Multi-Agent Policy Transfer via Task Relationship Modeling

Anonymous Author(s)
Affiliation
Address
email

Abstract

Team adaptation to new cooperative tasks is a hallmark of human intelligence, which has yet to be fully realized in learning agents. Previous works on multi-agent transfer learning accommodate teams of different sizes, but heavily rely on the generalization ability of neural networks for adapting to unseen tasks. We posit that the relationship among tasks provides the key information for policy adaptation. To utilize such relationship for efficient transfer, we try to discover and exploit the knowledge among tasks from different teams, propose to learn effect-based task representations as a common latent space among tasks, and use it to build an alternatively fixed training scheme. We demonstrate that the task representation can capture the relationship among teams and generalize to unseen tasks. As a result, the proposed method can help transfer learned cooperation knowledge to new tasks after training on a few source tasks, and the learned transferred policies can also help solve tasks that are hard to learn from scratch.

1 Introduction

Cooperation in human groups is characterized by resiliency to unexpected changes and purposeful adaptation to new tasks [39]. This flexibility and transferability of cooperation is a hallmark of human intelligence. Computationally, multi-agent reinforcement learning (MARL) [56] provides an important means for machines to imitate human cooperation. Although recent MARL research has made prominent progress in many aspects of cooperation, such as policy decentralization [26, 32, 42, 48, 5], communication [11, 20], and organization [19, 44, 45], how to realize the ability of group knowledge transfer is still an open question.

Compared to single-agent knowledge reuse [60], a unique challenge faced by multi-agent transfer learning is the varying size of agent groups. The number of agents and the length of observation inputs in unseen tasks may differ from those in source tasks. To solve this problem, existing multi-agent transfer learning approaches build population-invariant [25] and input-length-invariant [47] learning structures using graph neural networks [1] and attention mechanisms like transformers [16, 59]. Although these methods handle varying populations and input lengths well, their knowledge transfer to unseen tasks mainly depends on the inherent generalization ability of neural networks. The relationship among tasks in MARL is not fully exploited for more efficient transfer.

Towards making up for this shortage, we study the discovery and utilization of common structures in multi-agent tasks and propose Multi-Agent Transfer reinforcement learning via modeling TAsk Relationship (MATTAR). In this learning framework, we capture the common structure of tasks by modeling the similarity among transition and reward functions of different tasks. Specifically, we train a forward model for all source tasks to predict the observation, state, and reward at the next timestep given the current observation, state, and actions. The challenge is how to embody the similarity and the difference among tasks in this forward model, we specifically introduce difference by giving each

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

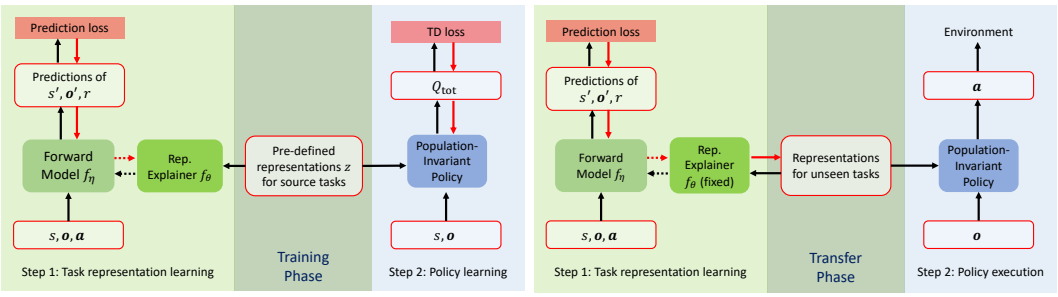


Figure 1: Transfer learning scheme of our method. The black arrows indicate the direction of data flow and the red ones indicate the direction of gradient flow. The dashed arrows indicate the flow between the hypernetwork and the generated network.

source task a unique representation and model the similarity by generating the parameters of the forward model via a shared hypernetwork, which we call the representation explainer.

To learn a well-formed representation space that encodes task relationship, an alternative-fixed training method is proposed to learn the task representation and representation explainer. During training, representations of source tasks are pre-defined and fixed as mutual orthogonal vectors, and the representation explainer is learned by optimizing the forward model prediction loss on all source tasks. When facing an unseen task, we fix the representation explainer and backpropagate gradients through the fixed forward model to learn the representation of the new task by a few samples. Furthermore, we design a population-invariant policy network conditioned on the learned task representation. During policy training, the representations for all source tasks are fixed, and the policy is updated to maximize the expected return over all source tasks. On an unseen task, we obtain the transferred policy by simply inserting the new task representation into the learned policy network.

On the SMAC [33] and MPE [26] benchmarks, we empirically show that the learned knowledge from source tasks can be transferred to a series of unseen tasks with great success rates. We also pinpoint several other advantages brought by our method. First, fine-tuning the transferred policy on unseen tasks achieves better performance than learning from scratch, indicating that the task representation and pre-trained policy network provide a good initialization point. Second, training on multiple source tasks gets better performance compared to training on them individually and other multi-task learning methods, showing that MATTAR also provides a method for multi-agent multi-task learning. Finally, although not designed for this goal, our structure enables comparable performance against single-task learning algorithms when trained on single tasks.

