Lateral Transfer Learning for Multiagent Reinforcement Learning

Haobin Shi, Jingchen Li, Jiahui Mao, and Kao-Shing Hwang, *Senior Member, IEEE*

***Abstract*—Some researchers have introduced transfer learning mechanisms to multiagent reinforcement learning (MARL). However, the existing works devoted to cross-task transfer for multiagent systems were designed just for homogeneous agents or similar domains. This work proposes an all-purpose cross-transfer method, called multiagent lateral transfer (MALT), assisting MARL with alleviating the training burden. We discuss several challenges in developing an all-purpose multiagent cross-task transfer learning method and provide a feasible way of reusing knowledge for MARL. In the developed method, we take features as the transfer object rather than policies or experiences, inspired by the progressive network. To achieve more efficient transfer, we assign pretrained policy networks for agents based on clustering, while an attention module is introduced to enhance the transfer framework. The proposed method has no strict requirements for the source task and target task. Compared with the existing works, our method can transfer knowledge among heterogeneous agents and also avoid negative transfer in the case of fully different tasks. As far as we know, this article is the first work denoted to all-purpose cross-task transfer for MARL. Several experiments in various scenarios have been conducted to compare the performance of the proposed method with baselines. The results demonstrate that the method is sufficiently flexible for most settings, including cooperative, competitive, homogeneous, and heterogeneous configurations.**

***Index Terms*—Attention mechanism, multiagent reinforcement learning (MARL), transfer learning.**

I. Introduction

TRANSFER learning is reusing, combining, and adapting pretrained knowledge from different source tasks to speed up the training process in the target domain, especially for the case of unaffordable cost or small samples [1]. The primary challenge in transfer learning is how to make full use of source tasks to improve the performance of the model in target tasks but avoid the negative transfer at the same time [2]. As a typical technique in adaptive learning control [3], reinforcement learning has achieved remarkable success in many complicated tasks [4]. Many researchers have introduced transfer learning to reinforcement learning for some applausive successes [5], for instance, multitask learning [6], lifelong learning [7], etc. Multiagent reinforcement learning (MARL) is hard to gain an optimal joint policy due to the nonstationarity of multiagent systems [8]. The convergence of MARL is only guaranteed by additional mechanisms such as centralized training [9]. Besides, the joint state–action space is so vast that the cost of training multiagent becomes unaffordable, especially since training always starts from scratch. Therefore, reusing previous experiences and policies is necessary for MARL [10].

Transfer learning for MARL is to transfer knowledge or experiences gained from previous tasks or other peer agents, and use this knowledge to reduce the training cost in various application scenarios. There are two types of transfer for MARL: 1) intraagent transfer and 2) interagent transfer [11]. The intraagent transfer aims at reusing knowledge from pretrained tasks, in which the target task and the source task come from the same domain or not. The interagent transfer reuses knowledge received through communication with other agents, which may have different sensors and internal representations. Much research is focused on subdirections of multiagent transfer learning. Some works introduced curriculum learning into MARL algorithms, involving enhancing the learning in a complicated scenario by the pretrained policies in a low-level scenario [12]. Curriculum learning requires the full similarity of the source and target tasks. Several researchers used single-agent reinforcement learning to learn every agent's independent policy, and then reusing the parameters of these policies for further MARL [13]. Even when the training cost is ignored, this mechanism is still unable to achieve cross-task transfer. Other works acted on a particular multiagent system: in *ad hoc* team, agents are required to be prepared to cooperate with many types of teammates and collaborate without precoordination [14]. A typical method is building a set of policies for agents, and these agents choose an appropriate one to cooperate with an unknown team. Bias transfer (BITER) transferred knowledge for multiagent according to the joint policy in the target task, but it is just suitable for homogeneous multiagent systems [15].

Too few works are denoted to intraagent transfer for MARL, and these works cannot work on all-purpose cross-task scenarios. Compared with that for single-agent reinforcement learning, cross-task transfer for MARL has three primary challenges. The first one is the source task selection [16]. In single-agent transfer learning, the source task can be selected

Manuscript received 30 April 2021; revised 15 July 2021; accepted 25 August 2021. Date of publication 10 September 2021; date of current version 15 February 2023. This work was supported by the National Natural Science Foundation of China under Grant 61976178 and Grant 62076202. This article was recommended by Associate Editor Y.-J. Liu. *(Corresponding author: Kao-Shing Hwang.)*

Haobin Shi, Jingchen Li, and Jiahui Mao are with the School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China.

Kao-Shing Hwang is with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan (e-mail: hwang@ccu.edu.tw).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2021.3108237.

Digital Object Identifier 10.1109/TCYB.2021.3108237

according to the correlation among tasks [17] or coupled dictionary learning [18]. However, in cross-task transfer for MARL, assigning the pretrained policies in the selected source task to agents in the target task is also a problem worth investigating. Braylan and Miikkulainen [19] used a measure of one-frame forward prediction accuracy to guide source selection, but the environment modeling limits the reinforcement learning to model-based algorithms. Another challenge is what to transfer. In an intraagent transfer, policies and experiences were transferred the most commonly [20]. Some works also transferred partial parameters, usually existing in *ad hoc* team [21] and curriculum learning. For many applications, applying all the policies from the source task is impossible for an agent when there are so many agents in the source task, and it may result in a suboptimal actuation, which is the last challenge. The suboptimal actuation comes from uncorrected mapping between the source task and the target task, irrelevant source task, or improper transfer object [22]. As a part of negative transfer, suboptimal actuation also needs to be avoided in cross-transfer for MARL. Although mostly works in the intraagent transfer community focus on curriculum learning [23] or similar-domain transfer [24], the cross-task transfer is still greatly required. The three mentioned challenges restrict the development of cross-task transfer, and the existing works just involved particular multiagent scenarios, such as homogeneous multiagent systems, high-similarity multiagent tasks, and curriculum leaning. The community is in dire need of a feasible direction for multiagent cross-task transfer [11], especially for all-purpose cross-task transfer.

This work is the first to propose an all-purpose cross-task transfer method for MARL, developing a multiagent lateral transfer (MALT) algorithm. MALT can transfer knowledge across tasks in most settings, including cooperative, competitive, homogeneous, and heterogeneous configurations. In MALT, the pretrained policies are mapped to the target task according to a Gaussian mixture model (GMM), which clusters pretrained policies according to the value function and assigns the policies for every agent in the target task according to the clustering result. Several pretrained policies with similar value functions transfer their features to the same agent through a lateral connection, which is inspired by the progressive network [25]. A soft-attention module weighs the features from different pretrained policies to improve the lateral transfer efficiency, which is achieved by processing the outputs of pretrained policy networks when training the target task. Agents in the target task reuse the features selectively and learn new features simultaneously, so that MALT hardly causes negative transfer when the source domain is very different from the target domain. The proposed method is evaluated in many scenarios, such as cooperative, competitive, homogeneous, and heterogeneous multiagent systems, and so on. The results show that MALT has no requirement for the relationship among agents or tasks.

The main contributions of this work are summarized as follows.

1) We discuss multiagent cross-task transfer learning, pointing out several challenges in the development of the cross-task transfer method.
2) Aiming at the discussed challenge, we propose a MALT method, which is feasible for most settings. The developed method provides multiagent cross-task transfer with a workable direction.
3) Because all existing methods have limitations, we compare our method with different baselines on corresponding settings. Moreover, some ablation experiments are conducted to validate our method.

The remainder of this article is organized as follows. The related work is discussed in Section II, followed by a detailed description of our approach in Section III. The experimental results are shown in Section IV. Finally, we conclude our work and discuss future work in Section V.

II. Related Work

As an important part of adaptive learning [26], reinforcement learning has gained remarkable success in various scenarios. As the neural network [27] is introduced, reinforcement learning has the ability to work on large-scale or continuous environments. However, researchers still desire to generalize beyond the learned experiences, accelerating the learning process for a new task. Recent works have introduced transfer learning into reinforcement learning models in different ways. For example, Gamrian and Goldberg [5] isolated the visual component in video games and used zero-shot analogy-based transfer to accelerate the similar-domain transfer; Gupta *et al.* [28] proposed *analogy making*, providing two different agents with a communication channel to share features; and Ammanabrolu and Riedl [29] transferred knowledge graphs to pretrain agents, to enhance the deep Q-learning in text adventure games.

