Continuous self-adaptive optimization to learn multi-task multi-agent

Wenqian Liang^{1,2} · Ji Wang¹ · Weidong Bao¹ · Xiaomin Zhu¹ · Qingyong Wang¹ · Beibei Han¹

Received: 14 June 2021 / Accepted: 11 November 2021 / Published online: 17 December 2021
© The Author(s) 2021

Abstract

Multi-agent reinforcement learning (MARL) methods have shown superior performance to solve a variety of real-world problems focusing on learning distinct policies for individual tasks. These approaches face problems when applied to the non-stationary real-world: agents trained in specialized tasks cannot achieve satisfied generalization performance across multiple tasks; agents have to learn and store specialized policies for individual task and reliable identities of tasks are hardly observable in practice. To address the challenge continuously adapting to multiple tasks in MARL, we formalize the problem into a two-stage curriculum. Single-task policies are learned with MARL approaches, after that we develop a gradient-based Self-Adaptive Meta-Learning algorithm, SAML, that cannot only distill single-task policies into a unified policy but also can facilitate the unified policy to continuously adapt to new incoming tasks. In addition, to validate the continuous adaptation performance on complex task, we extend the widely adopted StarCraft benchmark SMAC and develop a new multi-task multi-agent StarCraft environment, Meta-SMAC, for testing various aspects of continuous adaptation method. Our experiments with a population of agents show that our method enables significantly more efficient adaptation than reactive baselines across different scenarios.

Keywords Multi-task · Multi-agent · Meta-learning · Reinforcement learning · Self-adaptive optimization

Introduction

Recent progress in multi-agent reinforcement learning (RL) has achieved impressive performance on many complex tasks [1–4]. Despite the progress, these works mostly have

been designed to deal with a single-task RL with stationary environments which face problems when applied to the non-stationary real-world [2]. Agents trained in specialized tasks cannot achieve satisfied generalization performance across multiple tasks. When solving a variety of real-world problems with a set of specialized policies, agent have to store identities of individual task to utilize the appropriate specific-task policy according to the environment. Unfortunately, reliable identities of tasks are hardly observable in practice.

Rather than focus on a single task, we argue that agent should be able to solve multiple tasks with various sources of reward. Recent work in multi-task RL has attempted to address this by achieving generalized performance across all tasks [5]. MTRL methods often focus on the tasks which are explicitly described with natural language instruction, programs, or graph structures [6,7]. However, such explicit knowledge descriptions and reliable identities of tasks may not readily be available.

Existing MTRL methods usually focus on multi-task single-agent tasks [8–11]. They uses the multi-task learning method to generalize on all original tasks by sharing representations between related tasks. A more flexible approach

W. Liang and J. Wang contributed equally to this work.

✉ Weidong Bao
wdbao@nudt.edu.cn

Wenqian Liang
wenqian.liang@nudt.edu.cn

Ji Wang
wangji@nudt.edu.cn

Xiaomin Zhu
xmzhu@nudt.edu.cn

Qingyong Wang
wang@cnu.ac.cn

Beibei Han
hanbei969@163.com

¹ College of Systems Engineering, National University of Defense Technology, Changsha 410000, China

² North Automatic Control Technology Institute, Taiyuan 030000, China

is meta-RL which has the agents implicitly inferring the task by interacting with the environment and quickly adapt to them [12–14]. However, they have focused on relatively simple tasks with single agent and/or fully observable settings. By contrast, we proposed a gradient-based Self-Adaptive Meta Learning method called SAML which combined meta-learning with MARL approach to deal with the problem continuously adapting to multi-task multi-agent tasks.

SAML proposes a new MT-MARL framework and decomposes the training process into single-task phase and multi-task phase (Fig. 1). The first phase uses centralized training with decentralized execution (CTDE) paradigm to learn distinct policy for single-task MARL. This first phase enables the same Deep Recurrent Q-Network (DRQN) to control all the agents in single-task MARL. The input and output of scenarios are reconstructed into a unified form with same dimension so that specific DRQN of the single-task in first phase can be used in meta-learning process of second phase. Each task-specific DRQN utilizes their policy to interactive with environment to collected supervised data for meta-learning in phase II. The second phase of our approach distills each specialized action-value DRQN of single-task into a generalized recurrent multi-task DRQN with maximizing transfer and minimizing interference between multiple tasks. Existing benchmarks mainly focus on relatively simple agent in multiple tasks (e.g., Meta-World [15]) or multi-agent learning in a single-task scenario (e.g., SMAC [16]), which cannot meet our experimental needs. To validate the continuous adaptation performance on complex MT-MARL task, we develop a meta-learning experimental environment based on a widely adopted StarCraft II benchmark SMAC. As far as we know, this is the first work to validate the continuous adaptive performance of multi-agent in complex StarCraft II multi-task scenarios.

The main novelties of this paper are summarized as follows:

- A novel two-phase approach for continuous adaptation on multi-task MARL is provided, which first conducts single-task MARL process to achieve task-specific policy, and then distills task-specific policies into a unified DRQN that performs well on all tasks without explicit provision of task ID.
- We analyze the Transfer-Interference Trade-off problem and propose a self-adaptive meta-learning method, SAML, to enhance and facilitate MT-MARL on phase II with maximizing transfer and minimizing interference between multiple tasks.
- To validate our proposed method, we extend the widely adopted StarCraft benchmark SMAC and develop a new multi-task multi-agent StarCraft environment, Meta-SMAC, to test the various aspects of continuous adaptation on complicated MT-MARL. Our experiments with a

population of agents show that our method enables significantly more efficient adaptation than reactive baselines, such as FOMAML and REPTILE, across different scenarios.

The rest of this paper is organized as follows. We first introduce the basic MARL theories and some typical methods of MARL and meta learning in the next section. In the third section, we present the two-phase continuous adaptation framework in details and the gradient-based Self-Adaptive Meta-Learning method. Main experimental results and ablation studies are shown in the fourth section. Then, the related work are given in the fifth section. We conclude the paper in the last section.

