Efficient Multi-Task and Transfer Reinforcement Learning with Parameter-Compositional Framework

Lingfeng Sun*†1 Haichao Zhang*2 Wei Xu2 Masayoshi Tomizuka1

Abstract— **In this work, we investigate the potential of improving multi-task training and also leveraging it for transferring in the reinforcement learning setting. We identify several challenges towards this goal and propose a transferring approach with a parameter-compositional formulation. We investigate ways to improve the training of multi-task reinforcement learning which serves as the foundation for transferring. Then we conduct a number of transferring experiments on various manipulation tasks. Experimental results demonstrate that the proposed approach can have improved performance in the multi-task training stage, and further show effective transferring in terms of both sample efficiency and performance.**

I. INTRODUCTION

How to equip robots with various skills that are necessary for solving a diverse set of tasks is a long-time quest in the robotics field. Using reinforcement learning to empower robots with various skills is an active and valuable research direction. However, many robotics-related tasks, such as different manipulation skills, are commonly treated as isolated tasks and used individually for learning [1], [2]. While this setup is suitable for some particular research problems, learning one policy for each skill and learning from scratch every time is not practical for real-world robots, as it does not scale well with the number of skills to be acquired and introduces additional wear and tear if a physical robot is involved. This motivates the pursuit of more effective learning algorithms in research.

Reusing previously acquired knowledge in learning new tasks is attractive both in terms of learning efficiency and performance. A natural approach is to transfer the previously learned policy for the learning of a new skill. This is challenging as the transfer performance between tasks is strongly affected by relationships between individual tasks. A more effective approach for acquiring various skills is to solve the multi-task reinforcement learning (MTRL) problem [3]. Recent works compare multi-task learning and gradient-based meta learning [4], [5] under the supervised learning setting and conclude that for both methods, the learned representations are the key to transfer.

Inspired by the long-term quest on transferring learning in RL and encouraging progress on supervised multi-task learning and transferring [4], in this work, we aim to further investigate the potential of leveraging multi-task learning for transfer in the RL setting. There are some algorithmic necessities for bringing the benefits of MTRL to the transfer learning on new tasks:

1) **Performant MTRL Method.** The MTRL framework's high performance on a diverse set of manipulation tasks is a prerequisite. If MTRL fails to learn a well-performing policy, it is natural to expect that the transferring performance built upon it will be limited.
2) **Proper Architecture.** Not all MTRL frameworks are capable of being used for transferring to new tasks. Some task-specific information contained in the inputs of policy (*e.g.*, task-specific information such as task id) cannot be directly transferred to another new task where that piece of information is different.

In addition to algorithmic design, the performance of transfer is also strongly affected by the level of difficulty of the new task itself as well as its relation to the trained tasks. Few-shot meta reinforcement learning for new tasks typically works under a task distribution of limited variations [3], *e.g.* pushing the object to different goal locations. In practice, tasks typically come from a family with a more significant variation, *e.g.*, different types of manipulations skills as shown in Figure 1, posing a great challenge to these approaches both in terms of performance as well as learning efficiency, calling for the development of new approaches.

To overcome these challenges, one possible approach is to first design an MTRL training framework that succeeds in training high-performance policies for various tasks, with a transferable policy structure, and then design an efficient transfer algorithm that can benefit from the multi-task policies. For this purpose, we build on top of a recent parameter-compositional MTRL framework called PaCo [6]. PaCo maintains a policy subspace for all the individual task policies and by learning on multiple tasks with a compositional structure for task-adaptive parameter sharing.

Training multiple tasks with task-adaptive parameter sharing has two major benefits for learning skills: ***i)*** The MTRL training process benefits from similarities between tasks and is usually more efficient in both the number of parameters and the roll-out samples required to solve multiple tasks [7], [6]; ***ii)*** The task-adaptive parameter sharing schemes learned for solving different tasks gives us a posterior relation between tasks and can benefit both the multi-task training as well as transferring to unseen but similar tasks. The PaCo approach fits well in our context both because of its good MTRL performance and architecture design regarding the necessities discussed above. Firstly, it reaches state-of-the-art performance on the Meta-World MT10 [3] consists of 10 different manipulation tasks, each with multiple goals. Secondly, there is a clear separation of task-specific and task-agnostic parameters in the policy [6]; thus the task-agnostic

---

*Equal contribution. †Work done while interning at Horizon Robotics.