2 Method

In this paper, we focus on knowledge transfer among fully cooperative multi-agent tasks that can be modeled as a Dec-POMDP [27] consisting of a tuple $G=\langle I, S, A, P, R, \Omega, O, n, \gamma\rangle$, where I is the finite set of n agents, $s \in S$ is the true state of the environment, and $\gamma \in [0, 1)$ is the discount factor. At each timestep, each agent i receives an observation $o_i \in \Omega$ drawn according to the observation function $O(s, i)$ and selects an action $a_i \in A$. Individual actions form a joint action $\boldsymbol{a} \in A^n$, which leads to a next state s' according to the transition function $P(s'|s, \boldsymbol{a})$, and a reward $r = R(s, \boldsymbol{a})$ shared by all agents. Each agent has local action-observation history $\tau_i \in \mathrm{T} \equiv (\Omega \times A)^* \times \Omega$. Agents learn to collectively maximize the global action-value function $Q_{\text{tot}}(s, \boldsymbol{a}) = \mathbb{E}_{s_{0:\infty}, a_{0:\infty}}[\sum_{t=0}^{\infty} \gamma^t R(s_t, \boldsymbol{a}_t)|s_0 = s, \boldsymbol{a}_0 = \boldsymbol{a}]$ (a little notation abuse: the subscript here for s and $\boldsymbol{a}$ indicates the timestep while the subscript for observation and action elsewhere in this paper indicates the index of the agent).

Overall, our framework first trains on several source tasks $\{\mathcal{S}_i\}$ and then transfers the learned cooperative knowledge to unseen tasks $\{\mathcal{T}_j\}$. As shown in Fig. 1, our learning framework achieves this by designing modules for (1) task representation learning and (2) policy learning. In the following sections, we first introduce how we design the representation learning module and its learning scheme in different phases. Then, we describe the details of policy learning, including the population-invariant structure for dealing with inputs and outputs of varying sizes.

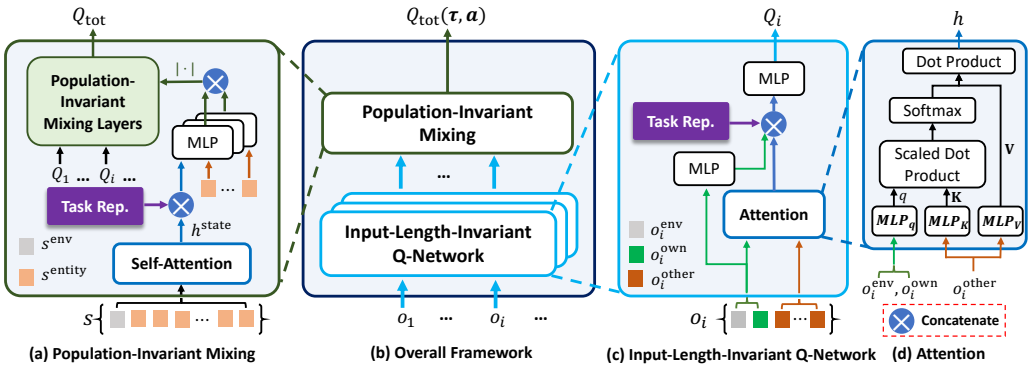


Figure 2: Population-invariant network structure for policy learning.

2.1 Task representation learning

Our main idea in achieving knowledge transfer among multi-agent tasks is to capture and exploit both the common structure and the unique characteristics of tasks by learning task representation. A task distinguishes itself from other tasks by its transition and reward functions. Therefore, we incorporate our task representation learning component into the learning of a forward model that predicts the next state, local observations, and reward given the current state, local observations, and actions.

We associate each task i with a representation $z_i \in \mathbb{R}^m$ and expect it to reflect the relationship of the transition dynamics of tasks. For modeling task similarity, all source and unseen tasks share a representation explainer, which takes as input the task representations and outputs the parameters of the forward model. We then train the representation explainer on all source tasks. Concretely, for a source task $\mathcal{S}_i$, parameters of the forward model are generated as $\eta_i = f_\theta(z_i)$, where θ denotes the parameters of the representation explainer. The forward model contains three predictors: $f^s_{\eta_i}$, $f^o_{\eta_i}$, and $f^r_{\eta_i}$. Given the current state s, agent j's observation o_j, and action a_j, these predictors estimate the next state s', the next observation o'_j, and the global reward r, respectively.

A possible method for training on source tasks is to backpropagate the forward model's prediction error to update both the representation explainer and task representations. However, in practice, this scheme leads to representations with very small norms, making it difficult to get a meaningful representation space. To solve this problem, we propose pre-determining the task representation for each source task and learning the representation explainer by backpropagating the prediction error. Such a method can help form an informative task representation space and build a mapping from task representation space to the space of forwarding model parameters. In practice, we initialize source task representations as mutually orthogonal vectors. Specifically, we first randomly generate vectors in $\mathbb{R}^m$ for source tasks, and then use the Schimidt orthogonalization [4] on these vectors to obtain source task representations. With the pre-defined task representations, the representation explainer is optimized to minimize the following loss function:

$$J(\theta) = \sum_{i=1}^{N_{\text{src}}} J_{\mathcal{S}_i}(\theta),$$

where N_{src} is the number of source tasks, and

$$J_{\mathcal{S}_i}(\theta) = \mathbb{E}_{\mathcal{D}}\Big[\sum_j \big[\|f^s_{\eta_i}(s, o_j, a_j) - s'\|^2 + \lambda_1 \|f^o_{\eta_i}(s, o_j, a_j) - o'_j\|^2 + \lambda_2 (f^r_{\eta_i}(s, o_j, a_j) - r)^2\big]\Big]$$

is the per-task prediction loss. Here, $\mathcal{D}$ is the replay buffer, and λ_1, λ_2 are scaling factors.

We fix the source task representations and learn the representation explainer during the training phase. When it comes to the transfer phase, we aim to find a good task representation that can reflect the similarity of the new task to source tasks. To achieve this goal, we fix the trained representation explainer and learn the task representation by minimizing the prediction loss of the forward model on the new task. Specifically, we randomly initialize a task representation z, keep θ fixed, and get the forward model parameterized by $\eta = f_\theta(z)$. Then the task representation z is updated by backpropagating the prediction loss for transition and reward functions through the fixed f_η.