Background

In this section, the basic theories and important concepts in MT-MARL are introduced. We also present some typical MARL and meta learning methods that aim at learning a generalized centralized value function with the CTDE framework.

Multi-agent RL

We consider a fully cooperative multi-agent setting involving a set of agents in a shared environment. MARL methods are separated into two classes: independent learners and joint action learners [17]. Independent learners is to learn the individual agent's action-value function independently, i.e., independent Q-learning, whereas joint action learners observe joint-actions taken by all agents. Recently, Centralized Training with Decentralized Execution (CTDE) paradigm is employed to train MARL agents especially with only partial observation [18]. CTDE allows agents to train decentralized policies with the global information while training and to make decisions based on the individual learned policies while executing. Recent works including VDN and QMIX employ CTDE to utilize optimization at the individual action-value functions to optimize of the joint action-value function. VDN [19] represents a central action-value function as a sum of individual action-value function. QMIX [3] represents joint action-value function with a large family of monotonic non-linear per-agent action-value function rather than just as a sum of them. VDN's representation is sufficient to satisfy Eq. 1. QMIX extends VDN's representation and satisfies Eq. 3 when Eq. 2 is satisfied:

$$Q_{\text{tot}}(\boldsymbol{\tau}, \mathbf{u}) = \sum_{i=1}^n Q_i(\tau^i, u^i). \quad (1)$$

Fig. 1 Overview of self-adaptive meta-learning framework to multi-task multi-agent tasks. The continuous adaptation is formalized into a two-phase approach. The approach first conducts single-task MARL process to achieve task-specific policy. The input and output of scenarios are reconstructed into a unified form with same dimension so that specific DRQN of the single-task in first phase can be used in meta-learning process of second phase. Each task-specific DRQN utilizes their policy to interactive with environment to collect supervised data for meta-learning in phase II. Then, in phase II, we distill task-specific policies into a unified policy with our proposed gradient-based adaptive meta-learning method (SAML)

$$\frac{\partial Q_{\text{tot}}}{\partial Q_i} \geq 0, \quad \forall i \in A. \quad (2)$$

$$\max_u Q(\tau, \mathbf{u}) \\ = Q_{\text{tot}} \left(\tau, \arg \max_{u^1} Q_1(\tau^1, u^1), \dots, \arg \max_{u^N} Q_N(\tau^N, u^N) \right). \quad (3)$$

Multi-task multi-agent RL

Multi-task multi-agent RL (MT-MARL) aims to learn a policy that performs well on a set of related source MARL tasks to a target task. MT-MARL is beneficial when source tasks share common features [20], and challenging when the task description is not explicitly observed to agents during execution [8]. The MT-MARL problem in partially observation extends the single-agent fully observable setting [21]. A partially observable MT-MARL Task T_j is a tuple $\langle P_j, S_j, R_j, D \rangle$, where P_j, S_j, R_j are, respectively, the state transition function, observation functions and reward. MT-MARL Domain D is partially observable and denoted as a tuple $\langle A, S, U, O, \gamma \rangle$, where A is the set of agents, S is the environment state space, U is the joint action space, O is the joint observation space, and γ is the discount factor. In MT-MARL, task T_j is sampled from domain D and is consisted of episode $e \in \{1, \dots, E\}$. The MT-MARL agents can observe the task ID, $j \in \{1, \dots, J\}$ during learning, but not while executing. The objective is to find a optimal joint policy to achieve maximal reward in all E episodes, $R^* = \frac{1}{J} \frac{1}{E} \sum_{j=1}^J \sum_{e=1}^E \gamma^t R_e(s_t, a_t)$ [21].

Meta-reinforcement learning

Recent meta-RL approaches contain two broad categories: popular gradient-based meta-learners [22–24] and RNN-based meta-learners [25]. Gradient-based meta-learners aims to learn a good initialization of agent's policy network by taking gradient steps and enable agents rapidly adapt to an unseen task, such as MAML [13] and REPTILE [23]. RNN-based meta-RL methods complete process of adaptation by updating the hidden states of a RNN [25]. In addition, other variants of RNN-based meta-learner have been explored, such as temporal convolutions [26]. Our approach has referred to the first category, but different from these works as we combine the meta-RL approaches with existing MARL methods and consider the transfer-interference trade-off problem.

Methods

Figure 1 overviews our approach with two-phase curriculum to address the MT-MARL problems. Our main idea is to

employ two policies: task-specific policy and adaptation policy. During the phase I, we first conduct single-task MARL process to achieve task-specific policy. In phase II, we distill task-specific policies into a adaptation policy that performs well in all tasks with collected trajectories produced by task-specific policies.

Phase I: Dec-POMDP single-task MARL

As Fig. 1 shows, Dec-POMDP single-task MARL algorithm of phase I has an architecture consisting of agent networks, a mixing network [27] and a meta-experience transition module. All agents share the same agent network and use it to generate the action policy. The mixing network is a feed-forward neural network that used for central training in CTDE paradigm. The meta-experience transition module is used to transform the input and output of all tasks and make them have same dimension.

Figure 2 illustrates the learning process of Single-Task MARL of phase I. Each agent i represents its individual action-value function $Q_i(\tau_i, u_i)$ as a DRQN named $Q_network_local$. After receiving state s_t at each time step, a random action u_t is executed with probability ϵ , otherwise action is produced by $Q_network_local$ with ϵ -greedy policy. We execute action u_t to StarCraft II environment, and then receive reward r_t and next state s_{t+1} . These experiences are stored in replay memory and will be sampled for mixing network to produce the joint action-value function Q_{tot} . With the joint Q -value produced by mixing network and the target Q -value produced by $Q_network_target$ network, we can compute MSELoss and perform a gradient descent step to update the parameters of $Q_network_local$. $Q_network_target$ will be reset with the weights of $Q_network_local$ every C iterations until the satisfied policy is learned.