[1]UC Berkeley {`lingfengsun, tomizuka`}`@berkeley.edu`
[2]Horizon Robotics {`haichao.zhang, wei.xu`}`@horizon.ai`

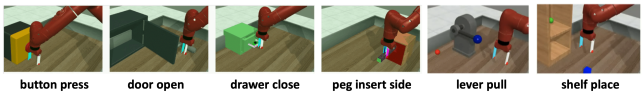


Fig. 1. Some example tasks from Meta-World benchmark [3].

part of PaCo policies can be naturally transferred for learning new tasks. The compositional vectors can be used to compute task relationships in policy subspace, as shown in Figure 2.

Motivated by these signs of progress, in this work, we focus on designing approaches for PaCo-based multi-task and transfer RL and investigating its potential. Our main contribution in this work can be summarized as follows:

1) **Improved MTRL**: further improve MTRL performance based on PaCo by incorporating a mechanism that adjusts task distributions during training;
2) **Transfer RL**: investigate the possibility and potential of MTRL for transferring, which can be viewed as further expanding recent results on multi-task learning for transferring from supervised setting [4], [5] to reinforcement learning;
3) **Empirical Validations**: validate the improved performance of transfer learning using the MTRL policy trained with the improved MTRL approach.

II. RELATED WORK

A. *Multi-Task and Transfer Reinforcement Learning*

Multi-Task RL. The goal of MTRL is to train a single policy $\pi_\theta(a|s)$ that can be applied to a set of different tasks. Every single task can be defined by a unique MDP, with differences in either state/action space, transition, or reward functions. In MTRL, we solve a set of MDPs from a task family using a universal policy $\pi_\theta(a|s, w)$, with w the task-relevant context information. By doing this, we can potentially have at least two benefits: ***i)*** better performance: by leveraging mutual connections between tasks, we can potentially improve the sample efficiency or final performance, *e.g.* through parameter sharing. ***ii)*** transferring: the learned MTRL policy could serve as a starting point to be transferred to a new task. While the first point has been extensively explored in MTRL literature [3], [8], [9], [10], [11], [7], [12], [13], the second one is less investigated [14], [15], [16], potentially due to many practical challenges in MTRL itself, *e.g.*, conflicts between tasks [9] and training stability [12], [6], limiting its effectiveness on transferring.

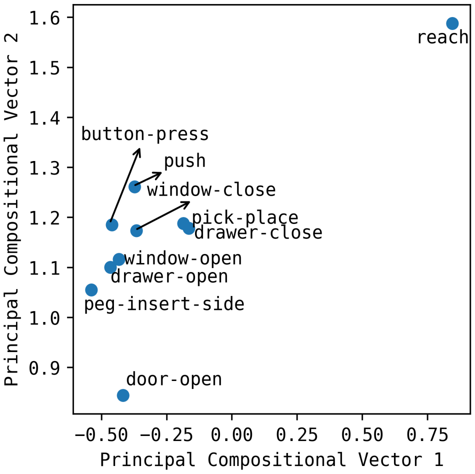
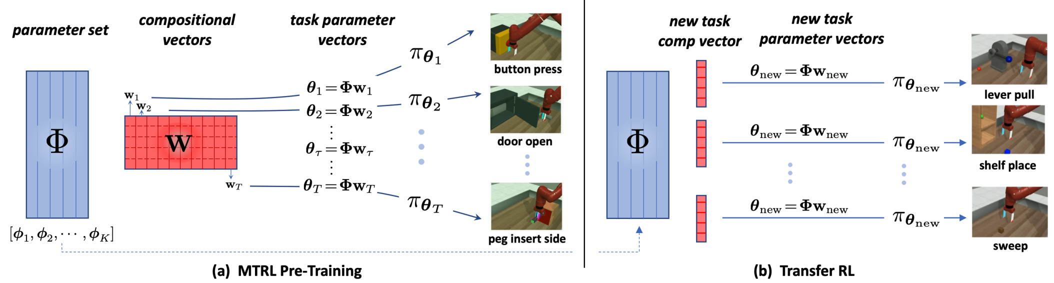


Fig. 2. PCA projections of $\{\mathbf{w}_\mathcal{T}\}$ learned using PaCo [6] on MT10 [3] benchmark. Each dot represents a projected $\mathbf{w}_\mathcal{T}$ vector of a task. Arrows are used to connect the cluttered dots with their respective task names.