To keep the new task representation in the well-formed space of source task representations, we learn the new task representation as a linear combination of source task representations:

$$z = \sum_{i=1}^{N_{\text{src}}} \mu_i z_i \ \text{ s.t. } \ \mu_i \geq 0, \sum_{i=1}^{N_{\text{src}}} \mu_i = 1.$$

In this way, what we are learning is the weight vector μ. To make the learning more stable, we additionally optimize an entropy regularization term $\mathcal{H}(\mu)$. The final loss function for learning z is:

$$J_{\mathcal{T}}(\mu) = \lambda \mathcal{H}(\mu) + \mathbb{E}_{\mathcal{D}}\Big[\sum_j \big[\|f_\eta^s(s, o_j, a_j) - s'\|^2 + \lambda_1 \|f_\eta^o(s, o_j, a_j) - o_j'\|^2 + \lambda_2 (f_\eta^r(s, o_j, a_j) - r)^2\big]\Big].$$

The detailed architectures for task representation learning are described in App. E.

2.2 Task policy learning

After the task representation is learned by modeling the transition and reward functions, it will be used to learn and transfer the policy on the source and unseen tasks.

A difficulty faced by multi-agent transfer learning is that the dimensions of inputs and outputs vary across tasks. We use a population-invariant network (PIN, Fig. 2) to solve this problem. The idea is not novel, and our contribution here is a light-weight structure that achieves similar performance to other complex PIN modules [18]. Our PIN uses the value decomposition framework and consists of two main components: an individual Q-network shared by the agents, and a monotonic mixing network [32] that learns the global Q-value as a combination of local Q-values.

For the individual Q-network, like in previous work [18], we decompose the observation o_i into parts relating to the environment o_i^{env}, agent i itself o_i^{own}, and other entities $o_i^{\text{other}} = \{o_i^{\text{other}_{j \neq i}}\}$. We adopt attention mechanism to get a fixed-dimensional embedding h:

$$q = \texttt{MLP}_q([o_i^{\text{env}}, o_i^{\text{own}}]), \quad \mathbf{K} = \texttt{MLP}_K([o_i^{\text{other}_1}, \dots, o_i^{\text{other}_j}, \dots]),$$
$$\mathbf{V} = \texttt{MLP}_V([o_i^{\text{other}_1}, \dots, o_i^{\text{other}_j}, \dots]), \quad h = \texttt{softmax}(q\mathbf{K}^{\text{T}}/\sqrt{d_k})\mathbf{V},$$

where $[\cdot, \cdot]$ is the vector concatenation operation, d_k is the dimension of the query vector, and bold symbols are matrices. Embedding h is then fed into the subsequent network together with the task representation z for estimating action values. For the mixing network, we decompose the state s into parts relating to different entities s^{entity} and the environment s^{env}. We apply a similar self-attention module to integrate information from these parts of the state and obtain a fixed-dimensional embedding vector h^{state}. h^{state}, the task representation z, and s^{entity} are used to generate the parameters of the mixing network. In some multi-agent tasks, the number of actions also varies in different tasks. We use a mechanism similar to other popular population-invariant networks [46, 16] to deal with this issue, which is discussed in detail in App. F.

During the whole process of policy learning, we fix the task representation z. Compared to policy learning, which typically lasts for 2M timesteps, the training of task representation costs few samples. In practice, we collect 50K samples for learning task representations and the representation explainer. When **transferring to new tasks**, we use the individual Q-network and the representation explainer trained on source tasks. We learn the task representation for 50K timesteps and insert it into the individual Q-network. Agents execute in a decentralized manner according to this Q-network.

3 Experiments

In this section, we design experiments to evaluate the following properties of the proposed method: (1) Generalizability to unseen tasks. Our learning framework can extract knowledge from multiple source tasks and transfer the cooperation knowledge to unseen tasks, and task representations play an indispensable role in transfer (Sec. 3.1). (2) A good initialization for policy fine-tuning. Fine-tuning the transferred policy can succeed in super hard tasks, which can not be solved when learning from scratch (Sec. 3.2). (3) Benefits of multi-task training. Our multi-task learning paradigm helps the model better leverage knowledge of different source tasks to boost the learning performance compared to training on source tasks individually (Sec. 3.3). (4) Performance advantages on single tasks. Although we did not design our framework for single-task learning, our network performs better against the underlying algorithm (App. G).

We evaluate MATTAR on the SMAC [33] and MPE [26] benchmarks. To better fit the multi-task training setting, we extend the original SMAC maps and sort out three task series, which involve various numbers of Marines, Stalkers/Zealots, and Marines/Maneuvers/Medivacs, respectively. For the MPE tasks, we test on a discrete version of `Spread` and `Gather`, with different numbers of agents

Table 1: Transfer performance (mean win rates with variance) on the first series of SMAC maps.

