parameters η:

$$\Delta\eta \propto \frac{1}{K} \sum_{k=1}^{K} \nabla_\eta(k) \tag{9}$$

where $\nabla_\eta(k)$ is the meta-gradient derived through $\theta_c(k)$ in the same way as Equation (8). We hypothesize that such blending in Equation (9) can reduce the variance of the gradient estimation and therefore benefit the optimization of the meta-parameters.

The resulting algorithm is specified in Algorithm 1.

5 Experiments

We aim to answer the following questions in Sections 5.1-5.3, respectively:

- How effective is our algorithm for learning cooperative communication topology?
- How effective is the meta-parameters *after* being optimized?
- How important are the design choices described in Section 4.2?
- What kind of update rule is learned from the meta-learning process?

Environment. We consider the following scenarios in multiple particle environment (MPE) [33]: (1) *cooperative navigation*: N agents move as a team to cover N landmarks. Each agent observes its

Algorithm 1: Learning cooperative communication topology via meta-gradients

Input: A cooperative multi-agent task with agents $\mathcal{N}$, meta-optimization hyperparameter K

Initialize parameters for communication and policy networks $\theta_c \equiv \{\theta_c^i\}_{i\in\mathcal{N}}$, $\theta_\pi \equiv \{\theta_\pi^i\}_{i\in\mathcal{N}}$, centralized critic θ_Q, baseline ϕ, meta-parameters η, and replay buffer $\mathcal{D}$

repeat

  $\theta_c(0), \theta_\pi(0) \leftarrow \theta_c, \theta_\pi$

  for $k = 1, 2, ..., K$ **do**

    Generated a trajectory $\tau(k-1)$ using $(\theta_c(k-1), \theta_\pi(k-1))$

    Add the trajectory to replay buffer $\mathcal{D}$

    Update the centralized critic θ_Q by minimizing $\mathcal{L}(\theta_Q)$ in Equation (1) on samples from $\mathcal{D}$

    Update the policy networks as $\theta_\pi(k-1) \rightarrow \theta_\pi(k)$ using Equation (3) on samples from $\mathcal{D}$

    Update the communication networks as $\theta_c(k-1) \rightarrow \theta_c(k)$ using Equation (5) with η

  end

  Generated a trajectory $\tau(K)$ using $(\theta_c(K), \theta_\pi(K))$

  Perform updates for baseline ϕ on trajectories $\{\tau(k)\}_{k=1}^K$

  Update η using meta-gradient Equation (9) computed from trajectories $\{\tau(k)\}_{k=1}^K$

  $\theta_c, \theta_\pi \leftarrow \theta_c(K), \theta_\pi(K)$

until convergence

location and velocity, the relative location of the nearest l landmarks, and the relative location of other agents. Our experiments include the tasks of $(N = 3, l = 3)$ as `navigation_N3_l3`, $(N = 6, l = 1)$ as `navigation_N6_l1`, and $(N = 6, l = 6)$ as `navigation_N6_l6`. (2) *predator and prey*: N cooperating predators (agents) are tasked to capture M preys. The preys are environment-controlled by fixed policies that are pretrained. The movement of both predators and preys is impeded by L landmarks. Each predator observes its location and velocity, the relative location of the nearest l landmarks and the nearest l preys, and the relative location of other predators. Our experiments include the tasks of $(N = 3, M = 1, l = 3)$ as `predator_prey_N3_l3`, $(N = 6, M = 2, l = 1)$ as `predator_prey_N6_l1`, and $(N = 6, M = 2, l = 6)$ as `predator_prey_N6_l6`. (3) *cooperative push*: N cooperating agents are tasked to push a large ball to a target position. Each agent observes its location and velocity, the relative position of the large ball and the target position, and the relative location of other agents. We include the tasks of $N = 3$ as `push_N3` and $N = 6$ as `push_N6`.

Implementation details. Consistent with prior implementations [2, 34] on MPE, the trajectory length is set to be equal to the episode length, which is 25 timesteps. The meta-parameters are updated every $K = 10$ updates of the parameters, after a grid search over $K \in \{1, 5, 10, 20\}$. All algorithms are implemented using JAX [35].

Baselines. We compare our meta-gradient approach against the four baselines discussed in Section 4.1: no-communication, full-communication, learning communication topology with the policy gradient algorithm of REINFORCE, and learning communication topology with Straight Through Gumbel-Softmax (STGS). For the ablation study in Section 5.3, we additionally compare against variants of our algorithm with changes to our key design choices.