Due to the decentralized policies in MARL, introducing transfer learning into MARL cannot directly extend the existing method in the single-agent transfer field, especially in cross-task transfer. Recent works on intraagent transfer are always aimed at subdirections. For example, some researchers proposed *ad hoc team* [14], involving enhancing the training process when an agent cooperates with unknown peers. Some other researchers focus on curriculum learning [12], that is, transferring the knowledge learned in simple scenarios to complicated but similar scenarios, in which the source and target tasks differ in the number of agents, reward functions, environment sizes, etc. Some works pretrained each agent by single-agent reinforcement learning, and then transfer the learned policies to multiagent environments to gain a more advanced joint policy [13]. All of these cannot achieve general cross-task transfer or are just suitable for particular multiagent systems.

Table I summarizes popular intraagent transfer methods according to their scopes of application. All existing works can transfer knowledge in the same domain with homogeneous agents because agents need no consideration for the policy selection. Methods aiming at the same domain with homogeneous agents were always designed for curriculum learning. For high-similar domains, the relationship among agents was regarded as the transferred objects. Reusing policies or parameters causes negative transfer in the case of different domains,

TABLE I
SCOPES OF APPLICATION FOR THE EXISTING INTRAAGENT TRANSFER METHODS

Work	Object	Same Domain	high-similar Domain	low-similar Domain	Homogeneous	Heterogeneous
Boutsioukis et al. [15]	joint policy	√	×	×	√	×
Taylor et al. [30]	experience	√	×	×	√	×
Vrancx et al. [31]	relationship	√	√	×	√	×
Schutera et al. [13]	parameter	√	×	×	√	×
Didi et al. [32]	policy	√	×	×	√	×
Schwab et al. [24]	policy	√	×	×	√	×
Shao et al. [33]	parameter	√	×	×	√	×
Zhao et al. [34]	relationship	√	√	×	√	×
Agarwal et al. [23]	relationship	√	×	×	√	×
Long et al. [12]	parameter	√	×	×	√	×

and the unsuitable experiences limit the learned policy to a local optimum. Transferring knowledge to low-similar tasks or heterogeneous (the agents in a low-similar task are usually different from the source task) is a research gap.

In fact, the all-purpose cross-transfer technique plays an important role in the MARL community. Researchers want to accelerate the training process for high-similarity tasks by reusing some similar elements, including policies, experiences, and features. For low-similarity tasks, we also desire to transfer some generalized knowledge. Unfortunately, the MARL community has no related work. The existing cross-task transfer techniques for reinforcement learning just focused on single-agent systems.

III. PROPOSED METHOD

This work proposes MALT for transferring knowledge across multiagent tasks. First, we present the preliminary knowledge, and then the overview of the developed MALT is given. The policy assignment module and the attention module are described separately. Finally, we show the lateral transfer framework while the corresponding pseudocode is given.

A. Preliminary Knowledge

1) Progressive Neural Network: Compared with other transfer learning technologies, the progressive neural network can act on low-similarity tasks [25]. It retains pretrained models, and learns lateral connections from the pretrained models for the target task. By transferring features to new tasks, the progressive neural network can solve independent tasks at the end of training and accelerate learning by transferring knowledge to new domains. The progressive neural network has made an excellent performance in single-agent reinforcement learning.

The progressive neural network prevents catastrophic forgetting by instantiating a column for each task. The column for the source task retains the features, and these features are transferred to the target task. A progressive neural network starts with a single deep neural network, and this network has L layers, h_i^1 is the output of the ith layer, and θ_i^1 is the parameters of the ith layer.

When a new task is added to the progressive network, a new column is built. Each layer of the new column receives input from all previous columns via the lateral connection. The

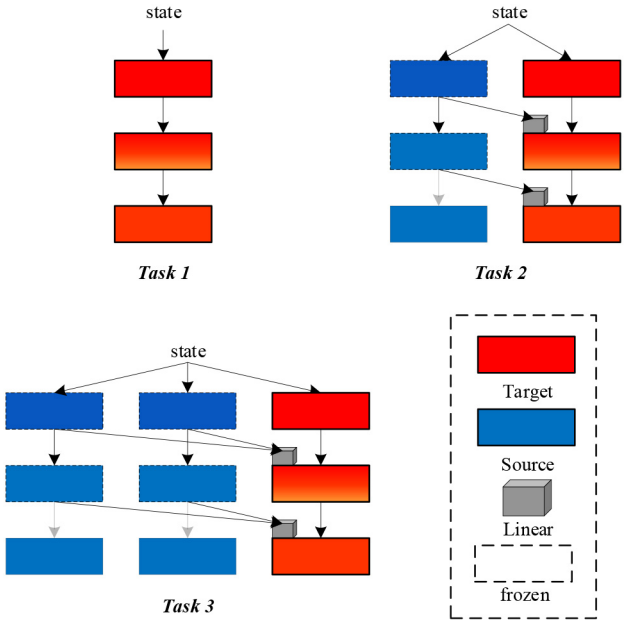


Fig. 1. Progressive neural network has three columns. Each column has three layers.

output of the kth column's ith layer is h_i^k

$$h_i^k = f\left(W_i^k h_{i-1}^k + \sum_{j<k} U_i^{k\cdot j} h_{i-1}^j\right) \tag{1}$$

where U_i^k is the linear layer for the lateral connection from column k to column i, and W_i^k is the weight matrix. The progressive neural network requires that the networks in the target task have the same structure as the networks in the source task. Fig. 1 is a progressive network with three columns.

2) Multiagent Transfer Learning: A MARL task is described as a tuple $\langle S, A, P, R, \gamma\rangle$, where $S = s_0 \times s_1 \times \cdots \times s_n$ is the joint-state space, and each state is composed of the local state for the agent and a local environment state, which is obtained by observing or perceiving the environment; $A = a_0 \times a_1 \times \cdots \times a_n$ is the joint-action space, and it is composed of local action spaces for all the agents in the multiagent system; $P : S \times A \to S = [0, 1]$ is the transition function of the environment; $R = r_0, r_1, \ldots, r_n$ is the reward function for agents; and γ is the discounted factor. According to the state space and the action space, multiagent systems can be divided into two types: 1) homogeneous and 2) heterogeneous [35]. Agents in a homogeneous multiagent

system have the same state space and action space, and heterogeneous multiagent systems allow agents to have different action spaces and state spaces. Multiagent systems can also be divided into cooperative and competitive according to the relationship among reward functions of different agents [36], and it also affects the way of training multiagent.

In the transfer learning for MARL, the domain of a multiagent system is its joint-state space and joint-action space [11]. The same domain means the target domain $D_t = \{S_t, A_t\}$ and the source domain $D_s = \{S_s, A_s\}$ have the same joint-action space $S_t = S_s$ and joint-state space $A_t = A_s$, in which the source task is commonly reused without concerning negative transfer. The similar domains have similar action spaces or state spaces, for example, the number of agents in D_t is different from that in D_s, or S_t and A_t is the subset of S_s and A_s. The target task in D_t can be described as $T_t = \{D_t, R_t, P_t, \gamma\}$, and the source task in D_s can be described as $T_s = \{D_s, R_s, P_s, \gamma\}$.

B. Overview

In MALT, we train multiagent systems in a decentralized way, that is, each agent has its individual policy. In the source task, the number of agents may be different from that in the target task, or more than one source tasks are available. Hence, we first cluster the policies in the source task and according to the corresponding value functions. The cluster is conducted by a GMM. In the generated GMM, the number of Gaussian kernels is the same as the number of agents in the target task, which also forms corresponding policy assignment. Then, we introduce an attention module to weigh the features transferred from different policies, in which the attention values are calculated by the responses of these pretrained policies to the target task. MALT uses a lateral connection to transfer features from the source task to the target task, and the lateral connection happens at the end of every module in policy networks. The developed MALT allows agents to learn new feature representations while receiving features by lateral connections to act on low-similarity tasks.

C. Policy Assignment

In multiagent transfer learning, transferring all pretrained policies from the source task to an agent is difficult. The source domain may have numerous agents, and these agents have different policies. That makes transfer costs expensive. Moreover, in the training of the source task, when agents learn policies in a competitive multiagent environment or a heterogeneous multiagent system, they may learn very different policies, or their policies have opposite features. When these policies are assigned to the same agent in the target task, the effect is worse than policies with similar features.

In a competitive multiagent system, the value functions for two opposing agents are different in most cases, and it is more obvious in zero-sum games. If we want to select several similar policies or the policies of several agents with similar goals, a feasible method is clustering them according to value functions. The value function reflects the expected reward, which can distinguish agents from others with different goals. In MALT, each agent in the target task needs to receive knowledge from several similar policies, and the pretrained policies are assigned by a GMM, which clusters them according to the value functions.