Transfer RL. Transferring learning is a general approach to re-use previously learned knowledge in learning new situations. It is a powerful paradigm in supervised learning, leading to many successful approaches such as pre-training + fine-tuning that is widely used in computer vision [17], [18] and natural language processing [19], [20]. However, for reinforcement learning, simply fine-tuning a previously learned policy is not very useful [21], [22]. The reason is that different from supervised learning, RL requires sufficient exploration for successful learning. And a policy transferred from a previously trained task could be overly deterministic for exploring in the current task. There are also some previous attempts on leveraging MTRL for transferring, with focuses on goal-conditional/contextual policy [1], [23], [24], [25], [26], [27], policy modularity [15], [8] or reward learning [28]. For example, a large portion of works leverages the context-conditional form for generalization $\pi_\theta(a|s, w)$ where w denotes the task-relevant context information [24]. This approach relies on the the generalization ability of the network *w.r.t.* the input context and sufficient coverage of it during training for transferring to new tasks (new contexts).

B. *Connection with Meta Reinforcement Learning*

Meta Reinforcement Learning is one set of approaches for policy reusing and transferring in RL. Meta-learning or few-shot learning focuses on the fast adaptation of the model parameters with a few data points from the new task. One commonly used type of gradient-based formulation for meta-learning is a bi-level optimization form as in MAML [29], with the inner loop optimizing for the adapted parameters based on few-shot samples and the outer loop optimizing task loss given the adapted parameters. Recent progress shows that feature reuse is the main contributing factor in MAML and its bi-level can be reduced to an almost no inner loop (ANIL) version [5], [4]. It is important to note that while connected, gradient-based meta-learning [4], [5], [29] mainly focuses on fast adaptation from few-shot data. This scheme is effective when there are limited differences between source and target tasks (*e.g. reach different goal locations, move in different directions, etc.*). When applied to typical robotics tasks, where the difference between tasks could be large, the few-shot regime is typically not enough and has low performance in practice (*e.g.* the average success rate over transferred task is relatively low on MetaWorld benchmark [3]). We take an attempt towards solving challenging scenarios involving the transferring beyond simple variations such as goal locations, but between different skills as typical in real-world robotics, thus operating in a zone that is complementary to typical meta RL methods.



(a) MTRL Pre-Training

(b) Transfer RL

Fig. 3. Transfer with Parameter Compositional MTRL (TaCo). There are two phases in TaCo. (a) MTRL pre-training on a set of source tasks. (b) Transfer RL on a set of new tasks with the parameter set learned in the pre-training phase. Here we illustrate with transferring to each new task separately. Transferring jointly to a set of new tasks is also possible with TaCo.

III. REVIEW OF PARAMETER COMPOSITIONAL MULTI-TASK RL (PACO)

We briefly review the PaCo framework [6], which is also visually depicted in Figure 3 (a). Given a task $\tau \sim \mathcal{T}$, we use $\boldsymbol{\theta}_\tau \in \mathbb{R}^n$ to denote the vector of all the trainable parameters for task τ. The following decomposition for $\boldsymbol{\theta}_\tau$ is used:

$$\boldsymbol{\theta}_\tau = \boldsymbol{\Phi}\mathbf{w}_\tau, \tag{1}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_i, \cdots, \boldsymbol{\phi}_K] \in \mathbb{R}^{n\times K}$ denotes a matrix formed by a set of K parameter vectors $\{\boldsymbol{\phi}_i\}_{i=1}^K$ (referred to as *parameter set*), each of which has the same dimensionality as $\boldsymbol{\theta}_\tau$, *i.e.*, $\boldsymbol{\phi}_i \in \mathbb{R}^n$. $\mathbf{w}_\tau \in \mathbb{R}^K$ is a *compositional vector*, which is implemented as a trainable embedding for the task index τ. In essence, Eqn.(1) decomposes the parameters to two parts: *i)* task-agnostic $\boldsymbol{\Phi}$ and *ii)* task-aware $\mathbf{w}_\tau$.