5.1 Effectiveness of the meta-learning framework

Figure 2 shows the training curves comparing our method of meta-learning communication topology against the four baselines. Please view all of the figures in this section in color. Our key observation is that: no method is uniformly the best among all tasks, yet our method is consistently comparable to the best for most tasks while each baseline can perform significantly worse in some tasks. Full communication is, perhaps surprisingly, among the worst in all tasks except for `push_N3`. No communication is among the worst in `push_N3` and worse than the best in `push_N6`. REINFORCE is among the worst in `navigation_N3_l3`, `predator_prey_N6_l1`, and `push_N3`. STGS is among the worst in all tasks except for `navigation_N6_l1` and `push_N3`. Our meta-learning method is consistently among the top two in all eight tasks.

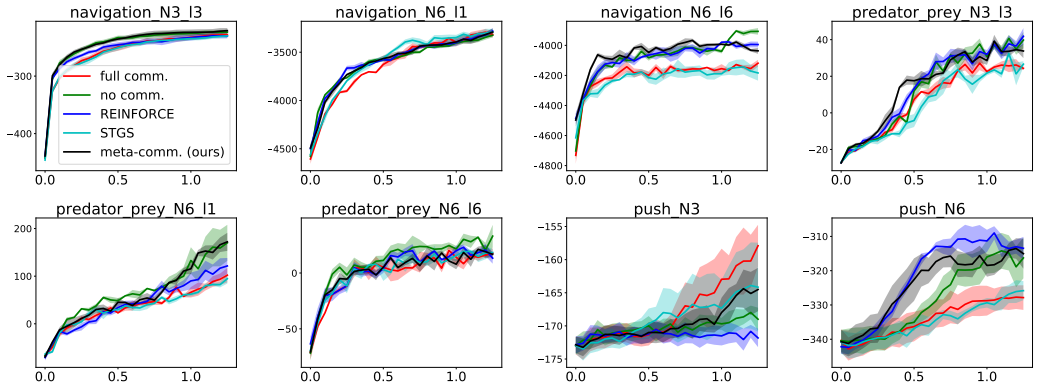

Figure 2: Training curves of three runs comparing our algorithm of meta-learning communication topology against the baselines. Y-axis: episodic reward. X-axis: training step (1e6).

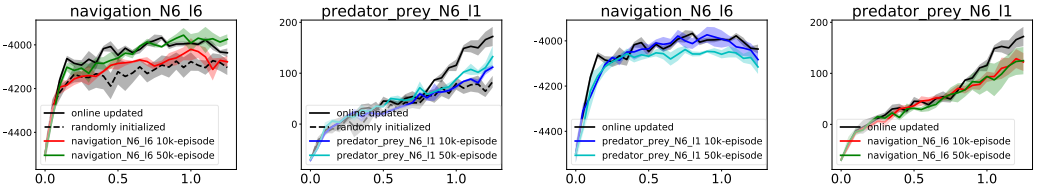

Figure 3: Effectiveness of fixed meta-parameters. Y-axis: episodic reward. X-axis: training step (1e6).

5.2 Effectiveness of the meta-learned update rule

Our algorithm continually updates meta-parameters η online. An interesting question is: how effective is the communication topology learning rule defined by some fixed η that is obtained after meta-gradient optimization? Figure 3 compares training curves for the two tasks of `navigation_N6_l6` and `predator_prey_N6_l1` using fixed η that is randomly initialized, or obtained after 10k or 50k episodes, with the online updated η as a reference point. We consider both the settings in which the fixed η is obtained from the same task and from a different task. As a sanity check, meta-learned η (either fixed or online updated) outperforms random initialized η. In `navigation_N6_l6`, the meta-learned, fixed η can outperform the online updated η, even for the η obtained from the other task of `predator_prey_N6_l1`. However, in `predator_prey_N6_l1`, the online updated η tends to perform best. This indicates that the answer to our question is task-dependent: some tasks may require continually updating meta-parameters for best performance.

5.3 Ablation study and analyses of learned communication topology

Our ablation study that examines the effect of our design choices of entropy regularization and intermediate meta-gradients introduced in Section 4.2, as well as the effect of the number of parameter updates K per meta-update Figure 4 (left) summarizes the ablation results . The results show that both design choices are crucial for performance. As for K, an intermediate value such as 5 and 10 tends to work better than the extreme values of 1 and 20.