Several states $(s_1, s_2, \ldots)$ are randomly sampled from the state space S_s of the source domain. The sampled states are mapped to a set of state value by each policy. The policy for the ith agent in the source task is π_i', and the state value $v(s_j|\pi_i')$ is calculated

$$v\left(s_j|\pi_i'\right) = \sum_a \pi_i\left(a|s_j\right)Q\left(s_j, a\right) \tag{2}$$

where $Q(s, a)$ is the action value, and $\pi_i'(a|s_j)$ is the probability of executing action a. Then, $(s_1, s_2, \ldots)$ are mapped to a state-value set y_i for π_i'

$$y_i = \left(v_{i,1}, v_{i,2}, \ldots\right) = \left[v\left(s_1|\pi_i'\right), v\left(s_2|\pi_i'\right), \ldots\right]. \tag{3}$$

After state-value sets $(y_1, y_2, \ldots, y_{n'})$ are calculated, they are used to generate a GMM $p(x) = \sum_{k=1}^{n} \phi_k \aleph(x|\mu_k, \Sigma_k)$, where n' is the number of agents in the source task, n is the number of agents in the target task, ϕ_k is the mixture coefficient, and (μ_i, Σ_i) is the mean and the variance of the ith Gaussian distribution.

The parameters of the GMM are estimated by maximizing its likelihood function

$$\begin{aligned} L(y_1, y_2, \ldots, y_{n'}; \mu, \Sigma) &= \prod_{i=1}^{n'} p(y_i) \\ &= \prod_{i=1}^{n'} \sum_{k=1}^{n} \phi_k \aleph(y_i|\mu_k, \Sigma_k). \end{aligned} \tag{4}$$

The corresponding log-likelihood function is

$$\begin{aligned} \ln L(y_1, y_2, \ldots, y_{n'}; \mu, \Sigma) &= \ln \prod_{i=1}^{n'} p(y_i) \\ &= \sum_{i=1}^{n'} \ln \sum_{k=1}^{n} \phi_k \aleph(y_i|\mu_k, \Sigma_k). \end{aligned} \tag{5}$$

The generated GMM contains n Gaussian distributions. For the ith agent in the target task, several policies whose corresponding state-value sets have the highest probability densities in $(\mu_j, \sum_j)$ are assigned for the lateral transfer. The assigned policies for agent i is $\langle \pi_{i_1}', \ldots, \pi_{i_m}' \rangle$, where m is the number of policies assigned to each agent, and

$$i_1, \ldots, i_m = \arg\max_{i_1, \ldots, i_m} \prod_{k=1}^{m} \aleph\left(y_{i_k}|\mu_i, \Sigma_i\right). \tag{6}$$

When n' is not larger than n, this operation can be canceled, and a policy can be assigned to several agents simultaneously. However, when n' is larger than n, this step is essential. The GMM divides policies into several sets, and policies in each set are similar, or the corresponding agents in the source task have similar goals. By assigning pretrained policies through the GMM, each agent in the target task obtains a suitable transfer set, and the target task avoids negative transfer caused

by transferring several policies with competitive relationships to an agent at the same time.

The proposed policy assignment method has a precondition that the similarity of policies can be reflected by the corresponding value functions. From (2), we know that the value function is related to the policy and Q-value (action value). In value-based reinforcement learning methods such as Q-learning, the policy is represented by Q-table, so that two similar policies must have similar value functions. For policy-based reinforcement learning methods, the value function can be overwritten as

$$v(s|\pi) = \sum_{t=t_{(0)}}^{T} \gamma^{t-t_{(0)}} r(s,a)|_{a\ \pi(s)} \tag{7}$$

where $r(s,a)$ is the reward function. Given an arbitrary trajectory $[(s_{t_0}, a_{t_0}), (s_{t_1}, a_{t_1}), \ldots, (s_T, a_T)]$, the probability for policy π to follow this trajectory is $\prod_{t=t_0}^{T} \pi(a_t|s_t)$, so that similar policies will make similar trajectory. The value function for a policy is the expected reward under trajectories. Hence, in the case of the same reward function, policies with similar value functions must make parallel trajectory, which suggests the policies are also similar.

D. Attention Module

Although each agent in the target task is assigned similar pretrained policies through GMM, there still will be some differences among these pretrained policies. Even in homogeneous and cooperative multiagent systems, the policy that an agent obtains through reinforcement learning is different from each other. In MALT, the pretrained policies also give outputs in the training of the target task. These outputs have no direct effect on the behaviors of agents, but they retain the responses of the pretrained policies to the target task.

For improving the transfer efficiency in MALT, this work introduces a soft-attention module [37]. The attention mechanism is widely used to optimize various tasks in machine learning, such as natural language processing [38], computer vision [39], and speech recognition [40]. In the source task, the pretrained policies have learned how to cooperate (or compete) with each other, and they already considered each other before making decisions. In MALT, considering that the outputs of the pretrained policies must have some certain relationships learned in the source task, we use a soft-attention module to weigh the pretrained policies according to the outputs of the pretrained policies.

There are m pretrained policies $\langle \pi'_{i_1}, \ldots, \pi'_{i_m} \rangle$ that transfer knowledge to the ith agent. In the training of the target task, pretrained policies also map the current state to actions. Each pretrained policy generates a deterministic action or distribution according the state. $\pi(s)$ is the output of the policy π. In the training process, the ith agent's state is s_i, and each assigned policy also responses to s_i. The outputs of the assigned policies $\langle \pi'_{i_1}, \ldots, \pi'_{i_m} \rangle$ is $\langle \pi'_{i_1}(s_i), \ldots, \pi'_{i_m}(s_i) \rangle$, and the outputs are processed by linear layers, respectively. For the jth assigned policy π'_{i_j}

$$g_{i_j} = \text{Linear}\left[\pi'_{i_j}(s_i)\right]. \tag{8}$$

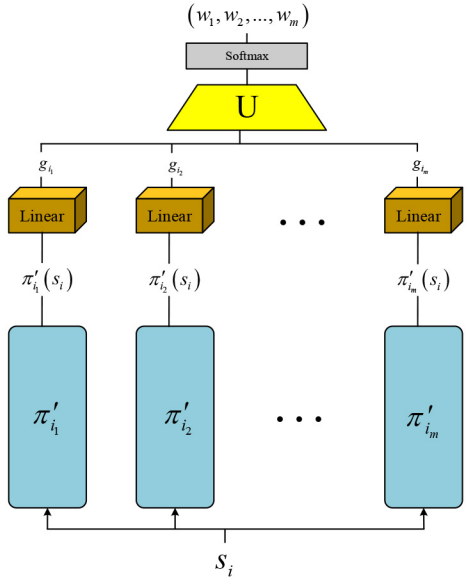


Fig. 2. Attention module in MALT, where U is the fully connected layer, and m pretrained policies are assigned to the same agent.

Then, a fully connected layer is used to generate weights

$$(w_1, \ldots, w_m) = \text{Softmax}\left[U\left(g_{i_1}, \ldots, g_{i_m}\right)\right] \tag{9}$$

where U is the projection matrix.

The attention module is shown in Fig. 2. It should be noticed that we build an individual projection matrix for every transfer layer instead of sharing the same projection matrix. In a deep reinforcement learning model, the former layers in the policy network are used to extract features, while the later layers are always used to make decisions. Considering various functions of these layers, we build different projection matrices for the lateral connections.

E. Lateral Connection With Attention Module

The proposed MALT transfers feature from the pretrained policies to the target task by lateral connection. The lateral connection allows agents to reuse features of the source task while learning new features. The lateral connection is flexible enough to improve the training speed when transferring in similar domains and hardly causes negative transfer when the target domain is much different from the source domain.