When faced with multiple tasks as in MTRL, the decomposition in Eqn.(1) offers opportunities for flexible parameter sharing between tasks, by sharing the task-agnostic $\boldsymbol{\Phi}$ across all the tasks, while still ensuring task awareness via $\mathbf{w}_\tau$, leading to:

$$\begin{gathered}[\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_\tau, \cdots, \boldsymbol{\theta}_T] = \boldsymbol{\Phi}[\mathbf{w}_1, \cdots, \mathbf{w}_\tau, \cdots \mathbf{w}_T] \\ \boldsymbol{\Theta} = \boldsymbol{\Phi}\mathbf{W}.\end{gathered} \tag{2}$$

It also enables a way to stabilize MTRL training by masking out the exploding loss (when the loss for a particular task η is larger than a threshold ϵ, *i.e.* $\mathcal{L}_\eta > \epsilon$) and re-initialize $\mathbf{w}_\eta$ as [6]:

$$\mathbf{w}_\eta = \sum_{j\in\mathcal{V}} \beta_j \mathbf{w}_j, \quad \boldsymbol{\beta} = [\beta_1, \beta_2, \cdots] \sim \Delta^{|\mathcal{V}|-1} \tag{3}$$

where $\mathcal{V} \triangleq \{j | \mathcal{L}_j \leq \epsilon\}$, and $\boldsymbol{\beta}$ is uniformly sampled from a unit $|\mathcal{V}|-1$-simplex $\Delta^{|\mathcal{V}|-1}$. Our implementation is based on SAC [30].

IV. TACO: TRANSFER RL WITH IMPROVED PACO

A. Improved MTRL with Non-Uniform Task Sampling

During the MTRL stage, we sample tasks for training according to a certain distribution P for T tasks. There are many potential choices for P. A uniform distribution ($p_\tau = 1/T$) is typically used in standard MTRL [7], [12], [6]. A potential way to further improve the MTRL performance is by incorporating non-uniform task sampling during learning using prior or posterior knowledge of training task relationships. We group tasks into G, where $G = \{g_1, g_2, ...g_p\}$, where g_i is a set of tasks, and $\sum_{i=1}^p |g_p| = T$. Then we uniformly distribute sample steps among groups (for task τ in group g_i, $p_\tau = \frac{1}{|G||g_i|}$).

When we don't have access to pre-defined task groups based on difficulty or similarity before training, we can perform online adjustments on the task distribution (grouping) using the feedback during training(e.g., task relation in Figure 2). A simple approach to doing online adjustment is to perform clustering on task-dependent policy parameters θ_τ. In experiments on MT10, we use DBSCAN [31] to cluster the policy parameters of 10 tasks into groups $G = \{g_1, g_2, ...g_p\}$ and balance the distribution of task groups (referred to as TaCo-online).

Alternatively, when information on task groups is available, we can set the distribution of different task groups directly according to this information when desirable.

After a base parameter $\boldsymbol{\Phi}^*$ is obtained after MTRL training, it can be further used for learning new tasks as described in the next section.

B. Leveraging Previously Learned Shared Parameter Set for Transfer RL

Following the paradigm of using supervised multi-task learning for transferring [4], we train a transferable model via MTRL and then transfer the model to the learning on the target task. More concretely, we formulate the learning of a transferable model as follows:

$$\text{(Phase 1 MTRL) } \boldsymbol{\Phi}^*, \{\mathbf{w}^*_\tau\} = \arg\min_{\boldsymbol{\Phi}, \{\mathbf{w}_\tau\}} \sum_{\tau\in\mathcal{T}} \mathcal{L}(\boldsymbol{\Phi}, \mathbf{w}_\tau) \tag{4}$$

where $\mathcal{L}$ denotes the standard RL loss [30]. $\mathcal{T}$ denotes the set of tasks with $|\mathcal{T}| = T$. After training on a set of tasks $\mathcal{T}$, we obtain a base parameter $\boldsymbol{\Phi}^*$, where the superscript * is used to denote the optimal parameter.