	Source Tasks			Unseen Tasks				
	2s3z	3s5z	3s5z_3s6z	1s8z	1s9z	2s8z	2s9z	7s3z
MATTAR	**1.00**±0.00	**0.99**±0.01	**0.48**±0.13	**0.79**±0.09	**0.60**±0.12	**0.93**±0.09	**0.84** ±0.04	**0.16**±0.12
w/o task rep.	0.99±0.01	0.96±0.02	0.20±0.08	0.12±0.13	0.07±0.12	0.47±0.18	0.25±0.20	0.15±0.19
0 task rep.	0.23±0.12	0.08±0.05	0.00±0.00	0.02±0.03	0.00±0.00	0.01±0.01	0.02±0.02	0.00±0.00
UPDeT-b	0.94±0.04	0.86±0.13	0.09±0.08	0.16±0.11	0.11±0.10	0.29±0.22	0.15±0.13	0.02±0.04
UPDeT-m	0.60±0.11	0.47±0.15	0.03±0.03	0.08±0.06	0.04±0.04	0.14±0.12	0.06±0.05	0.01±0.01
REFIL	0.75±0.09	0.43±0.13	0.01±0.01	0.08±0.04	0.03±0.01	0.08±0.05	0.05±0.04	0.06±0.04

Table 2: Transfer performance (mean win rates with variance) on the second series of SMAC maps.

	Source Tasks			Unseen Tasks				
	MMM	MMM2	MMM4	MMM0	MMM1	MMM3	MMM5	MMM6
MATTAR	**1.00**±0.00	**0.92**±0.20	**0.93**±0.12	**0.98**±0.02	**0.97**±0.04	**0.86**±0.10	**0.47**±0.15	**0.09**±0.02
w/o task rep.	0.94±0.05	0.23±0.39	0.33±0.25	0.81±0.15	0.37±0.36	0.07±0.05	0.22±0.30	**0.09**±0.17
0 task rep.	0.61±0.07	0.07±0.06	0.21±0.22	0.28±0.19	0.11±0.13	0.08±0.10	0.08±0.12	0.02±0.04
UPDeT-b	**1.00**±0.00	0.78±0.04	0.41±0.14	0.73±0.21	0.84±0.07	0.57±0.15	0.00±0.00	0.00±0.00
UPDeT-m	0.48±0.03	0.15±0.19	0.20±0.07	0.30±0.16	0.27±0.13	0.28±0.08	0.00±0.00	0.00±0.00
REFIL	0.97±0.01	0.04±0.02	0.06±0.03	0.93±0.02	0.38±0.06	0.12±0.04	0.00±0.00	0.00±0.00

Table 3: Transfer performance (mean win rates with variance) on the third series of SMAC maps.

	Source Tasks				Unseen Tasks			
	5m	5m_6m	8m_9m	10m_11m	3m	4m	4m_5m	6m
MATTAR	**1.00**±0.00	0.72±0.05	**0.83**±0.05	0.81±0.09	**0.94**±0.27	**0.97**±0.02	0.04±0.05	**1.00**±0.00
w/o task rep.	0.97±0.01	0.01±0.02	0.01±0.01	0.01±0.03	0.86±0.03	0.88±0.04	0.00±0.00	0.95±0.03
0 task rep.	0.78±0.39	0.16±0.12	0.30±0.24	0.40±0.28	0.00±0.00	0.21±0.15	0.01±0.01	0.67±0.47
UPDeT-b	**1.00**±0.00	**0.93**±0.05	0.81±0.19	**0.94**±0.04	0.81±0.08	0.95±0.06	**0.29**±0.17	**1.00**±0.00
UPDeT-m	0.77±0.09	0.32±0.03	0.35±0.05	0.43±0.02	0.36±0.04	0.57±0.03	0.10±0.06	0.91±0.09
REFIL	0.73±0.03	0.00±0.00	0.01±0.01	0.03±0.02	0.68±0.06	0.74±0.02	0.00±0.00	0.71±0.02
	Unseen Tasks							
	6m_7m	7m	7m_8m	8m	9m	9m_10m	10m	10m_12m
MATTAR	0.74±0.15	**1.00**±0.00	**0.83**±0.04	**1.00**±0.00	**1.00**±0.00	**0.84**±0.09	**1.00**±0.00	**0.07**±0.01
w/o task rep.	0.03±0.02	0.94±0.03	0.08±0.10	0.93±0.04	0.86±0.05	0.04±0.02	0.52±0.22	0.00±0.00
0 task rep.	0.31±0.22	0.67±0.47	0.49±0.35	0.67±0.47	0.66±0.46	0.32±0.24	0.65±0.46	0.00±0.00
UPDeT-b	**0.78**±0.05	0.99±0.01	0.73±0.11	0.99±0.02	0.99±0.01	0.80±0.16	0.99±0.01	**0.07**±0.04
UPDeT-m	0.35±0.10	0.92±0.03	0.38±0.05	0.83±0.05	0.66±0.11	0.33±0.09	0.17±0.08	0.03±0.02
REFIL	0.01±0.00	0.66±0.03	0.01±0.01	0.63±0.05	0.55±0.05	0.01±0.00	0.46±0.02	0.00±0.00

trying to cover all or one landmarks. The detailed environmental settings are described in App. B. To ensure fair evaluation, we carry out all the experiments with five random seeds, and the results are shown with a 95% confidence interval in all figures. For a more comprehensive description of experimental details, please refer to App. D.

3.1 Generalizability to unseen tasks

As the major desiderata of our method, we expect that MATTAR has better transfer performance on unseen tasks. We note that there are few related works applicable to tasks of varying sizes, and we compare our method against the state-of-the-art multi-agent transfer method UPDeT [16] and REFIL [18]. UPDeT transfers knowledge from a single source task. For a fair comparison, we transfer from each source task to every unseen task and calculate the best (UPDeT-b) and mean (UPDeT-m) performance on each unseen task. For the test phase, we conduct transfer experiments on both source tasks and unseen tasks. Results on the SMAC benchmark are shown in Tab. 1∼3, and those on the MPE are shown in Tab. 4 and 5.