The structure of mixing network is shown in Fig. 2. The mixing network is a feed-forward neural network with non-negative weights generated by separate hyper-networks which take the additional states as input [28]. The agent network consists of a DRQN with parameter θ , which represent the individual agent's action-value function Q . After sampling batch of experiences from the replay memory, MSELoss is computed with the joint Q -value with double DQN paradigm. We can learn the parameter θ by performing a gradient descent step to learn the parameters of $Q_network_local$ (Eq. 4):

$$L(\theta) = \sum_{i=1}^n [(y_i^{\text{tot}} - Q_{\text{tot}}(\tau, u, s; \theta))^2]. \quad (4)$$

The meta-experience transition module is used to transform the input and output of the task. Note that the state space is different between these tasks, therefore, the transitions

Fig. 2 Overview of Dec-POMDP single-task MARL in phase I. The input and output of scenarios are reconstructed into a unified form with same dimension with meta-experience module so that action-value DRQN of the single-task in first phase can be used in meta-learning pro-

cess of second phase. The framework of single-task MARL is similar to QMIX algorithm, and details of learning process is illustrated in section of method

sampled from replay memory cannot be used to compute the MSE Loss directly. Therefore, we use meta-experience transition module to reconstruct the samples into a unified form with same dimension. We define a formal semantic mapping function $\phi(\cdot)$ to indicate the mapping between different state spaces.

Definition 1 (Semantic Mapping Function) Given three tasks τ_1 , τ_2 and τ_3 with different state space dimensions, if state s_{τ_1} and s_{τ_2} contain similar semantic information while s_{τ_3} does not. Through the mapping function $\phi(\cdot)$ to transform the state dimension, there exists a state space to make the following Inequation establish: $dis(\phi(s_{\tau_1}), \phi(s_{\tau_2})) < dis(\phi(s_{\tau_1}), \phi(s_{\tau_3}))$, $dis(\phi(s_{\tau_1}), \phi(s_{\tau_2})) < dis(\phi(s_{\tau_2}), \phi(s_{\tau_3}))$, where $dis(\cdot)$ is the distance between two vectors [29].

By the Definition 1, we can map the different states of each task τ_i into the same semantic state space. This concept is often used in domain adaptation area [30,31]. If states of different tasks have the same semantics with different dimensions and can be transformed into same latent state space, then we can easily transfer knowledge between these tasks, e.g., adding zero padding for those samples from small-scale environment or exchanging similar positions in the same latent state space.

Phase II: Dec-POMDP multi-task MARL

The second phase is used to distill each DRQN of single task into a unified DRQN that can perform well over all tasks without explicit provision of task ID. Actually, once task-specific DRQN is conducted for each task, multi-task MARL can be formulated as a regression problem over Q -values, which can be resolved by continuously conducts data collection and regression.

For data collection, agents use each task-specific DRQN (from Phase I) to receive the states and execute actions in corresponding environment. The episode data of regression experiences (s_t, u_t) is stored in shared replay memory and used to compute cross entropy loss of unified DRQN (referring to Algorithm 1). To improve the performance, we also concern the transfer and interference on the multiple tasks. As shown in Fig. 3, two arbitrary distinct examples (s_i, a_i) and (s_j, a_j) are trained with SGD.

Transfer occurs when Eq. 5 is satisfied which will enhance the weight sharing on two neural networks, while interference occurs when Eq. 6 are satisfied which will attenuate the weight sharing.

$$\frac{\partial L_i}{\partial \theta} \cdot \frac{\partial L_j}{\partial \theta} > 0, \quad (5)$$

Algorithm 1: MT-MARL SAML algorithm

```

1 Achieve action-value function  $Q_i$  of task-specific DRQN from
  phrase  $I$ ;
2  $p(T_i)$  is the distribution over tasks ;
3 Initialize action-value function  $Q$  of the unified DRQN with
  random weights  $\theta$ ;
4 Initialize step size hyper-parameters  $\alpha, \beta, \gamma$ ;
5 Initialize replay memory  $D$  to capacity  $M$ ;
6 while not done do
7   // Produce supervised data for each task;
8   for  $T_i = 1, \dots, n$  do
9     for  $t = 1, \dots, k$  do
10      Select  $a_t = \arg \max_a Q_i(s_t, a_{t-1}, \theta_i)$ ;
11      Store step transition  $(s_t, a_t)$  in episode data
12    Store episode data in  $D$  and update replay memory;
13 Sample batches  $((B_1, \dots, B_s))$  of tasks from replay memory  $D$ ,
  each batch has  $k$  episodes and each episode only has data of a
  single task;
14  $\theta_0^A \leftarrow \theta$ ;
15 for  $B_i = 1, \dots, s$  do
16    $\theta_{i,0}^W \leftarrow \theta$ ;
17   for  $E_j = 1, \dots, k$  do
18     Compute Cross Entropy Loss and its gradient ;
19      $\theta_{i,j}^W \leftarrow SGD(s_t, a_t, \theta_{i,j-1}^W, \alpha)$ 
20   // Within batch Reptile meta-update:
21    $\theta \leftarrow \theta_{i,0}^W + \beta(\theta_{i,k}^W - \theta_{i,0}^W), \theta_i^A \leftarrow \theta$ ;
22   // Across batch Reptile meta-update:
23    $\theta \leftarrow \theta_0^A + \gamma(\theta_s^A - \theta_0^A)$ ;

```

$$\theta = \arg \min_{\theta} E_{[(s_i, a_i), (s_j, a_j)] \sim D} [L(s_i, a_i) + L(s_j, a_j) - \alpha \frac{\partial L_i}{\partial \theta} \cdot \frac{\partial L_j}{\partial \theta}] \quad (8)$$