In MALT, the lateral connection does not happen in all layers as other research [41]. When an agent is trained in a complex multiagent environment, a deep network is required to represent policy for the agent, and the network always contains a variety of networks, such as convolutional neural network, recurrent neural network, and fully connected network. In MALT, several contiguous layers with the same type are regarded as a module in the policy network, and the lateral connection happens at the end of each module. Although the transfer does not happen at every layer, it does not affect the reuse of features learned in the source task. The output layer of the pretrained policy is independent of all modules. Fig. 3 gives an example to illustrate it, in which the first

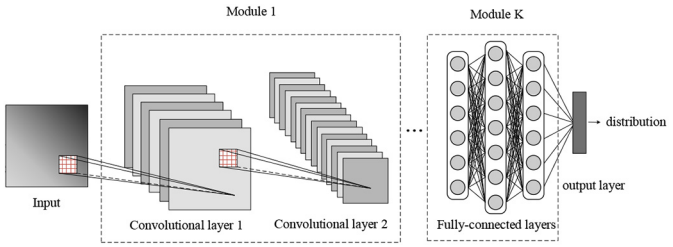


Fig. 3. Policy network with K modules. The first module is multiple convolutional layers, and the latest module is multiple fully connected layers. The output layer is independent of all modules.

module consists of two convolutional layers while three fully connected layers are regarded as the last module.

Let K denote the number of modules in a policy network, and all policy networks have the same structure. In this work, the lateral connection transfers features at the latest layer of each modules. In training, the assigned pretrained policies are frozen. For agent i, the output of the kth module in the jth assigned policy is h'_{k,i_j}. First, the outputs of the kth modules in all assigned policies are concatenated, and h_k is generated

$$h'_k = \left(h'_{k,i_1}, \ldots, h'_{k,i_m}\right). \tag{10}$$

Then, the weights $W_{i,k} = (w_1, \ldots, w_m)$ for the kth module is output by an attention module. $W_{i,k}$ is acted on h'_k, and u_k is generated

$$u_k = h'_k \otimes \mathrm{W}_i^k = \left(h'_{k,i_1} \cdot w_1, \ldots, h'_{k_{i,i_m}} \cdot w_m\right). \tag{11}$$

As a part of inputs for $(k+1)$th module in the policy of agent i, u_k is processed by a linear layer together with h_k

$$h_{k+1} = f(\mathrm{Linear}(u_k, h_k); \theta_{k+1}) \tag{12}$$

where θ_{k+1} is the parameter of the $(k+1)$th module. The linear layer integrates u_k and h_k, and the dimensionality of its output is consistent with that of the next module's input. The action or the action distribution of an agent is output by the output layer

$$a = f(\mathrm{Linear}(u_L, h_L); \theta_{\mathrm{out}}) \tag{13}$$

where θ_{out} is the parameter of the output layer.

When training the target task, the assigned pretrained policy networks are frozen, and each agent reuses the transferred features while learning new features. The attention module weights the transferred features according to the assigned pretrained policy networks' outputs. The parameters of the attention modules are updated in the training of the target task. Fig. 4 gives an example of the lateral transfer in MALT, in which three pretrained policy networks are assigned to agent i, and there are three modules and an output layer in all policy networks.

In MALT, each agent in the target task reuses knowledge by lateral connections. Although an agent still needs to train an entire policy network, the training for target tasks is accelerated by transferring the features from the pretrained policies. When transferring knowledge to a similar domain, the lateral connection retains the source task's features, which are still valid for the target task. When the source task is unsuitable for the target task, the entire policy network also has the ability to obtain new features, so that MALT hardly causes the negative transfer.

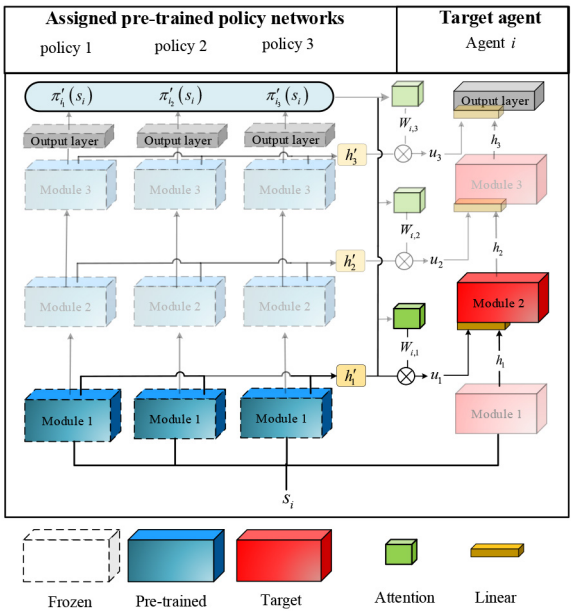


Fig. 4. Lateral connections for agent i in the target task. Three pretrained policies in the source task are assigned to the agent, and each policy network has three modules.

Algorithm 1 is the pseudocode of MALT. The computation complexity for the policy assignment process is affected by the number of agents (n' and n), and it can be ignored when compared with the forward propagation of neural networks. In the training process of MALT, the pretrained policies also conduct feedforward propagation, and an additional attention module and lateral connection network are needed. However, the pretrained policies need no backpropagation. As for the space complexity, the parameter count for the additional attention module and lateral connection networks is far less than that for policy networks.

IV. Experiments

Because no existing works can achieve all-purpose cross-task transfer for MARL, we first validate MALT with several simple baselines and then compare them with other works in their home fields.

A. Validation Experiment

In this section, the performance of MALT is evaluated across several settings. First, MALT is used to transfer knowledge in the same domain, in which the source task and the target task have the same state space and action space, but different reward functions and transition functions. Next, we conduct an experiment on transferring knowledge in similar domains, in which the source domain and the target domain have similar but different state spaces or action spaces. Then, the performance of MALT is investigated in entirely different MARL domains. The three experiments are designed to investigate whether MALT can achieve all-purpose cross-task transfer.

Algorithm 1 Multiagent Transfer Learning

Definition the state space of the source domain: S_s, the number of agents in the source task: n', the number of agents in the target task n, the policy networks in the source task: $(\pi'_1, \pi'_2, \ldots, \pi'_{n'})$, the number of modules in a policy network: L, and the number of assigned policies for an agent: m.
Sample $(s_1, s_2, \ldots)$ from S_s randomly.
for $i = 1$ **to** n' **do**
    $y_i = [v(s_1|\pi'_i), v(s_2|\pi'_i), \ldots]$
Generate $p(x) = \sum_{k=1}^{n} \phi_k \aleph(x|\mu_k, \Sigma_k)$ with $y_1, \ldots, y_{n'}$ by Expectation-Maximization.
for $i = 1$ **to** n **do**
    Get $i_1, \ldots, i_m$ by (6)
while not done **do**
    for $i = 1$ **to** n **do**
        for $k = 2$ **to** L **do**
            Get h'_k for agent i by (10).
            Get u_k for agent i by (11).
            Get h_{k+1} for agent i by (12).
        Output ith agent's action a_i by (13).
    Execute joint action $(a_1, \ldots, a_n)$ and store experience.
    if update **then**
        Freeze $(\pi'_1, \pi'_2, \ldots, \pi'_{n'})$ and update.

1) Setups: To evaluate the degree to which pretrained policies contribute more to the target task, we calculate the weighted average attention value Z. Z_i is the weighted average attention of the ith policy in the transfer to an agent, and it is calculated at the end of the training

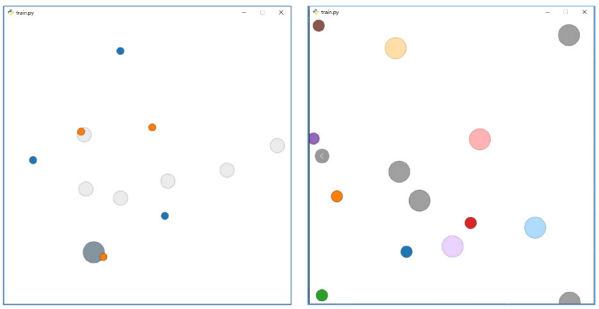
$$Z_i = \frac{1}{L} \sum_{t}^{T} \sum_{l}^{L} w_{i,t}^{l} |r_t / R|, R = \sum_{t}^{T} r_t \tag{14}$$

where R is the total reward for the agent in a round, r_t is the reward for the agent at time t, and $w_{i,t}^{l}$ is the weight for the ith assigned policy in module l at time t.

The weighted average attention value represents the importance of each pretrained policy in the transfer to the target task. For an agent, a more important policy has a more weighted average attention.

In all scenarios, the source tasks are trained by multiagent deep deterministic policy gradient (MADDPG) [42]. MADDPG is a common algorithm for MARL, in which agents are trained by a centralized training and decentralized execution (CTDE) mechanism. There are few transfer learning methods for MARL that can achieve all-purpose cross-task transfer for MARL, so that we compare MALT with some frequently used transfer methods in reinforcement learning and control groups. The baselines are specified as follows.