We find that MATTAR shows superior transfer performance on unseen tasks, significantly outperforming UPDeT-b and REFIL, especially in complex settings requiring sophisticated coordination like the MMM series. MATTAR is effective even for unseen tasks that are very different from source tasks. For example, the MATTAR policy learned on `2s3z`, `3s5z`, and `3s5z_vs_3s6z` can win 79% of games on `1s8z`. On `Spread` tasks, trained on source tasks with 2, 4, 7, and 9 agents,

Table 4: Evaluation on MPE: Transfer performance (mean success rates with variance) on `Spread` with different numbers of agents.

	Source Tasks				Unseen Tasks			
	2	4	7	9	3	5	6	8
MATTAR	**1.00**±0.00	**0.97**±0.03	**0.17**±0.16	**0.12**±0.05	**0.98**±0.01	**0.75**±0.12	**0.19**±0.10	**0.09**±0.09
w/o task rep.	**1.00**±0.00	0.55±0.30	0.02±0.01	0.00±0.00	0.83±0.13	0.30±0.13	0.03±0.04	0.00±0.00
0 task rep.	0.98±0.01	0.96±0.03	0.13±0.18	0.09±0.04	0.94±0.07	0.70±0.17	0.09±0.09	**0.09**±0.07
UPDeT-b	0.73±0.23	0.09±0.00	0.02±0.02	0.00±0.00	0.41±0.16	0.05±0.02	0.00±0.00	0.02±0.02
UPDeT-m	0.39±0.12	0.06±0.00	0.00±0.00	0.00±0.00	0.18±0.08	0.02±0.01	0.00±0.00	0.01±0.01
REFIL	**1.00±0.00**	0.93±0.04	0.07±0.04	0.01±0.01	0.96±0.03	0.73±0.06	0.06±0.05	0.05±0.00

Table 5: Evaluation on MPE: Transfer performance (mean success rates with variance) on `Gather` with different numbers of agents.

	Source Tasks				Unseen Tasks			
	2	4	7	9	3	5	10	15
MATTAR	**1.00**±0.00	**1.00**±0.00	0.82±0.11	0.84±0.17	**1.00**±0.00	**1.00**±0.00	0.63±0.12	**0.55**±0.22
w/o task rep.	**1.00**±0.00	**1.00**±0.00	0.88±0.09	0.84±0.17	**1.00**±0.00	0.99±0.01	0.62±0.13	0.53±0.18
0 task rep.	**1.00**±0.00	**1.00**±0.00	0.73±0.26	0.75±0.18	**1.00**±0.00	0.99±0.01	0.54±0.26	0.50±0.29
UPDeT-b	0.66±0.34	0.38±0.38	0.25±0.12	0.09±0.09	0.61±0.36	0.36±0.27	0.03±0.03	0.06±0.06
UPDeT-m	0.32±0.16	0.15±0.15	0.07±0.02	0.04±0.04	0.20±0.10	0.11±0.07	0.01±0.01	0.02±0.02
REFIL	**1.00**±0.00	**1.00**±0.00	**0.99**±0.02	**0.99**±0.01	**1.00**±1.00	**1.00**±0.00	**0.73**±0.14	0.51±0.14

MATTAR shows good generalization performance on unseen tasks with different numbers of agents and significantly outperforms the baselines and ablations. In `Gather` tasks, MATTAR also exhibits significant advantage over UPDeT. In order to further test the generalization ability of MATTAR, we design unseen tasks with quite larger numbers of agents compared to source tasks, such as the task with 15 agents. Results show that MATTAR can still achieve the highest success rate, about 55%.

To explain the performance of our method, we first check the learned representations on unseen tasks and then carry out ablation studies.

What representations are learned for unseen tasks? When encountering an unseen task, we first learn its representation as a linear combination of the representations for source tasks. Specifically, we directly update the coefficients of this linear combination by backpropagating the prediction error of the forward model. For a deeper understanding of how our method transfers the learned knowledge, here we investigate the learned coefficients of the linear combination.

In Tab. 6, we show the coefficients of two unseen tasks for each series of maps. We observe that the largest coefficient typically corresponds to the source task that is the most similar to the unseen task. For example, `5m` is the closest source task to the unseen task `4m`, and the coefficient of `5m` is 61%. There are also some exceptions. For example, for the unseen task `10m_vs_12m`, the coefficients of two source tasks, `5m` and `10m_vs_11m`, are equal, and they together take up 86% of all the coefficients. While `10m_vs_11m` is very similar to `10m_vs_12m` as the team composition is similar, the policy for solving `5m` is different from that for `10m_vs_12m`, we find that agents first form some groups to set up an attack curve quickly in `10m_vs_12m`. Therefore, on a local battlefield, there are around five allies fighting against a similar number of enemies. In this case, the policy learned from `5m` can be used locally to promote coordination in `10m_vs_12m`.

We conclude that, in MATTAR, task representation learning captures the similarity of task dynamics and thereby discovers opportunities of policy reuse. These representations help unseen tasks effectively leverage the knowledge from the most similar source tasks or reuse the knowledge from a mixing of source tasks, which is also consistent with the results in case study (Sec.**??**).

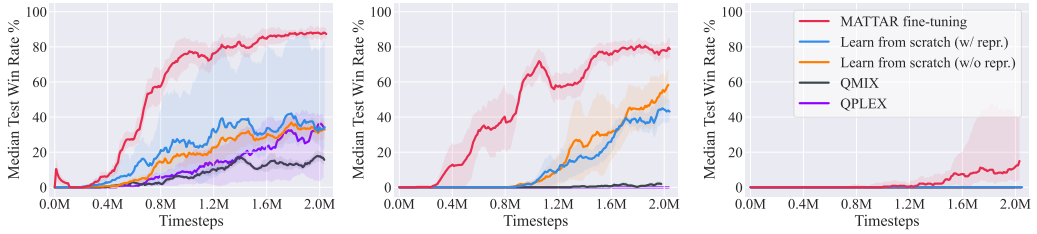
Ablations. To further investigate the role of task representations in our method, we design two ablations. (1) `0 task rep.` uses the trained MATTAR models but feed a zero-valued task representation into the policy network when transferring. (2) `w/o task rep.` denotes completely remove components related to task representation, including the forward model, the representation learning process, and the task representation in the policy network. Train and transfer the population-invariant policy. This ablation can reveal the generalization ability of the policy network itself.