Considering in terms of gradients, REPTILE algorithm approximately optimizes its objective on a set of s batches [23]:

$$\theta = \arg \min_{\theta} E_{[B_1, \dots, B_s] \sim D} \times \left[2 \sum_{i=0}^s \left[L(B_i) - \sum_{j=1}^{i-1} \alpha \frac{\partial L(B_i)}{\partial \theta} \cdot \frac{\partial L(B_j)}{\partial \theta} \right] \right], \quad (9)$$

where $B_1, \dots, B_s$ are batches within D . We find that it is similar to our motivation in Eq. 8, the difference is that Eq. 8 both computes gradients produced within and across these batches. Therefore, we express our optimization objective in Eq. 10:

$$\theta = \arg \min_{\theta} E_{[(s_{11}, a_{11}), \dots, (s_{sk}, a_{sk})] \sim D} \left[2 \sum_{i=1}^s \sum_{j=0}^k \left[L(s_{ij}, a_{ij}) - \sum_{q=1}^{i-1} \sum_{r=1}^{j-1} \alpha \frac{\partial L(s_{ij}, a_{ij})}{\partial \theta} \cdot \frac{\partial L(s_{qr}, a_{qr})}{\partial \theta} \right] \right]. \quad (10)$$

We sample s batches from D , and each batch has k examples. Further details for SAML is provided in Algorithm 1.

Fig. 3 A depiction of transfer and interference in weight space

$$\frac{\partial L_i}{\partial \theta} \cdot \frac{\partial L_j}{\partial \theta} < 0, \quad (6)$$

where $(\cdot)$ is the dot product operator. Maximizing weight sharing will maximize the potential for transfer, while minimizing weight sharing will minimize the potential for interference. In typical offline supervised learning, we can optimize for the following objective over the stationary distribution of (s, a) pairs within the data-set D :

$$\theta = \arg \min_{\theta} E_{(s,a) \sim D} [L(s, a)]. \quad (7)$$

To maximize the potential for transfer and minimize the potential for interference, we can express our optimization objective as follows:

Experimental evaluation

The experiment is separated into a two-stage curriculum. Single-task MARL process is first conducted to achieve task-specific policy, and task-specific policies are distilled into a unified DRQN that performs well in all tasks without explicit provision of task ID.

Environment

We focus on the continuous adaptation of multiple multi-agent tasks. However, the existing benchmarks mainly focus on simple single agent in multiple tasks (e.g., Meta-World [15]) or multi-agent learning in single-task scenario (e.g., SMAC [16]), which cannot meet our experimental needs. Therefore, we extend the widely adopted StarCraft benchmark SMAC and develop a new multi-task multi-agent StarCraft environment, Meta-SMAC, for testing various aspects of continuous adaptation. We reconstruct the input and output of scenarios into a unified form with same dimension so that action-value DRQN of the single-task can have same dimension.

We design three experimental scenarios, Homogeneous scenario, Heterogeneous scenario and Mixed scenario, to

Table 1 The details of three multi-task scenario

Scenario name	Map contained in each scenario
Homogeneous scenario	10m, 10m_vs_11m, 6m_vs_7m, 7m_vs_8m, 8m_vs_9m, 9m, 9m_vs_10m and 5m_vs_6m
Heterogeneous scenario	2s3z, 2s3z_vs_2s4z, 2s4z_vs_2s5z, 2s5z_vs_2s6z, 3s4z_vs_3s5z, 3s5z_vs_3s6z and 1s5z_vs_1s6z
Mixed scenario	10m_vs_11m, 1c3s5z, 3s4z_vs_3s5z, 3s5z, 5m_vs_6m, 9m_vs_10m, MMM2 and 1s5z_vs_1s6z

Fig. 4 Maps included in heterogeneous scenario. Figure 4d shows the map 2s4z_vs_2s5z is an asymmetric scenario in heterogeneous scenario that requires the 2 allied stalker and 4 allied zealots (left) to defeat the 2 enemy stalker and 5 enemy zealots (right), that is, we use 6 allied game unit to defeat 7 enemies. Multiple agents require a range of coordination skills such as focusing fire and avoiding overkill to attack the enemy

units and win the game. Figure 4e shows the map 2s5z_vs_2s6z that requires 7 allies to defeat 8 enemies. The map 2s4z_vs_2s5z and map 2s5z_vs_2s6z are regarded as different tasks and task-specific DRQN on map 2s4z_vs_2s5z achieve poor performance on map 2s5z_vs_2s6z (validated in Table 4)

test the performance of the algorithms. The homogeneous scenario contains only one kind of game unit, while the heterogeneous and mixed scenarios contain multiple game units, in order that three scenarios have different difficulties to get generalized performance. They all contain eight micromanagement maps, for example, heterogeneous scenario is shown in Table 1. Figure 4d shows the map 2s4z_vs_2s5z is an asymmetric scenario in heterogeneous scenario that requires the 2 allied stalker and 4 allied zealots (left) to defeat the 2 enemy stalker and 5 enemy zealots (right), that is, we use 6 allied game unit to defeat 7 enemies. At the training of each episode, multiple agents with partial observation require a range of coordination skills such as focusing fire and avoiding overkill to attack the enemy units and win the game. In Sect. 4.4.1, seven maps were used to test the adaptation performance. Map 5m_vs_6m and 1s5z_vs_1s6z are assigned to test the ability of continuous adaptation performance to a new task in Sect. 4.5.

Baselines and experimental settings

Baselines. We compare to three baselines to test the performance of our algorithm as following: (i) Multi-Task Learning: an algorithm which enables unified model to generalize on all original tasks by sharing representations between related tasks [32] with computing cross entropy loss. (ii) REPTILE is a meta-learning method with comput-

ing first-order gradient information [23]. (iii) FOMAML: a meta-learning algorithm ignoring the second derivative terms at the expense of losing some gradient information [23].