While Eqn. (4) is general and is compatible with potentially many variants of MTRL approaches, in this work, we use the Parameter Compositional (PaCo) MTRL approach [6] due to its compelling performance and its suitable structure.

Upon learning of the new task, it is possible to reuse $\boldsymbol{\Phi}^*$ (and possibly fine-tune from it) and learn a $\mathbf{w}_{\text{new}}$ from scratch. To achieve this goal, we have

$$\text{(Phase 2 TransferRL) } \min_{\boldsymbol{\Phi}, \mathbf{w}_{\text{new}}} \mathcal{L}_{\text{new}}(\boldsymbol{\Phi}, \mathbf{w}_{\text{new}}),\ \boldsymbol{\Phi} \xleftarrow{\text{init}} \boldsymbol{\Phi}^* \tag{5}$$

This scheme of transferring is a path naturally induced from the formulation in Eqn.(4). where $\mathcal{L}_{\mathbf{\Phi},\mathbf{w}_{\text{new}}}$ denote the loss for task τ, $\mathbf{\Phi}$ is the base parameter shared across all tasks, and $\mathbf{w}_{\text{new}}$ is the parameter what will be adapted when transferred to new tasks. Compared with a standard MTRL formulation of $\min_\theta \sum_{\tau \in \mathcal{T}} \mathcal{L}(\theta)$, Eqn.(4) is different in that it explicitly separates all the parameters into two parts. This separation in parameters could have two implications: ***i)*** it facilitates a natural path to transferring (*c.f.* Eqn.(5)); ***ii)*** it introduces a perspective on the architecture design of the model. More concretely, it suggests that there should be two types of parameters; one serves for the purpose of retaining previously learned knowledge and can be reused as fixed or fine-tuned parameters for transferring, while the other is more specific that can be learned for each task, including the new task to be transferred to, for which purpose PaCo fits well. The transfer RL leveraging the parameter compositional form is shown in Figure 3 (b).

In the transfer RL stage, we reuse pre-trained parameters from the previous MTRL stage. The task-agnostic parameter $\mathbf{\Phi}^*$ consists of both policies value function networks for pre-trained multi-tasks. Using the pre-trained sub-policy space in training can make use of the task similarities between trained and transferred tasks and potentially improve the efficiency in exploration. However, for value functions, reusing the pre-trained parameters usually results in failure in RL training [22]. Therefore, during transfer, we only transfer the policy-related parameters from $\mathbf{\Phi}^*$ (denoted as $\mathbf{\Phi} \overset{\pi}{\longleftarrow} \mathbf{\Phi}^*$) as initialization and retrain the value function of the new task.

Exploration from Transfer Stage. The original SAC algorithm starts with collecting data with a random policy. To make full use of the transferred policy subspace $\mathbf{\Phi}$, we replace the policy with random sub-space policy $\pi_{\text{explore}} \triangleq \pi_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta} = \mathbf{\Phi}\tilde{\mathbf{w}}$ and $\tilde{\mathbf{w}} = \mathbf{W}\boldsymbol{\beta}$, $\boldsymbol{\beta} \sim \Delta^K$ and collect n_e steps of data. The value function for the new task is trained during this stage, but the policy parameters remain the same and are released after this stage.

We refer to our instantiation of the general transferring setting Eqn.(4)$\sim$(5) with the parameter compositional form Eqn.(1)$\sim$(2) incorporating improved training as Transfer with Parameter Compositional MTRL (TaCo). Detailed algorithmic procedures are presented in the Algorithms below.

V. EXPERIMENTS

In this section, we empirically test the performance of Multi-task RL and transfer RL on the Meta-World benchmark [3]. Meta-World benchmark is a robotic environment consisting of a number of distinct manipulation tasks, with example tasks shown in Figure 1. All the tasks share the same action space and the same dimension of state space, but certain dimensions in the state space represent different semantic meanings across tasks. We used the Multi-task benchmark from Meta-World [3] with random goals, i.e., each manipulation task is configured with random goals, and the 10-task benchmark is referred to as MT10.