1) *No Transfer* uses a new model without the reuse of knowledge to train the target.
2) *Finetune* selects a policy from the source task randomly for each agent in the target task, and finetunes the last module and the output layer.



(a) (b)

Fig. 5. (a) CTC. The gray agents are the treasure hunters, the colored large agents are the "banker," and the colored small circles are the treasures. (b) RT. The gray agent is the speakers, the colored large agents are the listeners, and the colored small circles are the destinations.

3) *Fully Finetune* selects a policy from the source task randomly for each agent in the target task, but the entire model is finetuned.
4) *No GMM* is the proposed MALT but assigns knowledge randomly.
5) *No Attention* is the proposed MALT but without attention module.

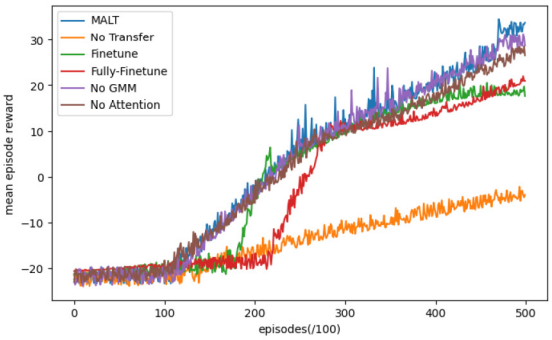
2) Same Domain: The multiagent practice environment (MPE) framework is used as the platform in which experiments are implemented. We compare MALT and baselines in the extended cooperative treasure collection (CTC) and rover tower (RT) [43]. The first one is a fully observable environment, and there are two kinds of agents, six "treasure hunters," and two "treasure banks," which are shown in Fig. 5(a). The other one is a speaker–listener environment with four speakers and four listeners, as shown in Fig. 5(b).

In these experiments, the target domain has the same state space and state space as the source domain, but the reward function in the target task is different from that in the source task.

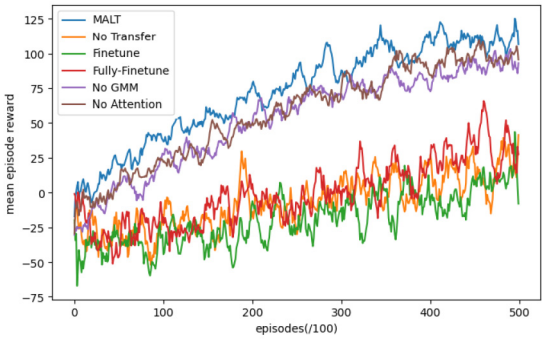
In CTC, the goal of the source task is that the hunters collect treasures and deposit them with the correctly colored banks, and the hunters are penalized for collisions and rewarded for collections, while the banks are penalized by the distance to the closest relevant holding agent and rewarded for obtaining treasure from collectors. In the target task, the hunters collect treasures and deposit them with the bank in a different color.

In RT, the source task is that the speakers communicate with the listeners to make them move toward the destination. The agents in the source task are rewarded by the distance from the listeners to the destinations. In the target task, the speakers communicate with the listeners to make them move to the speakers and avoid colliding with the destinations.

There are two kinds of agents in each scenario so that the multiagent systems are heterogeneous, and there are cooperative relationships among agents. Although there are different agents in environments, we allow agents to reuse knowledge from policies of agents with different types. In training, we use actor–critic networks to train agents, where the actor networks are the policy networks of agents and the critic networks in the target task are trained from scratch.



(a)



(b)

Fig. 6. (a) Mean episode rewards on the CTC. (b) Mean episode rewards on the RT.

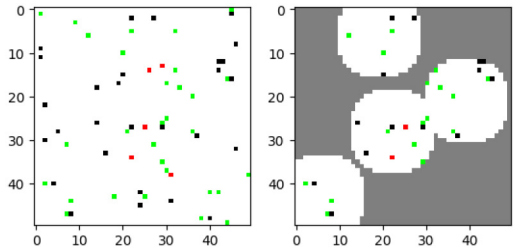


Fig. 7. Drones monitoringİtask with five humans and four drones. The left is the given area, and the right is the joint observations of all drones.

In the two experiments, the actor network of each agent has two modules. The first module has two fully connected layers, and the other has a fully connected layer. When training the target task, we assign each agent with three pretrained policies. The learning rate for the actor network and critic network is 0.01, and the discounted factor is 0.97.

The results are shown in Fig. 6. In both figures, the mean episode reward of the MALT has a distinct advantage over the finetune. In the experiments on CTC, the target task is similar to the source task, and finetune (Finetune and Fully Finetune) is an effective method to transfer knowledge. Compared with Finetune and Fully Finetune, the lateral transfer methods (MALT, No GMM, and No Attention) perform better when the target task is similar to the source task. After 50 000 episodes, the mean episode reward of MALT is larger than Finetune and Fully Finetune, and it is far larger than training without transfer (No transfer). As known from Fig. 6(b), when the target task has a significant difference with the source task, finetune (Finetune and Fully Finetune) cannot improve the training, and it may cause negative transfer (Finetune). The advantage of

TABLE II
WEIGHTED AVERAGE ATTENTION VALUES OF THE PRETRAINED POLICIES IN THE EXPERIMENT ON CTC

Z	Agent1	Agent2	Agent3	Agent4	Agent5	Agent6
Policy 1			0.43	0.50		0.78
Policy 2	0.69	0.11				
Policy 3					0.22	
Policy 4		0.54	0.28			0.13
Policy 5	0.07	0.35	0.29	0.21	0.38	
Policy 6	0.24			0.29	0.40	0.09

TABLE III
WEIGHTED AVERAGE ATTENTION VALUES OF THE PRETRAINED POLICIES IN THE EXPERIMENT ON RT, WHERE AG. IS THE ABBREVIATION FOR AGENT, AND PO. IS THE ABBREVIATION FOR POLICY

Z	Ag.1	Ag.2	Ag.3	Ag.4	Ag.5	Ag.6	Ag.7	Ag.8
Po.1					0.81	0.44		
Po.2	0.48						0.28	0.53
Po.3		0.11					0.15	
Po.4		0.02	0.16	0.83				0.28
Po.5		0.87	0.62		0.10	0.44	0.57	0.19
Po.6				0.16	0.09			
Po.7	0.07					0.12		
Po.8	0.45		0.22	0.11				

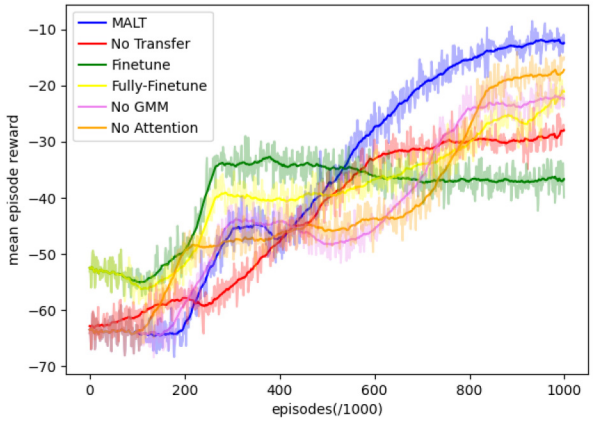


Fig. 8. Mean episode rewards on drone monitoring. The mean episode rewards are recorded every 1000 episodes, and each experiment runs 1 000 000 episodes.

the MALT can be known from this experiment. Comparing MALT with No GMM, we can know that the policy assignment in this work improves the lateral transfer a little on the same domain. Although the multiagent systems are heterogeneous, the same state space makes all policy networks have similar feature representations, so that the policy assignment cannot show its worth greatly. The effect of the attention module can be verified by comparing MALT and No Attention. With the attention module, agents can receive knowledge selectively, gaining more suitable features from pretrained policy networks.