Experimental settings. Phase I is a MARL process, while phase II is a meta-learning process. In phase I, the training will be paused after every 10 thousand time steps, meanwhile, 32 test episodes will run to evaluate the performance of the algorithm with test win rate. In phase II, the training will be paused after 128 episodes, meanwhile, 32 test episodes will run to evaluate the performance of the algorithm. We measure the performance in terms of mean win rate averaged over all the tasks in each scenarios shown in Table 1. Test win rate is the percentage of test episodes in which the trained agents defeat enemy units within the permitted time limit. All the resulting plots contain the median mean performance as well as the 25–75% percentiles over 5 runs with different seeds. Mean win rate is used to evaluate the mean performance of all the maps in the scenarios in phase II. All the hyperparameters are shown in Table 2. The state, action and reward setting are the same as SMAC and can be referred to SMAC's appendix [16]).

Single-task performance validation

In phase I, we first conduct single-task MARL process to achieve task-specific policy. The task-specific DRQN model

Table 2 Hyper-parameters used in experiment

Hyper-parameter		Value
Phase I	Batch-size	32
	Replay memory size	5000
	Test episode	32
	Discount factor	0.99
	Optimizer	RMSProp
	Learning rate	0.0005
	Target-update-interval	200
	Double_q	True
	Mixing_network_embed_dim	32
	Mixing_network_hypernet_layers	2
Phase II	Agent_rnn_hidden_dim	64
	Evaluate episode	32
	Optimizer	RMSProp
	α (inner loop, learning rate)	0.005
	β (outer loop, within batch)	0.2
	γ (outer loop, across batch)	0.2
	Agent_rnn_hidden_dim	64
	Agent_rnn_hidden_dim_general	64
	Observation_input_shape	337
	Action_output_shape	18

is trained until the median win rate is above 94%, as shown in Table 3. To illustrate the necessity of the general model, we also verify the median win rate of task-specific DRQN model on other maps in the same scenario. The results of heterogeneous scenario is shown in Table 4. As the first row of Table 4 shown, median win rate of task-specific model of map 10m_vs_11m is about 97%. However, when we test the DRQN model of map 10m_vs_11m on other maps in mixed scenario, the median win rate of map 9m_vs_10m is relatively high, but in other maps, the median win rate is 0.

Therefore, the task-specific DRQN model has no ability of generalization.

Multi-task performance validation

In phase II, we validate the generalization performance of SAML algorithm in multi-task scenarios and its continuous adaptation performance when a new map is added.

Generalization performance validation

The generalization performance of the DRQN model is tested in homogeneous, heterogeneous and mixed scenarios. The homogeneous scenario contains only one kind of game unit, while the heterogeneous and mixed scenarios contain multiple game units. Figure 5a–c shows the training results of SAML, FOMAML, REPTILE and MTL in 10,000 epochs. These three scenarios contain 7 single tasks, as shown in Table 1. In these three scenarios, SAML algorithm achieves the highest mean win rate, about 90% in homogeneous scenario, 79% in heterogeneous scenario, and 88% in mixed scenario. MTL method has the lowest median win rate because it only uses the multi-task regression method instead of meta-learning paradigm. All the algorithms with meta-learning consistently outperform MTL algorithm across all the scenarios, showing that meta adaptation policy can efficiently improve the generalization performance. In addition, SAML algorithm performs better than FOMAML and REPTILE baselines in all the scenarios, showing that considering the transfer and interference trade-off while training multiple tasks is more effective than normalized meta-learning. The median win rate of SAML, FOMAML, REPTILE and MTL in mixed scenario is shown in Table 4. SAML algorithm can get relatively high performance on all tasks of mixed scenario, therefore, it DRQN model is a generalized model that can deal with MT-MARL tasks without provision of task entities.

Table 3 Median win rate of models trained by single-task MARL in phase I

Homogeneous scenarios		Heterogeneous scenarios		Mixed scenarios	
Map name	Median win rate (%)	Map name	Median win Rate (%)	Map name	Median win rate (%)
10m	100	2s3z	100	10m_vs_11m	98
10m_vs_11m	98	2s3z_vs_2s4z	97	1c3s5z	98
6m_vs_7m	94	2s4z_vs_2s5z	95	3s4z_vs_3s5z	95
7m_vs_8m	95	2s5z_vs_2s6z	96	3s5z	100
8m_vs_9m	90	3s4z_vs_3s5z	94	5m_vs_6m	95
9m	100	3s5z	99	9m_vs_10m	95
9m_vs_10m	95	3s5z_vs_3s6z	95	MMM2	97
5m_vs_6m	95	1s5z_vs_1s6z	94	1s5z_vs_1s6z	94

Table 4 Median win rate of trained model while tested on each task of heterogeneous scenario

Map name	Median win rate when tested on all maps of heterogeneous scenario (%)							Mean win rate (%)
	2s3z	2s3z_vs_2s4z	2s4z_vs_2s5z	2s5z_vs_2s6z	3s4z_vs_3s5z	3s5z	3s5z_vs_3s6z	
2s3z	100	0	0	0	0	0	0	14
2s3z_vs_2s4z	16	97	0	0	0	3	0	16
2s4z_vs_2s5z	16	0	94	15	6	37	0	24
2s5z_vs_2s6z	0	0	0	97	0	34	0	19
3s4z_vs_3s5z	13	0	0	9	91	91	0	29
3s5z	41	0	0	0	3	100	0	21
3s5z_vs_3s6z	3	0	0	0	0	19	94	17
FOMAML	91	65	64	62	56	92	5	62
REPTILE	92	64	66	63	54	91	11	63
MTL	92	9	3	13	0	59	0	25
SAML	100	88	69	78	72	97	52	79