Using experiments on Meta-World, we would like to demonstrate: ***i)*** TaCo's state-of-the-art performance for MTRL, and ***ii)*** TaCo's advantages and effectiveness in transfer learning on unseen tasks using a pre-trained multi-task policy. To make the presentation clear, we present results on each stage in subsection V-A and V-B respectively.

Algorithm 1 TaCo: MTRL Pre-Training Phase

```
Input: parameter-set size $K$, loss threshold $\epsilon$, learning rate $\lambda$, task distribution $P$
while termination condition not satisfied do
    # environmental interaction
    sample a task from $P$ upon environment reset and unroll it with the current policy; save task-id and transitions to a replay buffer
    # policy training
    for each gradient step do
        sample a batch of training tasks $\mathcal{T}$ and associated data from buffer
        $\mathcal{L}_\tau \leftarrow \mathcal{L}_\tau(\boldsymbol{\theta}_\tau),\ \boldsymbol{\theta}_\tau = \mathbf{\Phi}\mathbf{w}_\tau \quad \forall \tau \in \mathcal{T}$
        (loss maskout)  $\mathcal{L}_\eta \leftarrow 0$  if $\mathcal{L}_\eta > \epsilon$
        $\mathcal{L}_{\mathbf{\Theta}} \leftarrow \sum_\tau \mathcal{L}_\tau$
        $\mathbf{\Phi} \leftarrow \mathbf{\Phi} - \lambda \nabla_{\mathbf{\Phi}} \mathcal{L}_{\mathbf{\Theta}}$
        $\mathbf{w}_\tau \leftarrow \mathbf{w}_\tau - \lambda \nabla_{\mathbf{w}_\tau} \mathcal{L}_\tau(\mathbf{w}_\tau)$
        (w-reset)  $\mathbf{w}_\eta \leftarrow$ Eqn.(3)  if $\mathcal{L}_\eta > \epsilon$
    end for
end while
```

Algorithm 2 TaCo: Transfer RL Phase

```
Input: MTRL parameter set $\mathbf{\Phi}^*$
Initialize $\mathbf{\Phi} \overset{\pi}{\longleftarrow} \mathbf{\Phi}^*$
Collect $n_e$ steps of data with $\pi_{\text{explore}}$, update value functions.
while termination condition not satisfied do
    $\boldsymbol{\theta}_{\text{new}} = \mathbf{\Phi}\mathbf{w}_{\text{new}}$
    $\mathcal{L}_{\text{new}} \leftarrow \mathcal{L}_{\text{new}}(\boldsymbol{\theta}_{\text{new}})$
    $\mathbf{\Phi} \leftarrow \mathbf{\Phi} - \lambda \nabla_{\mathbf{\Phi}} \mathcal{L}_{\text{new}}$
    $\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{new}} - \lambda \nabla_{\mathbf{w}_{\text{new}}} \mathcal{L}_{\text{new}}$
end while
```

A. MTRL Experiments and Results

Training Setting. For MTRL training on MT10 in Meta-World, we follow the settings introduced in [12] and use ***i)*** 10 parallel environments, ***ii)*** 20 million environment steps for the 10 tasks together (2 million per task), ***iii)*** repeated training with 10 different random seeds for each method.

Baselines. For MTRL benchmarks in Meta-World, we compare it against *(i)* **Multi-task SAC**: extended SAC [30] for MTRL with one-hot task encoding; *(ii)* **Multi-Head SAC**: SAC with shared a network apart from the output heads, which are independent for each task; *(iii)* **SAC+FiLM**: the task-conditional policy is implemented with the FiLM module [32] on top of SAC; *(iv)* **PCGrad** [9]: a representative method for handling conflicting gradients during multi-task learning via gradient projection during optimization; *(v)* **Soft-Module** [7]: which learns a routing network that guides the soft combination of modules (activations) for each task; *(vi)* **CARE** [12]: leveraging additional task-relevant metadata for state representation. *(vii)* **PaCo** [6]: Parameter-Compositional MTRL with uniform task distribution.[1]

Evaluation Metrics and Results. The evaluation metric for the MTRL policy is based on the success rate of the policy for all the tasks. For Meta-World benchmarks, we evaluate

---

[1] Note that some experiments reported in the works mentioned above are implemented and evaluated on Meta-World-V1 and/or with fixed-goal setting. We adapt these methods and experiment on Meta-World-V2.