Table 6: **Our method models task relationship and exploits it for knowledge transfer.** Task representations of unseen tasks are learned as a linear combination of source tasks' representations, whose coefficients are shown in this table. These weights reveal the encoded task relationship.

Source	Unseen Tasks	
	4m	10m_12m
5m	**0.61**	**0.43**
5m_6m	0.13	0.07
8m_9m	0.14	0.08
10m_11m	0.12	**0.43**

Source	Unseen Tasks	
	3s4z	3s5z_3s7z
2s3z	0.21	0.18
3s5z	**0.59**	0.21
3s5z_3s6z	0.21	**0.61**

Source	Unseen Tasks	
	MMM0	MMM6
MMM	**0.44**	0.30
MMM2	0.15	0.14
MMM4	0.40	**0.56**



(a) 10m_vs_12m (b) MMM6 (c) 3s5z_vs_3s7z

Figure 3: On unseen tasks: task representations provide a good initialization. Fine-tuning the policy can learn cooperation policies effectively than learning from scratch.

Results are shown in Tab. 1∼3. We can see that these ablations bring about a large drop in the performance of MATTAR on nearly all unseen tasks. For example, after trained on `2s3z`, `3s5z`, and `3s5z_vs_3s6z`, MATTAR achieves a win rate of 0.79 on `1s8z`, while `w/o task rep`. obtains a win rate of 0.12 and `0 task rep`. does not win any games. We can thus conclude that **task representations play an indispensable role in policy transfer**.

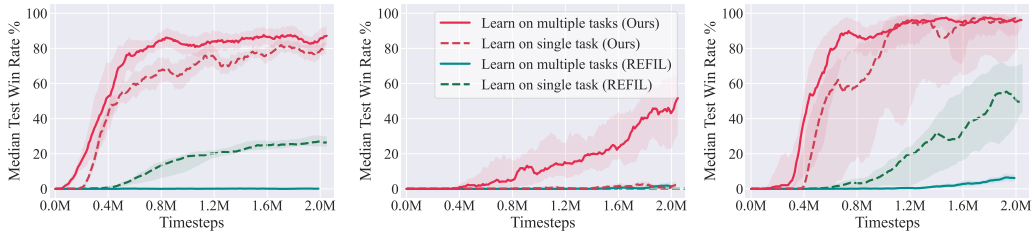
3.2 A good initialization for policy fine-tuning

When evaluating the performance of MATTAR on unseen tasks, we only train the task representations and reuse other parts of the policy. In this section, we investigate the performance of MATTAR after fine-tuning the policy. Specifically, on an unseen task, we first fix the representation explainer and train the task representation z for 50K timesteps. Then we fix the learned task representation and fine-tune the policy for 2M timesteps. We compare against learning from scratch in Fig. 3. Learning from scratch means the policy is randomly initialized and trained for 2M timesteps. We consider two versions for learning from scratch. The first version (`Learn from scratch w/o repr.`) totally removes task representations from the policy, while the second version (`Learn from scratch w/ repr.`) inserts the same task representation z as in the fine-tuning experiments.

We observe that the task representation and the reused policy provide a good initialization. For example, on `10m_vs_12m`, after fine-tuning for 2M timesteps, `MATTAR fine-tuning` converges to a win rate of around 0.86, while both versions of learning from scratch can only achieve a win rate of about 0.4. Furthermore, with the help of task representation, `MATTAR fine-tuning` solves `3s5z_vs_3s7z`, a task harder than the super hard map `3s5z_vs_3s6z` and cannot be solved by learning from scratch.

3.3 Benefits of multi-task learning

MATTAR adopts a scheme where multiple sources are learned simultaneously. Our aim is to leverage knowledge from more tasks and to be able to generalize the learned knowledge to a larger set of unseen tasks. Empirically, we find that this multi-task training setting helps not only unseen tasks but also the source tasks themselves.



(a) 5m_vs_6m (b) 3s5z_vs_3s6z (c) MMM2

Figure 4: On source tasks: MATTAR provides a framework for multi-agent multi-task learning. Training on multiple tasks helps improve performance than learning on a single task.

In Fig. 4, we present the performance of MATTAR on source tasks when training with multiple and a single source task. The experiments are carried out on three tasks from three different series. We can see that training on multiple tasks significantly boosts learning performance. For example, on `3s5z_vs_3s6z`, after 2M training samples, MATTAR with multiple tasks converges to the win rate of around 0.6, while training solely on this task can only achieve a win rate of about 0.05. Furthermore, we can also observe that MATTAR significantly outperforms the state-of-the-art deep multi-agent multi-task method (REFIL [18]). These results reveal that MATTAR also provides a good learning framework for multi-agent multi-task learning. It can leverage experience on other tasks to improve performance on a similar task.