Fig. 5 Adaptation performance validation on three scenarios. **a–c** show the training results of SAML, FOMAML, REPTILE and MTL in 10,000 epochs. In these three scenarios, SAML algorithm achieves the highest mean win rate, about 90% in homogeneous scenario, 79% in heterogeneous scenario, and 88% in mixed scenario. All the algorithms with meta-learning consistently outperform MTL algorithm across all the scenarios, showing that meta-adaptation policy can efficiently improve the generalization performance. In addition, SAML algorithm performs better than FOMAML and REPTILE baselines in all the scenarios,

showing that considering the transfer and interference trade-off while training multiple tasks is more effective than normalized meta-learning. In **d**, SAML algorithm achieves the mean win rate of 91% and the new task's (map 5 m_vs_6m) the median win rate is about 86%. As **e** and **f** shows, SAML also achieves the optimal performance in heterogeneous and mixed scenarios. Our experimental results on StarCraft II environments show that the proposed method is able to effectively learn a unified generalization policy, and can adapt more efficiently to unseen tasks than existing baselines across different scenarios

Fig. 6 Effects on network capacity. The capacity of generalized DRQN model is set to 64, 128 and 256, respectively, such that we can train the DRQN model with respect to different capacity to measure how much it is likely to increase the mean win rate of the scenarios. Under the

same conditions, through 10,000 epochs of training, the mean win rate of the DRQN model with the capacity of 256 achieve 94, 92 and 93 in homogeneous, heterogeneous and mixed scenarios, respectively, which is close to the maximum mean win rate shown in Table 5

Table 5 Median win rate of trained models with different capacity

Scenario name	Mean win rate of different capacity (%)			Max mean win rate (%)
	dim_64	dim_128	dim_256	
Homogeneous scenario	90	93	94	96
Heterogeneous scenario	79	90	92	96
Mixed scenario	88	91	93	96

Continuously adaptation performance validation

We test the continuous adaptation performance when a new map is added to the experimental scenarios. As shown in Table 1, we add map 5m_vs_6m to Homogeneous scenario and map 5m_vs_6m map to the heterogeneous scenario and mixed scenarios. Figure 5d–f shows the mean win rate of the three scenarios and median win rate of the new map (map 5m_vs_6m and map 1s5z_vs_1s6z). As Fig. 5e, f shows, SAML achieves the optimal performance of continuously adaptation on all three scenarios. SAML algorithm achieves about 91% in homogeneous scenario, 81% in heterogeneous scenario, and 89% in mixed scenario. Our experimental results on StarCraft II environments show that the proposed method is able to effectively learn a unified generalization policy, and can adapt more efficiently to unseen tasks than existing baselines across different scenarios.

Effects on network capacity

To invalidate the effect of SAML method, the capacity of the unified DRQN in phase II is set 64 that is the same as the single-task DRQN in phase I. In Fig. 5, we found that SAML does not get sufficiently satisfactory generalized performance. We think generalized performance is limited on network capacity of unified DRQN network. In this section,

we discuss the effect of the size of the network structure on the phase of learning a unified generalization policy. The capacity of generalized DRQN model is set to 64, 128 and 256, respectively, such that we can train the DRQN model with respect to different capacity to measure how much it is likely to increase the mean win rate of the scenarios. As shown in Fig. 6, with the increase of the capacity of the DQRN model, the generalization performance of the model is significantly enhanced. Under the same conditions, through 10,000 epochs of training, the mean win rate of the DRQN model with the capacity of 256 achieve 94, 92 and 93 in homogeneous, heterogeneous and mixed scenarios respectively, which is close to the maximum mean win rate shown in Table 5. The results of Fig. 6 indicate that SAML is a feasible solution which can deal with complex MT-MARL problems. Through the application of high-capacity unified DQRN network, SAML has the potential to solve large-scale MT-MARL problems.

Related work

Recent progress in MARL has achieved impressive performance since the representative value-based MARL method, QMIX [3], is proposed. Plenty of outstanding MARL methods, such as Qatten [33], WQMIX [34], QPLEX [35] and ROMA [36], are improved based on QMIX. They all claim

that they have outperformed QMIX in their experiments. However, as rethinking the importance of implementation tricks in multi-agent reinforcement learning, a new research work [37] demonstrates that experimental results of these works seem to highly depend on “implementation tricks”. After minimal tuning, QMIX can attain extraordinarily high win rates and achieve SOTA in the StarCraft Multi-Agent Challenge (SMAC). Though QMIX is a powerful MARL method to solve the single-task MARL tasks, we found that it cannot achieve satisfied generalization performance across MT-MARL tasks. When solving a variety of real-world problems with a set of task-specific policies produced by single-task MARL method, agent has to store identities of all individual tasks to utilize the appropriate specific-task policy according to the specific condition of environment. Unfortunately, reliable identities of tasks are hardly observable in practice. Therefore, it is necessary to deal with multi-task problems without considering their identities.

There are a number of types of methods can achieve the adaption from a learnt task to others. Distral [9] is a method allowing effective sharing a “distilled” policy that captures common behavior across tasks. Each worker is trained to solve its own task while constrained to stay close to the shared policy. Therefore, knowledge gained in one task is distilled into the shared policy, then transferred to other tasks implicitly. AATEAM [10] uses the attention-based neural networks to cope with new teammates’ behavior in real-time. AATEAM trains the attention network to measure the similarity between the corresponding past teammates and new teammates, which helps to adjust the action selection in real time. MvDAN [11] is a novel multi-view deep attention network, which approximates multiple view-specific policy in parallel and integrates these functions based on attention mechanisms to generate a comprehensive strategy. MvDAN can exploit the specific statistical property of each view to learn a more comprehensive representation than single-view representation learning, which can reduce redundancy between multiple views and improve the generalization performance. In contrast to these methods, we propose to utilize the meta-learning paradigm to enhance and facilitate MT-MARL process with maximizing transfer and minimizing interference between multiple tasks.