TABLE I
RESULTS ON META-WORLD [3] MT10 (20M STEPS).

Methods	Success Rate (%) (mean ± std)
Multi-Task SAC [3]	62.9 ± 8.0
Multi-Head SAC [3]	62.0 ± 8.2
SAC + FiLM [32]	58.3 ± 4.3
PCGrad [9]	61.7 ± 10.9
Soft-Module [7]	63.0 ± 4.2
CARE [12]	76.0 ± 6.9
PaCo [6]	85.4 ± 4.5
TaCo (Ours)	**90.7** ± **3.6**

each skill with 5 episodes of different sampled goals using the final policy. The success rate is then averaged across all the skills. There is randomness in the MTRL training; therefore instead of picking the maximum evaluation success rate across training, we use the policy at 20M total environment steps (2M per task) and average across multiple trains for a fair evaluation. The 20M total environmental steps (i.e. 2M per task) metric is chosen based on previous work [12], [6] which empirically observed that all the methods have mostly saturated in their performance under this setting. We report the mean and standard deviation of the success rate in Table I. An improvement on the MT10 benchmark is observed on PaCo [6] compared to previous MTRL methods, demonstrating the advantage of the compositional structure and the reset during training. A further improvement of TaCo using pre-set task distribution during the MTRL training is also observed, showing the effectiveness of using task difficulty to adjust the task distribution of different tasks.

Adjustment of Task Distribution. An important component that contributes to the significant improvement of TaCo on MTRL is the adjusted sample distribution according to task difficulties, as discussed in Section IV. We investigate the impact of this factor empirically in this section. The comparison between PaCo (uniform task distribution) and TaCo-online is shown in Table II. We evaluate the performance at 20M step and also provide an evaluation at 30M steps to further inspect the potential performance changes w.r.t. increased environmental steps. As expected, the TaCo-online adjustment using task grouping has an improvement over the PaCo baseline. When additional information is available for designing task distributions, TaCo can further improve its performance. To demonstrate this, we empirically construct a task distribution as follows.

TABLE II
DIFFERENT TASK DISTRIBUTIONS IN MTRL TRAINING (MT10 TASKS).

Methods	Success Rate (%)	
	20M steps	30M steps
PaCo [6]	85.4 ± 4.5	85.7 ± 4.1
TaCo-online	86.4 ± 4.3	89.3 ± 1.2
TaCo	**90.7** ± **3.6**	**94.0** ± **5.4**

We set the probability of "*pick-place, peg-insert-side, drawer-open*" two times the probability of other tasks, based on the descriptions of each task in MT10.[2] As shown in Table II, TaCo achieves higher success rate with this task distribution. As a reference, the environment steps required for single tasks to converge are shown in Figure 4.

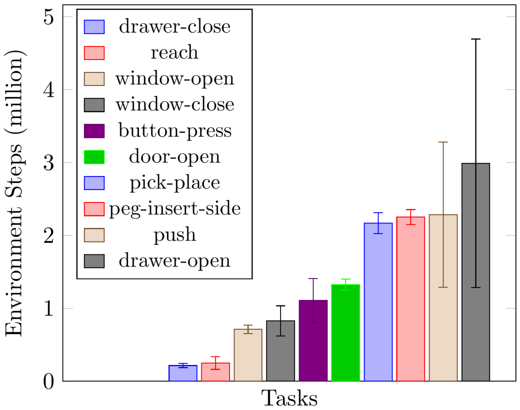


Fig. 4. Environment steps required for training a successful single-task policy for different tasks.

B. Transfer Experiments and Results

Training Setting. For transfer learning with MTRL policy, we use the best policy (in terms of average success rate) during MTRL training as the base policy. We use 10 parallel environments and 5 random seeds for each transfer experiment. During the transfer, we set the initial exploration steps for data collection as $n_{\text{warm}} = 20$K.