The policy assignment and weighted average attention on CTC are given in Table II, while that for RT is shown in Table III. The weighted average attention values are displayed by the table with the agents in the target task as rows and the pretrained policies in the source task as columns. The blank means that the actor network is not assigned to the agent. We can know that a pretrained policy in the source task may be assigned to many agents from both tables. For example, actor network 5 is assigned to 6 agents (Table III). The weighted

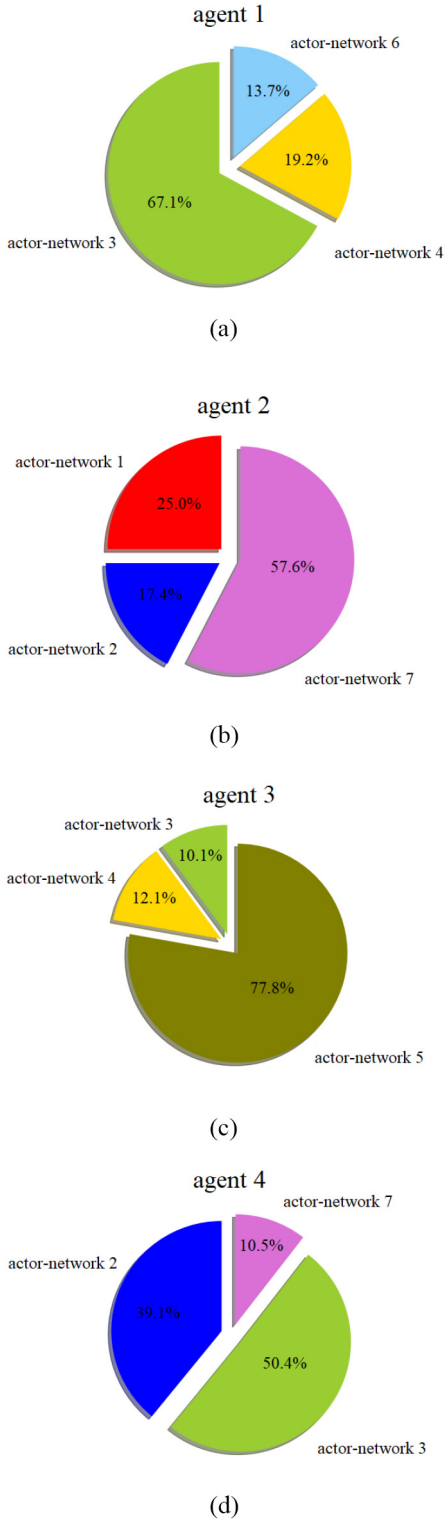


Fig. 9. Weighted average attention values for agents, and which actor networks (policies) are assigned for each agent is shown in figures.

average attentions for different agents have a significant difference. Although policy 2 is assigned to both agent 1 and agent 2, the weighted average attention of policy 2 for agent 1 is far larger than that for agent 2 (Table II).

3) Similar Domains: The "Drones Monitoring" task is used for this experiment. As shown in Fig. 7, in a square area, some humans (red rectangles) are randomly walking in the area, the

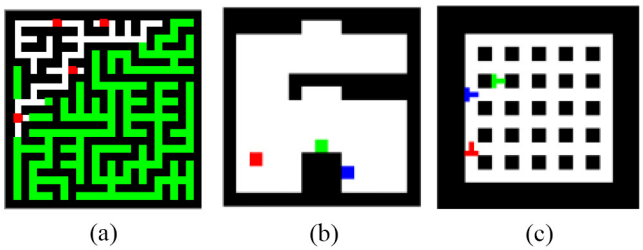


Fig. 10. Several multiagent scenarios. (a) Cleaner: Four cleaners (red) will clean the indoor environment of a room. Cleaners cannot go through walls (black) and should clean the dirty floors (green). At each step, each agent will clean the block it is standing on, and turn it to white. (b) MoveBox: Two agents (red and blue) carry a box (green). Only when both the agents are at the side of the box, they can carry the box to the corresponding direction. (c) CatchPig: Two agents (red and blue) run in a pigsty and try to catch the moving pig (green). Each agent can only observe about 90° of the environment in front of it.

green rectangles are trees, and the black rectangles are walls. The drones fly freely in the given area, and they can observe a partial area around themselves. This is a partially observable environment, and the drones cannot share observations. Humans walk randomly in the given area, but they cannot across the walls and the trees. The goal of the drones is to monitor all humans in the given area. Because the number of humans is larger than the number of drones and the observation range of the drones is small, the drones need to cooperate in monitoring all humans at the same time.

The reward of each drone is the sum of their local reward and the global reward for all drones. For a drone, each human in its observation makes its reward add 1. If there are humans not observed by any drones, each of them makes the global reward minus 5. In each round, drones need to monitor humans for 100-time steps. The goal of drones is to maximize the total reward for these 100-time steps.

In this experiment, there are two modules for the actor network. The first one has two convolutional layers, and the other has two fully connected layers. The learning rate for the actor network and critic network is 0.01, and the discounted factor is 0.97.

In the source domain, the map size is 100, the number of drones is 7, the view range is 10, and the number of humans is 9. In the target domain, the map size is 50, there are four drones and five humans, and the view range is the same as the source domain. The joint state space and the joint action space in the target domain are different from those in the source domain, but the source domain and the target domain have the same objects in them, so that the target domain is similar to the source domain. In these experiments, the learning rate for the actor network and critic network is 0.01, and the discounted factor is 0.97.

The results are shown in Fig. 8. The lateral connection methods (MALT, No GMM, and No Attention) are not as good as finetune (Finetune and Fully Finetune) at the beginning of the training. However, as the number of episodes increases, finetune is difficult to find better policies for the target task. That is because the source task and target task have the same goal (reward function), so that finetune (Finetune and Fully Finetune) can achieve better jump start. However, finetune

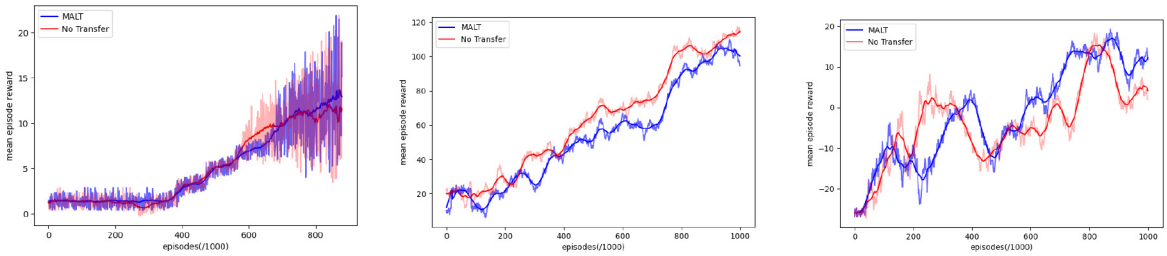


Fig. 11. Mean episode reward when transfer knowledge in different domains by MALT and No Transfer. (a) Results on Cleaner (b) Results on MoveBox (c) Results on CatchPig.

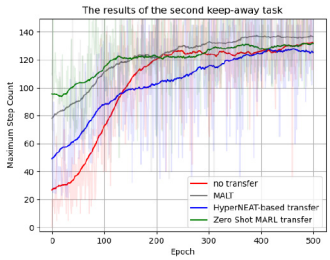


Fig. 12. Results in the fully similar transfer experiments.

(Finetune and Fully Finetune) may miss some important features and cause an optimal solution. With reusing features, the lateral connection can better help multiagent systems to find excellent policies for new tasks. Comparing MALT with No GMM, we can know the knowledge assignment is necessary for MALT in this case. Although multiagent systems are homogeneous in this experiment, the drones in the source task learned diversified policies, and the number of agents in the source task is much larger than that in the target task, so that the policy assignment proves its worth. Comparing MALT with No Attention, we can see the attention module improves the lateral connection and makes training more steady, which is consistent with the previous experiment. The weighted average attention values are given in Fig. 9.

4) Different Domains: In these experiments, the source task is the drone monitoring in the above, and the target tasks are several scenarios shown in Fig. 10.

When the target tasks are entirely different from the source task, the reuse of knowledge is invalid, and usually causes negative transfer. In these experiments, we compare MALT with No Transfer, and investigate whether MALT will cause negative transfer in different domains.

The results are shown in Fig. 11. The MALT results in positive transfer on Cleaner and CatchPig, with only one case of negative transfer. Comparing MALT with No Transfer, we can see the results of the MALT and No Transfer are similar, and the difference can be regarded as caused by random errors. This trend proves that MALT hardly caused negative transfer in different domains, while some generalized features may also be transferred to enhance the training.

B. Comparison Experiment

In this section, we compare our MALT with several existing works on particular settings. The developed MALT is the first method to achieve all-purpose cross-task transfer for MARL, and this experiment is to investigate whether MALT can rival other works in their home fields.

1) Fully Similar Transfer: Some works aim to transfer knowledge in fully similar tasks, which always have high similarity state space and action space. In these works, the source task and target task differed in just the number of agents or the size of environments.

HyperNEAT-based transfer [32] combined the behavioral diversity and neuro-evolution method to transfer evolved policies within multiagent tasks. In that work, HyperNEAT-based transfer was evaluated in keep-away RoboCup Soccer scenarios, in which the number of keepers in the target task was different from that in the source task.