As we know, meta-RL is a flexible approach which can implicitly infer the multiple tasks and quickly adapt to them. The general meta-learning paradigm [38] is often used to the problems of continuous adaptation. It learns a unified model that can achieve generalized performance across multiple tasks. There is a series of works on meta-learning, including methods for learning update rules for neural models that were explored in the past [39–41]. More recent approaches focus on learning optimizer for deep networks [42,43], learning to learn implicitly via RL [25,44] and generating model parameters [27,45]. Gradient-based meta-learners aim to learn a

good initialization of agent’s policy network by taking policy gradient steps and enable agents rapidly adapt to an unseen task, such as MAML [13] and REPTILE [23]. RNN-based meta-RL methods complete process of adaptation by updating the hidden states of a RNN [25,46,47]. Though the meta-learner’s advantage on solving the multi-task RL tasks, Existing meta-learning approaches often focus on relatively simple tasks with single agent and fully observable settings. Therefore, we proposed a gradient-based self-adaptive meta-learning method which combined meta-learning with MARL approach to deal with the multi-task multi-agent tasks.

The StarCraft II has already been used as RL environments, due to the many interesting challenges inherent to the games [48]. Multi-agent challenge (SMAC) is built on the popular real-time strategy game StarCraft II and makes use of the SC2LE environment. SMAC focuses on decentralized micromanagement where each unit is controlled by an independent learning agent. It uses the StarCraft II game engine to build a new set of rich cooperative multi-agent problems that bring unique challenges, such as the non-stationary of learning, multi-agent credit assignment [49], and the difficulty of representing the value of joint actions [3]. However, the SMAC mainly focus on multi-agent learning in single task scenario, which cannot meet our experimental needs. Other existing benchmarks (e.g., Meta-World [15]) mainly focus on simple single agent in multiple tasks which is too easy comparing with the StarCraft environment. Therefore, we extend the widely adopted StarCraft benchmark SMAC and design a new multi-task multi-agent StarCraft environment, Meta-SMAC, for testing various aspects of continuous adaptation.

Conclusion

In this work, we proposed a gradient-based meta-learning approach called SAML for continuous adaptation in MT-MARL non-stationary tasks. The key idea of the method is to formalize the problem into a two-stage curriculum. The approach first conducts single-task MARL learning process to achieve task-specific policy, and then distills task-specific policies into a unified DRQN with SAML meta-learning method by maximizing transfer and minimizing interference between multiple tasks. For the latter, we designed the meta-SMAC environment based on widely adopted StarCraft benchmark SMAC and defined iterated adaptation games that allowed us to test various aspects of adaptation strategies. Our experimental results on StarCraft II environments show that the proposed method is more efficient than reactive baselines to learn a unified generalization policy across different scenarios, and can adapt more efficiently to unseen tasks.