To distinguish REINFORCE-learned and meta-learned typologies, we compare the REINFORCE advantage (i.e., $G_t - V(o_t;\phi)$ for Equation (4) with learned baseline ϕ) and the meta-learned advantage (i.e., $\hat{A}_t^{i\leftarrow j}$ computed by η for Equation (5)) that respectively define the two update rules for the communication network. Figure 4 (right) visualizes the normalized advantages computed from a same episode of `predator_prey_N6_l1`. We present the meta-learned advantage $\hat{A}_t^{i\leftarrow j}$ computed by both the randomly initialized η and the updated η after 50k episodes, for the agent indexed by $i = 1$ against its neighbors $j \in \{2, ..., 6\}$. It is clear that the meta-learned η yields advantages that differ from those at initialization and are contrastingly different among the neighbors j. Interestingly,

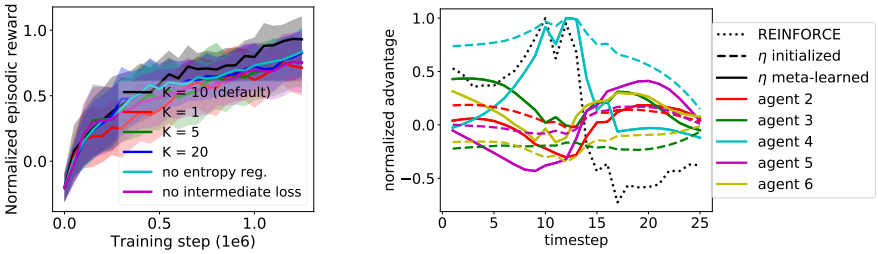

Figure 4: *Left*: Training curves for ablation study. For each task, we normalize its episodic reward to $[0, 1]$. *Right*: Visualization of the normalized REINFORCE and meta-learned advantages computed from an episode of `predator_prey_N6_l1`.

although most updates are significantly different, there is evidence that the meta-learned update rule occasionally resembles the REINFORCE update rule (see agent 4) for some trajectories.

6 Conclusion and discussion

In this paper, we have proposed a novel meta-learning framework to discover adaptive update rules for learning communication topology in an online manner. Empirical results showed that our method outperforms existing alternatives in a range of cooperative MARL tasks. Preliminary analyses suggest that the discovered update rules resemble the human-designed rules such as policy gradients, yet remain qualitatively different. We next acknowledge several limitations of this work, which point out promising future directions. Although we provided in Section 5 preliminary analyses on the topology learning update rules discovered by our meta-learning algorithm, its nature is not yet fully understood. It might be interesting and enlightening to better understand the discovered update rules for future work. Moreover, our meta-learning algorithm continually adapts the update rules online, and the meta-learned update rules exhibit limited generalization ability as shown in Figure 3. It remains an open question how we can discover topology learning update rules (defined by fixed meta-parameters) that work well for a large variety of MARL tasks.

References

[1] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*, 2017.

[2] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390, 2017.

[3] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in neural information processing systems*, pages 2244–2252, 2016.

[4] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Başar. Fully decentralized multi-agent reinforcement learning with networked agents. *arXiv preprint arXiv:1802.08757*, 2018.

[5] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems*, pages 2137–2145, 2016.

[6] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[7] Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. *arXiv preprint arXiv:1705.11192*, 2017.

[8] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018.

[9] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... hook*. Diplomarbeit, Technische Universität München, München, 1987.

[10] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Citeseer, 1990.

[11] Sebastian Thrun and Tom M Mitchell. Learning one more thing. In *International Joint Conferences on Artificial Intelligence*, 1995.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[13] Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*, 2018.

[14] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. *arXiv preprint arXiv:1804.06459*, 2018.

[15] Vivek Veeriah, Matteo Hessel, Zhongwen Xu, Richard Lewis, Janarthanan Rajendran, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. Discovery of useful questions as auxiliary tasks. *arXiv preprint arXiv:1909.04607*, 2019.

[16] Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. A self-tuning actor-critic algorithm. *arXiv preprint arXiv:2002.12928*, 2020.

[17] Zhongwen Xu, Hado van Hasselt, Matteo Hessel, Junhyuk Oh, Satinder Singh, and David Silver. Meta-gradient reinforcement learning with an objective discovered online. *arXiv preprint arXiv:2007.08433*, 2020.

[18] Junhyuk Oh, Matteo Hessel, Wojciech M Czarnecki, Zhongwen Xu, Hado van Hasselt, Satinder Singh, and David Silver. Discovering reinforcement learning algorithms. *arXiv preprint arXiv:2007.08794*, 2020.

[19] Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 32:4403–4414, 2019.

[20] 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*, pages 1538–1546. PMLR, 2019.