Policy-sharing MARL transfer [24] utilized fully convolutional Q-networks to represent the policies for agents, and leveraged policy sharing in tasks across team sizes and field sizes. The work also evaluated its method in keep-away RoboCup Soccer scenarios.

In this section, we compare our MALT with HyperNEAT-based transfer and policy-sharing MARL transfer in the RoboCup Soccer scenario (the scenario setup is consistent with [24]), investigating the performance of MALT in the two baseline methods' home field. In this experiment, all of the MALT and baseline methods use deep Q-learning as the reinforcement learning model (limited by policy-sharing MARL transfer). We set two tasks with different numbers of agents, pretraining the first task (source task) and then transferring the learned knowledge to the other task (target task). The numbers of teammates and opponents are 5 and 5 in the source task, while the target task has four teammates and six opponents.

The results are shown in Fig. 12. Although the policy-sharing MARL transfer gained the best performance at the earlier training, our MALT went beyond it quickly at about 170 epochs. HyperNEAT-based transfer also outperformed no transfer training at the beginning, but it cannot learn more advanced policies, which is also a shortage for policy-sharing MARL transfer. Our MALT allows agents to learn new feature representations while transferring the features in the source task, making the final policies more suitable for the target task. So the maximum step count (the maximum number of steps that the agents hold the ball) for MALT is larger than the two baselines and the no transfer training.

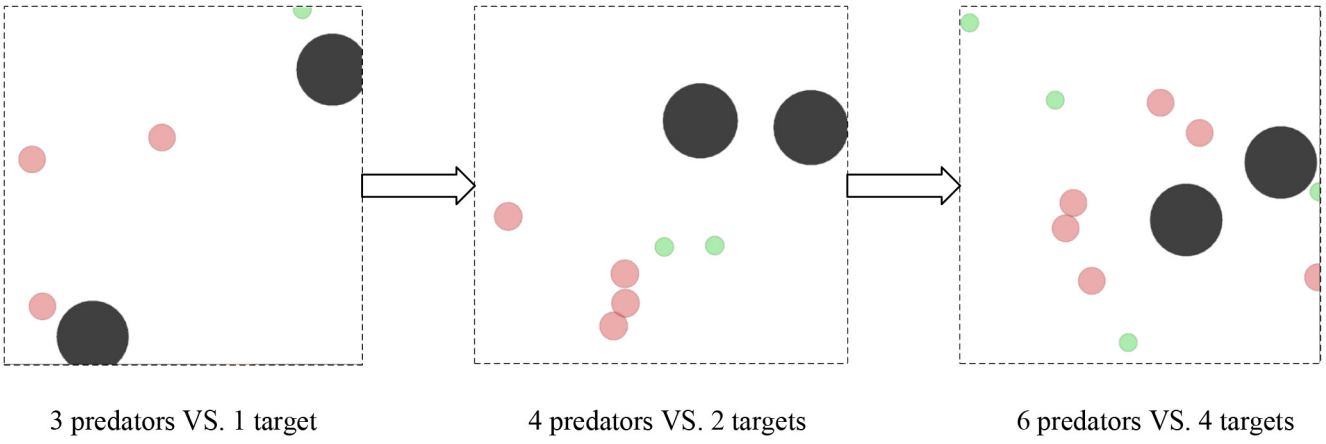


Fig. 13. Curriculum learning scenarios we used.

2) Curriculum Learning Experiment: Although curriculum learning cannot be regarded as cross-task transfer strictly, we still investigate the performance of MALT on curriculum learning task while comparing it with other algorithms. The previous experiments have shown that MALT can learn more progressive policies. This experiment is designed to provide more convincing demonstrations.

Inherent structure transfer [23] created a shared agent-entity graph, building a communication channel for agents to share messages. This method can achieve efficient curriculum learning by transferring the inherent structures.

Evolutionary population curriculum (EPC) [12] leveraged an evolutionary mechanism to fix an objective misalignment issue for multiagent curriculum learning, in which the evolutionary algorithm is used mix-and-match and fine-tuning to adjust multiple sets of agents.

In this experiment, the predator-prey task in MPE is used as the experiment scenario. As shown in Fig. 13, the first scenario has three predators and one target, and there are four predators and two targets in the second scenario, while the number of predators and targets are 6 and 4 in the third scenarios. The policies for targets in the three scenarios are pretrained, and these policies will not be updated in the learning processes. First, MADDPG [42] is used to train predators in the first scenario, then our MALT and the two baselines transfer knowledge learned by MADDPG to the second scenarios, and finally, the learned policies are transferred to the third scenarios by MALT and baselines. Because the number of agents in the previous curriculum is usually less than that in the current curriculum, the policy assignment module in MALT is not necessary, so that we assign the policies randomly instead of using GMM in this experiment. In these scenarios, predators need to capture the targets as soon as possible. The reward function for predators consists of relative distance and collision. The specific reward setup is as default as the MPE platform.

We conduct each experiment with 130 000 episodes and compare the mean predator rewards. As shown in Table IV, when transferring knowledge from Curriculum 1 to Curriculum 2, our MALT and baselines have similar performances with MADDPG, because the second scenario is relatively simple so that more advanced policies cannot be reflected directly. However, in Curriculum 3, both our MALT and baselines outperform MADDPG to some extent. Inherent structure transfer went ahead of MADDPG at the 130000th episode, benefiting from reusing the agents' inherent structure. EPC got the best result due to a feasible mix-and-match operation. The performance of our MALT is a bit less than that of EPC, but it still better than MADDPG. This curriculum experiment suggests that the developed MALT can still work on curriculum learning, and it can rival the specialized curriculum learning methods.

TABLE IV
MEAN PREDATOR REWARDS IN THE CURRICULUM LEARNING EXPERIMENT

	Curriculum 1	Curriculum 2	Curriculum 3
MADDPG (no transfer)	9.4	18.0	48.4
MALT		18.6	55.7
Inherent Structure Transfer		18.2	52.4
EPC		17.9	56.0

V. CONCLUSION

The combination of lateral connection with attention mechanism represents a promising approach for multiagent transfer learning. This work proposes a transfer learning algorithm, MALT, which is the first method to achieve all-purpose cross-task transfer for MARL. The developed MALT is mainly composed of a lateral connection, a soft-attention module, and a policy assignment module. Existing transfer methods for MARL cannot reuse knowledge across multiagent tasks or are limited to specific scenarios. MALT requires nothing in the source domain and target domain, but the policy networks of all agents have the same structure. MALT is compared with baselines in the same domain, similar domains, and different domains. Several experiments have verified that MALT can transfer knowledge efficiently in the same domain and similar domains. The proposed MALT allows agents to learn new features while reusing pretrained features, so that it hardly causes negative transfer in different domains. The policies assignment and attention module proposed in MALT are proved to be necessary for transfer knowledge better. We analyzed the

importance of each policy network in the lateral connection by the proposed weighted average attention value. Moreover, to show the advantages of MALT more convincingly, we compare MALT with some intraagent transfer methods in their home fields, and the results suggest that MALT can rival them even outperforming them. In future works, we will find a feasible bootstrapping algorithm for the policy assignment and try to extend MALT into continual learning.

REFERENCES

[1] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. Hershey, PA, USA: IGI Global, 2010, pp. 242–264.

[2] L. Ge, J. Gao, H. Ngo, K. Li, and A. Zhang, "On handling negative transfer and imbalanced distributions in multiple source transfer learning," *Stat. Anal. Data Min.*, vol. 7, no. 4, pp. 254–271, 2014.

[3] J. Yu, L. Zhao, H. Yu, and C. Lin, "Barrier Lyapunov functions-based command filtered output feedback control for full-state constrained nonlinear systems," *Automatica*, vol. 105, pp. 71–79, Jul. 2019.

[4] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[5] S. Gamrian and Y. Goldberg, "Transfer learning for related reinforcement learning tasks via image-to-image translation," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2063–2072.

[6] Y. Teh *et al.*, "Distral: Robust multitask reinforcement learning," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2017, pp. 4496–4506.

[7] E. Brunskill and L. Li, "Pac-inspired option discovery in lifelong reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 316–324.

[8] Y. Hu, Y. Gao, and B. An, "Multiagent reinforcement learning with unshared value functions," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 647–662, Apr. 2015.