Acknowledgements This research was supported by the National Natural Science Foundation of China (Grant numbers 62002369, 71702186); the Scientific Research Project of National University of Defense Technology (Grant number ZK19–03) and the National Scientific Research Project (Grant number 2019-JCJQ-ZD-002).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533
- Li J, Monroe W, Ritter A, Jurafsky D, Galley M, Gao J (2016) Deep reinforcement learning for dialogue generation. In: *Proceedings of the 2016 Conference on empirical methods in natural language processing*. Austin, pp 1192–1202
- Rashid T, Samvelyan M, Schroeder C, Farquhar G, Foerster J, Whiteson S (2018) Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: *ICML*. pp 4295–4304
- Mahajan A, Rashid T, Samvelyan M, Whiteson S (2019) Maven: multi-agent variational exploration. In: *NIPS*, pp 7613–7624
- Taylor M E, Stone P (2009) Transfer learning for reinforcement learning domains: a survey. *J Mach Learn Res* 10(7)
- Vh A, Asb D, Elp C (2020) Multitask deep learning for native language identification. *Knowl Based Syst* 209
- Cai Y, Huang Q, Lin Z, Xu J, Li Q (2020) Recurrent neural network with pooling operation and attention mechanism for sentiment analysis: a multi-task learning approach. *Knowl Based Syst* 203:105856
- Omidshafiei S, Pazis J, Amato C, How J P, Vian J (2017) Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In: Precup D, Teh YW (eds) *ICML*, vol 70, pp 2681–2690
- Teh YW, Bapst V, Czarnecki WM, Quan J, Kirkpatrick J, Hadsell R, Heess N, Pascanu R (2017) Distral: Robust multitask reinforcement learning. In: *NIPS*
- Chen S, Andrejczuk E, Cao Z, Zhang J (2020) Ateam: achieving the ad hoc teamwork by employing the attention mechanism. In: *AAAI*
- Hu Y, Sun S, Xu X, Zhao J (2020) Attentive multi-view reinforcement learning. *Int J Mach Learn Cybern* 11(7553)
- Wang D, Cheng Y, Yu M, Guo X, Zhang T (2019) A hybrid approach with optimization-based and metric-based meta-learner for few-shot learning. *Neurocomputing* 349 (Jul.15):202–211
- Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: *ICML*, pp 1126–1135
- Al-Shedivat M, Bansal T, Burda Y, Sutskever I, Mordatch I, Abbeel P (2018) Continuous adaptation via meta-learning in nonstationary and competitive environments. In: *ICLR*
- Yu T, Quillen D, He Z, Julian R, Hausman K, Finn C, Levine S (2020) Meta-world: a benchmark and evaluation for multi-task and meta reinforcement learning. In: *Conference on robot learning*, pp 1094–1100
- Samvelyan M, Rashid T, de Witt C S, Farquhar G, Nardelli N, Rudner T G, Hung C-M, Torr P H, Foerster J, Whiteson S (2019) The starcraft multi-agent challenge. [arXiv:1902.04043](https://arxiv.org/abs/1902.04043)
- Claus C, Boutilier C (1998) The dynamics of reinforcement learning in cooperative multiagent systems, vol 1998
- Lowe R, Wu YI, Tamar A, Harb J, Abbeel OP, Mordatch I (2017) Multi-agent actor-critic for mixed cooperative-competitive environments. In: *NIPS*, pp 6379–6390
- Sunehag P, Lever G, Gruslys A, Czarnecki WM, Zambaldi VF, Jaderberg M (2018) Value-decomposition networks for cooperative multi-agent learning based on team reward. In: *AAMAS*, pp 2085–2087
- Wilson A, Fern A, Ray S, Tadepalli P (2007) Multi-task reinforcement learning: a hierarchical Bayesian approach. In: *ICML*, pp 1015–1022
- Fernández F, Veloso M (2006) Probabilistic policy reuse in a reinforcement learning agent. In: *AAMAS*, pp 720–727
- Gupta A, Mendonca R, Liu Y, Abbeel P, Levine S (2018) Meta-reinforcement learning of structured exploration strategies. In: *NIPS*, pp 5302–5311
- Nichol A, Achiam J, Schulman J (2018) On first-order meta-learning algorithms. [arXiv:1803.02999](https://arxiv.org/abs/1803.02999)
- Yoon J, Kim T, Dia O, Kim S, Bengio Y, Ahn S (2018) Bayesian model-agnostic meta-learning. In: *NIPS*, pp 7332–7342
- Duan Y, Schulman J, Chen X, Bartlett PL, Sutskever I, Abbeel P (2016) RL2: Fast reinforcement learning via slow reinforcement learning. [arxiv:1611.02779](https://arxiv.org/abs/1611.02779)
- Mishra N, Rohaninejad M, Chen X, Abbeel P (2017) A simple neural attentive meta-learner. [arXiv:1707.03141](https://arxiv.org/abs/1707.03141)
- Ha D, Dai A, Le QV (2016) Hypernetworks. [arXiv:1609.09106](https://arxiv.org/abs/1609.09106)
- Dugas C, Bengio Y, Bélisle F, Nadeau C, Garcia R (2009) Incorporating functional knowledge in neural networks. *J Mach Learn Res* 10(6)
- Wang W, Yang T, Liu Y, Hao J, Hao X, Hu Y, Chen Y, Fan C, Gao Y (2020) From few to more: Large-scale dynamic multiagent curriculum learning. In: *AAAI*. pp 7293–7300
- Higgins I, Pal A, Rusu AA, Matthey L, Burgess CP, Pritzel A, Botvinick M, Blundell C, Lerchner A (2017) Darla: Improving zero-shot transfer in reinforcement learning. [arXiv:1707.08475](https://arxiv.org/abs/1707.08475)
- Atnekvist I, Kragic D, Stork JA (2019) Vpe: Variational policy embedding for transfer reinforcement learning. *International Conference on Robotics and Automation (ICRA)* 2019:36–42
- Ruder S (2017) An overview of multi-task learning in deep neural networks. [arXiv:1706.05098](https://arxiv.org/abs/1706.05098)
- Yang Y, Hao J, Liao B, Shao K, Chen G (2020) Qatten: a general framework for cooperative multiagent reinforcement learning. In: *ICML*
- Peng B, Whiteson S, Rashid T, Farquhar G (2020) Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning
- Wang J, Ren Z, Liu T, Yu Y, Zhang C (2020) Qplex: Duplex dueling multi-agent q-learning. In: *ICML*
- Wang T, Dong H, Lesser V, Zhang C (2020) Multi-agent reinforcement learning with emergent roles. In: *ICML*
- Hu J, Jiang S, Harding S A, Wu H, Liao SW (2021) Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning
- Schmidhuber J (1987) Evolutionary principles in self-referential learning, genetic programming
- Bengio Y, Bengio S, Cloutier J (2002) Learning a synaptic learning rule. In: *IJCAI*

40. Schmidhuber J (1992) Learning to control fast-weight memories: an alternative to recurrent nets. *Neural Comput* 4(1):131–139
41. Zhang W, Wang Q, Li J, Xu C (2020) Dynamic fleet management with rewriting deep reinforcement learning. *IEEE Access*, vol 1, no 1
42. Zheng J, Wang L, Wang S, Liang Y, Pan J (2021) Solving two-stage stochastic route-planning problem in milliseconds via end-to-end deep learning. *Complex Intell Syst*:1–16
43. Chen Y, Hoffman MW, Colmenarejo SG, Denil M, Lillicrap TP, Botvinick M, Freitas ND (2016) Learning to learn without gradient descent by gradient descent. In: *ICML*
44. Gan X, Guo H, Li Z (2019) A new multi-agent reinforcement learning method based on evolving dynamic correlation matrix. *IEEE Access* 7:162127–162138
45. Edwards H, Storkey A (2016) Towards a neural statistician. In: *ICLR*
46. Mehta B, Deleu T, Raparthy SC, Pal CJ, Paul L (2020) Curriculum in gradient-based meta-reinforcement learning. [arXiv:2002.07956](https://arxiv.org/abs/2002.07956)
47. Makmal A, Melnikov AA, Dunjko V, Briegel HJ (2017) Meta-learning within projective simulation. *IEEE Access* 4:2110–2122
48. Vinyals O, Ewalds T, Bartunov S, Georgiev P, Vezhnevets A, Yeo M, Makhzani A, Küttler H, Agapiou J, Schrittwieser J et al (2017) A new challenge for reinforcement learning. [arXiv:1708.04782](https://arxiv.org/abs/1708.04782)
49. Foerster JN, Farquhar G, Afouras T, Nardelli N, Whiteson S (2018) Counterfactual multi-agent policy gradients. In: *AAAI*, pp 2974–2982

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.