4 Related Work

Multi-task learning with metadata [53, 58] approaches have been explored to exploit richer context information in the form of learned task embeddings, with a focus on task relation discovery. **(author?)** [35] use context information in high-level, under-specified, and unstructured natural language for multi-task learning. Compared to the settings studied in this paper, multi-agent problems pose additional challenges. Transition dynamics and reward functions in multi-agent tasks are influenced by the interaction among all agents so the relation between tasks has to be conditioned on the interaction pattern of multiple agents. Different from previous contextual multi-task RL methods, we include agent interaction modeling in the task representation learning module to deal with this problem. Nevertheless, we think incorporating natural language as prior high-level metadata of tasks can further improve the performance of our method in further work.

Related to our work, **context-based meta-RL** also reuses experience from multiple tasks and infers how to solve a new task from collected small amounts of experience [61]. For example, PEARL [31] performs online task inference by probabilistic filtering of latent task variables. Other meta-RL methods exploit the dynamics of recurrent networks for implicit task inference and fast adaptation [41, 10]. These methods tend to form dynamic variables or embeddings, while the context information is utilized to obtain fixed task representations in our approach. Besides, although their ability of task adaptation has been proved, context-based meta-RL has not been extended to multi-agent cases and cannot deal with varying numbers of actions and different lengths of observation inputs. For future work, it is important to discuss whether the inspiration of context-based meta-RL can further improve the performance of our multi-agent multi-task learning framework.

Learning task or skill embeddings for multi-task transfer reinforcement learning has also been extensively explored. When testing, **(author?)** [14] learn a new skill embedding by interpolating learned skill embeddings on source tasks by approximate variational inference. **(author?)** [2] also focus on learning skill embeddings but is conditioned on optimal Q-functions for different skills. **(author?)** [6] adopt a hierarchical framework and learns a high-level policy to control a low-level skill latent space. Similar to ours, **(author?)** [6] learn the low-level skill space by encoding experience trajectories and decoding states and actions. Another work that is similar to ours is **(author?)** [55], which conditions the policy on latent representations learned by dynamics and reward module. **(author?)** [23] learn task embeddings, fixes the policy, and updates the encoder at the test time. Compared to these methods, our task representation learning method differs by (1) it is specially designed for multi-agent settings and considers varying input sizes and intra-agent interaction; and (2) to the best of our knowledge, the alternatively fixed learning scheme is proposed for the first time. More contents include **multi-task**

reinforcement learning, multi-agent transfer learning, multi-agent representation learning, and modular RL are shown in App. A.

5 Conclusion

In this paper, we study cooperative multi-agent transfer reinforcement learning by learning task representations that model and exploit task relationship. Previous work on multi-agent transfer learning mainly deals with the varying population and input lengths, relying on the generalization ability of neural networks for cooperation knowledge transfer, ignoring the task relationship among tasks. Our method improves the transfer performance by learning task representations that capture the difference and similarities among tasks. When facing a new task, our approach only needs to obtain a new representation before transferring the learned knowledge. Taking advantage of task relationship mining, our proposed method MATTAR achieves the best transfer performance and other bonuses compared with multiple baselines. An important direction in the future is the transfer among tasks from different task distributions, and whether a linear combination of source tasks' representations can fully represent unseen tasks is also a valuable topic.

References

[1] Akshat Agarwal, Sumit Kumar, Katia Sycara, and Michael Lewis. Learning transferable cooperative behavior in multi-agent teams. In *International Conference on Autonomous Agents and MultiAgent Systems*, pages 1741–1743, 2020.

[2] Isac Arnekvist, Danica Kragic, and Johannes A. Stork. VPE: variational policy embedding for transfer reinforcement learning. In *International Conference on Robotics and Automation*, pages 36–42, 2019.

[3] Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pages 501–510, 2018.

[4] Åke Björck. Numerics of gram-schmidt orthogonalization. *Linear Algebra and Its Applications*, 197:297–316, 1994.

[5] Jiahan Cao, Lei Yuan, Jianhao Wang, Shaowei Zhang, Chongjie Zhang, Yang Yu, and De-Chuan Zhan. Linda: Multi-agent local information decomposition for awareness of teammates. *arXiv preprint arXiv:2109.12508*, 2021.

[6] John D. Co-Reyes, Yuxuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International Conference on Machine Learning*, pages 1008–1017, 2018.

[7] Bruno Castro da Silva, George Dimitri Konidaris, and Andrew G. Barto. Learning parameterized skills. In *International Conference on Machine Learning*, pages 1443–1450, 2012.

[8] Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In *International Conference on Robotics and Automation*, pages 3876–3881, 2014.

[9] Yunshu Du, Wojciech M. Czarnecki, Siddhant M. Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

[10] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[11] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

[12] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. In *International Conference on Learning Representations*, 2018.

[13] Aditya Grover, Maruan Al-Shedivat, Jayesh K. Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *International Conference on Machine Learning*, pages 1797–1806, 2018.

[14] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin A. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[15] Yann-Michaël De Hauwere, Peter Vrancx, and Ann Nowé. Learning multi-agent state space representations. In *International Conference on Autonomous Agents and MultiAgent Systems*, pages 715–722, 2010.

[16] Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. UPDeT: Universal multi-agent reinforcement learning via policy decoupling with transformers. In *International Conference on Learning Representations*, 2021.

[17] Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464, 2020.

[18] Shariq Iqbal, Christian A Schroeder De Witt, Bei Peng, Wendelin Böhmer, Shimon Whiteson, and Fei Sha. Randomized entity-wise factorization for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4596–4606, 2021.

[19] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference on Learning Representations*, 2019.

[20] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pages 7254–7264, 2018.

[21] Jens Kober, Andreas Wilhelm, Erhan Öztop, and Jan Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4):361–379, 2012.

[22] Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson. My body is a cage: The role of morphology in graph-based incompatible control. In *International Conference on Learning Representations*, 2021.