[21] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Efficient communication in multi-agent reinforcement learning via variance based control. *arXiv preprint arXiv:1909.02682*, 2019.

[22] Hangyu Mao, Zhibo Gong, Zhengchao Zhang, Zhen Xiao, and Yan Ni. Learning multi-agent communication under limited-bandwidth restriction for internet packet routing. *arXiv preprint arXiv:1903.05561*, 2019.

[23] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *In Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337. Morgan Kaufmann, 1993.

[24] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *arXiv preprint arXiv:1805.07733*, 2018.

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

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

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

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[29] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[30] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[31] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4805–4814, 2019.

[32] Qi Zhang, Richard Lewis, Satinder Singh, and Edmund Durfee. Learning to communicate and solve visual blocks-world tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5781–5788, 2019.

[33] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[34] Iou-Jen Liu, Raymond A Yeh, and Alexander G Schwing. Pic: permutation invariant critic for multi-agent deep reinforcement learning. In *Conference on Robot Learning*, pages 590–602. PMLR, 2020.

[35] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

A Implementation details

A.1 Computing resources

Our implementation is based on JAX [35] using GPUs. A single run took approximately 17 hours using an NVIDIA Tesla V100 GPU and 7 CPU cores.

A.2 Hyperparameters

Table 1: Hyperparameters for meta-learning η.

Hyperparameter	Value
Backward LSTM architecture (η)	LSTM(128)-FC(128)-Linear(output size)
Optimizer for η	RMSprop with learning rate 0.00005
Gradient clipping norm for η	FC(128)-FC(128)-Linear(1)
Baseline~architecture (ϕ)	FC(128)-FC(128)-Linear(1)
Optimizer for ϕ	Adam with learning rate 0.001
Entropy regularization (β)	0.01
Number of parameter updates (K)	10

B Additional results

The main body presents the visualization of normalized advantages for a single episode of `predator_prey_N6_l1`. Figure 5 presents visualization for multiple episodes.

Table 2: Hyperparameters for learning θ. These hyperparameters are shared by our method and the baselines.

Hyperparameter	Value
Episode length	25
Trajectory length	25
Number of training episodes	50000
Discount factor	0.95
Critic network architecture	-FC(128)-FC(128)-Linear(1)
Critic network optimizer	Adam with learning rate 0.001
Policy network architecture	Concat[$o(\mathcal{C}^i)$-LSTM(128), o^i]-FC(128)-FC(128)-Linear(action dim)
Policy network optimizer	Adam with learning rate 0.001
Communication network architecture	-FC(128)-FC(128)-Linear(output size)
Communication network optimizer	RMSprop with learning rate 0.001
Target network Polyak averaging rate	0.01
Number of samples from replay buffer	256

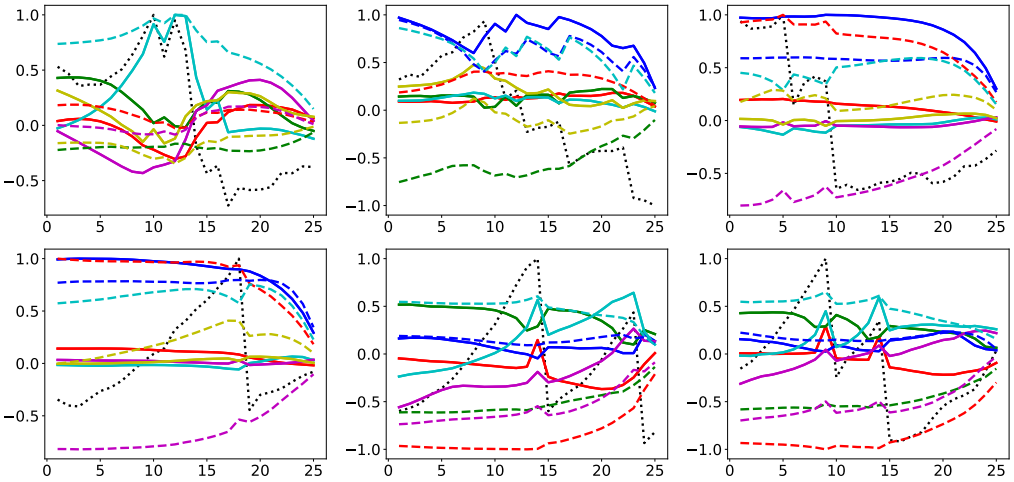

Figure 5: Visualization of the normalized REINFORCE and meta-learned advantages computed from multiple episodes of `predator_prey_N6_l1`.