[9] Z. Li, C. Guo, and Y. Xuan, "A multi-agent deep reinforcement learning based spectrum allocation framework for D2D communications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[10] H. Kono, A. Kamimura, K. Tomita, and T. Suzuki, "Transfer learning method using ontology for heterogeneous multi-agent reinforcement learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 10, pp. 156–164, 2014.

[11] F. L. Da Silva and A. H. R. Costa, "A survey on transfer learning for multiagent reinforcement learning systems," *J. Artif. Intell. Res.*, vol. 64, no. 1, pp. 645–703, 2019.

[12] Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, and X. Wang, "Evolutionary population curriculum for scaling multi-agent reinforcement learning," 2020. [Online]. Available: arXiv:2003.10423.

[13] M. Schutera, N. Goby, D. Neumann, and M. Reischl, "Transfer learning versus multi-agent learning regarding distributed decision-making in highway traffic," 2018. [Online]. Available: arXiv:1810.08515.

[14] P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein, "Ad hoc autonomous agent teams: Collaboration without pre-coordination," in *Proc. 24th AAAI Conf. Artif. Intell.*, 2010, pp. 1504–1509.

[15] G. Boutsioukis, I. Partalas, and I. Vlahavas, "Transfer learning in multi-agent reinforcement learning domains," in *Proc. Eur. Workshop Reinforcement Learn.*, 2011, pp. 249–260.

[16] S. Kelly and M. I. Heywood, "Knowledge transfer from keepaway soccer to half-field offense through program symbiosis: Building simple programs for a complex task," in *Proc. Annu. Conf. Genet. Evol. Comput.*, 2015, pp. 1143–1150.

[17] J. Sinapov, S. Narvekar, M. Leonetti, and P. Stone, "Learning inter-task transferability in the absence of target task samples," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2015, pp. 725–733.

[18] D. Isele, M. Rostami, and E. Eaton, "Using task features for zero-shot knowledge transfer in lifelong learning," in *Proc. IJCAI*, 2016, pp. 1620–1626.

[19] A. E. Braylan and R. Miikkulainen, "Object-model transfer in the general video game domain," in *Proc. 12th Artif. Intell. Interact. Digit. Entertainment Conf.*, 2016, pp. 136–142.

[20] L. Zhou, P. Yang, C. Chen, and Y. Gao, "Multiagent reinforcement learning with sparse interactions by negotiation and knowledge transfer," *IEEE Trans. Cybern.*, vol. 47, no. 5, pp. 1238–1250, May 2017.

[21] S. V. Albrecht and S. Ramamoorthy, "Comparative evaluation of MAL algorithms in a diverse set of ad hoc team problems," in *Proc. 11th Int. Conf. Auton. Agents Multiagent Syst.*, vol. 1, 2012, pp. 349–356.

[22] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 1–67, Jun. 2020.

[23] A. Agarwal, S. Kumar, K. Sycara, and M. Lewis, "Learning transferable cooperative behavior in multi-agent team," in *Proc. Int. Conf. Auton. Agents Multiagent Syst. (AAMAS)*, 2020, pp. 1741–1743.

[24] D. Schwab, Y. Zhu, and M. Veloso, "Zero shot transfer learning for robot soccer," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst.*, 2018, pp. 2070–2072.

[25] A. A. Rusu *et al.*, "Progressive neural networks," 2016. [Online]. Available: arXiv:1606.04671.

[26] G. Cui, J. Yu, and Q.-G. Wang, "Finite-time adaptive fuzzy control for MIMO nonlinear systems with input saturation via improved command-filtered backstepping," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Aug. 6, 2020, doi: 10.1109/TSMC.2020.3010642.

[27] C. Fu, Q.-G. Wang, J. Yu, and C. Lin, "Neural network-based finite-time command filtering control for switched nonlinear systems with backlash-like hysteresis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 3268–3273, Jul. 2021.

[28] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," 2017. [Online]. Available: arXiv:1703.02949.

[29] P. Ammanabrolu and M. Riedl, "Transfer in deep reinforcement learning using knowledge graphs," in *Proc. 13th Workshop Graph-Based Methods Nat. Lang. Process. (TextGraphs)*, 2019, pp. 1–10.

[30] A. Taylor, I. Duparic, E. Galván-López, S. Clarke, and V. Cahill, "Transfer learning in multi-agent systems through parallel transfer," in *Proc. Workshop Theor. Grounded Transf. Learn. 30th Int. Conf. Mach. Learn. (Poster)*, 2013, pp. 1–9.

[31] P. Vrancx, Y.-M. De Hauwere, and A. Nowé, "Transfer learning for multi-agent coordination," in *Proc. ICAART*, Rome, Italy, 2011, pp. 263–272.

[32] S. Didi and G. Nitschke, "Multi-agent behavior-based policy transfer," in *Applications of Evolutionary Computation*. Cham, Switzerland: Springer, 2016, pp. 181–197.

[33] K. Shao, Y. Zhu, and D. Zhao, "StarCraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 1, pp. 73–84, Feb. 2019.

[34] Q. Zhao, T. Jiang, N. Morozs, D. Grace, and T. Clarke, "Transfer learning: A paradigm for dynamic spectrum and topology management in flexible architectures," in *Proc. IEEE 78th Veh. Technol. Conf. (VTC Fall)*, 2013, pp. 1–5.

[35] Q. Ma and G. Miao, "Output consensus for heterogeneous multi-agent systems with linear dynamics," *Appl. Math. Comput.*, vol. 271, pp. 548–555, Nov. 2015.

[36] F. Buccafurri, A. Comi, G. Lax, and D. Rosaci, "Experimenting with certified reputation in a competitive multi-agent scenario," *IEEE Intell. Syst.*, vol. 31, no. 1, pp. 48–55, Jan./Feb. 2015.

[37] H. Yang, J.-Y. Kim, H. Kim, and S. P. Adhikari, "Guided soft attention network for classification of breast cancer histopathology images," *IEEE Trans. Med. Imag.*, vol. 39, no. 5, pp. 1306–1315, May 2020.

[38] G. Liu and J. Guo, "Bidirectional LSTM with attention mechanism and convolutional layer for text classification," *Neurocomputing*, vol. 337, pp. 325–338, Apr. 2019.

[39] Q. Ye, S. Yuan, and T.-K. Kim, "Spatial attention deep net with partial PSO for hierarchical hybrid hand pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 346–361.

[40] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2015, pp. 577–585.

[41] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Proc. Conf. Robot Learn.*, 2017, pp. 262–270.

[42] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2017, pp. 6379–6390.

[43] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.

Haobin Shi received the Ph.D. degree in computer science and technology from Northwestern Polytechnical University, Xi'an, China, in 2008.

He is a Professor with the School of Computer Science, Northwestern Polytechnical University and a Visiting Scholar with Electrical Engineering Department, National Sun Yat-sen University, Kaohsiung, Taiwan. He is the Director of the Chinese Association for Artificial Intelligence. His research interests include intelligent robots, decision support systems, artificial intelligence, multiagent systems, and machine learning.

Jingchen Li received the master's degree in computer application technology from the School of Computer Science, Northwestern Polytechnical University, Xi'an, China, in 2020, where he is currently pursuing the Ph.D. degree in computer science and technology.

His research interests include transfer learning, multiagent reinforcement learning, and machine learning.

Jiahui Mao received the bachelor's degree in underwater acoustic engineering from the School of Navigation, Northwestern Polytechnical University, Xi'an, China, in 2020, where he is currently pursuing the master's degree in computer science and technology with the School of Computer Science.

His research interest covers reinforcement learning, image processing, and machine learning.

Kao-Shing Hwang (Senior Member, IEEE) received the M.M.E. and Ph.D. degrees in electrical and computer engineering from Northwestern University, Evanston, IL, USA, in 1989 and 1993, respectively.

He is an ASE Chair Professor with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, and a Professor with the Department of Healthcare Administration and Medical Informatic, Kaohsiung Medical University, Kaohsiung. He is also a Visiting Chair Professor with the Department of Computer Engineering, Northwestern Polytechnical University, Xi'an, China. He had been with National Chung Cheng University, Chiayi, Taiwan, from 1993 to 2011, where he was the Deputy Director of Computer Center from 1998 to 1999, the Chairman of the Electrical Engineering Department from 2003 to 2006, and the Director of the Opti-Mechatronics Institute from 2010 to 2011. His research interest includes methodologies and analysis for various intelligent robot systems, visual servoing, and reinforcement learning for robotic applications.

Prof. Hwang is a Fellow of the Institution of Engineering and Technology.