[23] Lin Lan, Zhenguo Li, Xiaohong Guan, and Pinghui Wang. Meta reinforcement learning with task embedding and shared policy. In *International Joint Conference on Artificial Intelligence*, pages 2794–2800, 2019.

[24] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17:1334–1773, 2016.

[25] Qian Long, Zihan Zhou, Abhinav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. Evolutionary population curriculum for scaling multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2019.

[26] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[27] Frans A Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

[28] Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P. How. Learning to teach in cooperative multiagent reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pages 6128–6136, 2019.

[29] Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A. Efros. Learning to control self-assembling morphologies: A study of generalization via modularity. In *Advances in Neural Information Processing Systems*, pages 2292–2302, 2019.

[30] Janith C Petangoda, Sergio Pascual-Diaz, Vincent Adam, Peter Vrancx, and Jordi Grau-Moya. Disentangled skill embeddings for reinforcement learning. *arXiv preprint arXiv:1906.09223*, 2019.

[31] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*, pages 5331–5340, 2019.

[32] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4292–4301, 2018.

[33] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. In *International Conference on Autonomous Agents and MultiAgent Systems*, pages 2186–2188, 2019.

[34] Felipe Leno da Silva and Anna Helena Reali Costa. Transfer learning for multiagent reinforcement learning systems. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 15(3):1–129, 2021.

[35] Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning*, pages 9767–9779, 2021.

[36] Yanchao Sun, Ruijie Zheng, Xiyao Wang, Andrew E Cohen, and Furong Huang. Transfer RL across observation feature spaces via model-based regularization. In *International Conference on Learning Representations*, 2022.

[37] Mihai Suteu and Yike Guo. Regularizing deep multi-task networks using orthogonal gradients. *arXiv preprint arXiv:1912.06844*, 2019.

[38] Yunzhe Tao, Sahika Genc, Jonathan Chung, Tao Sun, and Sunil Mallya. Repaint: Knowledge transfer in deep reinforcement learning. In *International Conference on Machine Learning*, pages 10141–10152, 2021.

[39] Dean Tjosvold. Cooperation theory and organizations. *Human relations*, 37(9):743–767, 1984.

[40] Samir Wadhwania, Dong-Ki Kim, Shayegan Omidshafiei, and Jonathan P. How. Policy distillation and value matching in multiagent reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8193–8200, 2019.

[41] Jane X Wang, Zeb Kurth-Nelson, Hubert Soyer, Joel Z Leibo, Dhruva Tirumala, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matt M Botvinick. Learning to reinforcement learn. In *CogSci*, 2017.

[42] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex dueling multi-agent Q-learning. In *International Conference on Learning Representations*, 2021.

[43] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018.

[44] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. ROMA: Multi-agent reinforcement learning with emergent roles. In *International Conference on Machine Learning*, pages 9876–9886, 2020.

[45] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. RODE: Learning roles to decompose multi-agent tasks. In *International Conference on Learning Representations*, 2021.

[46] Weixun Wang, Tianpei Yang Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. Action semantics network: Considering the effects of actions in multiagent systems. In *International Conference on Learning Representations*, 2020.

[47] Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. From few to more: Large-scale dynamic multiagent curriculum learning. In *AAAI Conference on Artificial Intelligence*, pages 7293–7300, 2020.

[48] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. DOP: Off-policy multi-agent decomposed policy gradients. In *International Conference on Learning Representations*, 2021.

[49] Annie Xie, Dylan P. Losey, Ryan Tolsma, Chelsea Finn, and Dorsa Sadigh. Learning latent representations to influence multi-agent interaction. In *Conference on Robot Learning*, pages 575–588, 2020.

[50] Zhiyuan Xu, Kun Wu, Zhengping Che, Jian Tang, and Jieping Ye. Knowledge transfer in multi-task deep reinforcement learning for continuous control. In *Advances in Neural Information Processing Systems*, pages 15146–15155, 2020.

[51] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In *Advances in Neural Information Processing Systems*, 2020.

[52] Tianpei Yang, Weixun Wang, Hongyao Tang, Jianye Hao, Zhaopeng Meng, Hangyu Mao, Dong Li, Wulong Liu, Yingfeng Chen, Yujing Hu, Changjie Fan, and Chengwei Zhang. An efficient transfer learning framework for multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 17037–17048, 2021.

[53] Qiang You, Ou Wu, Guan Luo, and Weiming Hu. Metadata-based clustered multi-task learning for thread mining in web communities. In *Machine Learning and Data Mining in Pattern Recognition*, pages 421–434, 2016.

[54] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, pages 5824–5836, 2020.

[55] Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling dynamics and reward for transfer learning. In *International Conference on Learning Representations*, 2018.

[56] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.

[57] Shenao Zhang, Li Shen, and Lei Han. Learning meta representations for agents in multi-agent reinforcement learning. *arXiv preprint arXiv:2108.12988*, 2021.

[58] Zimu Zheng, Yuqi Wang, Quanyu Dai, Huadi Zheng, and Dan Wang. Metadata-driven task relation discovery for multi-task learning. In *International Joint Conference on Artificial Intelligence*, pages 4426–4432, 2019.

[59] Tianze Zhou, Fubiao Zhang, Kun Shao, Kai Li, Wenhan Huang, Jun Luo, Weixun Wang, Yaodong Yang, Hangyu Mao, Bin Wang, Dong Li, Wulong Liu, and Jianye Hao. Cooperative multi-agent transfer learning with level-adaptive credit assignment. *arXiv preprint arXiv:2106.00517*, 2021.

[60] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020.

[61] Luisa M. Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: Variational Bayes-adaptive deep RL via meta-learning. *Journal of Machine Learning Research*, 22(289):1–39, 2021.