Evaluation Metrics. From Figure 4, we observe that the MT10 benchmark actually covers a wide range of tasks in terms of single-task difficulty. These required environment steps can be used as calibration. The following metrics are used in reporting the results:

1) **required environmental steps** n: the number of environmental steps (M, in million) required to reach an evaluation success rate of 0.9, before $n_{\max}$ steps ($n_{\max} =$ 6M steps in evaluation)
2) **(transfer) success** α: for every single task, we train for 6M environmental steps. It is labeled as a success if the training reaches an evaluation success rate of 0.9.
3) **relative transfer cost** (main metric): ratio of the success-normalized environment steps ($\frac{n}{\alpha}$) used to train a new task with and without the transferring process: $\frac{n_{\text{transfer}}}{\alpha_{\text{transfer}}} / \frac{n_{\text{scratch}}}{\alpha_{\text{scratch}}}$.

We repeat the training 5 times with different random seeds and use the average as its empirical estimation for each metric. Results are summarized in Figure 6.

Illustrative Transferring within MT10 Tasks. To get an illustration of how multi-task policies can help train new tasks more efficiently, we first design some transfer learning experiments within the MT10 tasks. In Table III, we show the transfer performance of pre-training on three tasks and transfer to a new task in MT10. In general, transfer from MT policy can benefit training on new tasks, the improvement

---

[2] The "difficulty" of task is based on our intuition on reward sparsity based on task description, we do not assume single task training statistics as prior knowledge like [22].

can be relatively large for harder tasks if diverse skills are involved in pre-training. At the same time, there are cases where the new tasks are simple and unrelated to the pre-trained tasks. The transfer process may cost more steps compared to training from scratch. This phenomenon matches the observation on the transfer metric under a different setting (transferring between single tasks) reported in [22], where transfer cost between individual tasks could sometimes increase to a value larger than 1 depending on the task relationships.

Transfer to New Tasks using MT Policy from MT10. Compared to arbitrarily selected task combinations, the MT10 benchmark covers a set of different skills in manipulation with different difficulties. We now test its performance on transferring to some challenging unseen new tasks.

We chose four tasks (*sweep*, *sweep-into*, *lever-pull*, and *shelf-place*) from Meta-World that are not in MT10, but are representative of the rest of Meta-World and experience poor performance during few-shot transfer in reported results [3]). Training from scratch for these tasks all require more than 1 million steps, which means they are not easy tasks.[3] Besides **SAC-scratch** [30], we further compare with a recent transferring approach called **ANIL** [5], which is an *Almost No Inner Loop* version of MAML [29]. It learns the MTRL policy with a multi-head form and then adapts the output layer for transferring to the new task. For reference, it achieves a 70% success rate at convergence on the trained tasks on MT10.

In Table IV, we show the transfer cost and the transfer success of transferring from a fully trained (reach 100% success for all MT10 tasks) TaCo MTRL policy. For TaCo transfer, we observe an improvement in both transfer cost and transfer success. For tasks that occasionally fail in training from scratch, transferring with TaCo improves their stability. ANIL [5] has a larger transfer cost compared to from scratch on some tasks, and cannot always match the success rate of training from scratch. This is potentially due to its limited performance during the MTRL stage on MT10, and also the large deviation between trained and new tasks compared to typical meta-learning settings. This emphasizes the importance of using a performant MTRL method and the design of an architecture that is effective for handling large task derivations for effective transferring.

Transfer Performance on Hard Tasks. Shelf-place is a hard task that requires a huge number of steps in single-task training and is not stable. Nearly half of the experiments launched for the shelf-place task fail to converge to a successful policy due to a lack of exploration or loss explosion. Therefore, compared to other tasks, where efficiency of learning serves as the main metric for transfer learning, stability improvement, as well as the environment step efficiency, are both important. Figure 5 shows the comparison of the average success rate curve between training from the transferred policy and training from scratch. Note that the improvement we can see from using transferred multi-task policy largely benefits from the good exploration performed by random policies in the trained policy subspace. This is not guaranteed for arbitrary new tasks and multi-task sets. There are chances that the trained multi-task policy doesn't benefit the new training when the new task is fundamentally different from the trained tasks. This is also a non-negligible challenge we have for transfer learning. However, by introducing more diversity in the pre-trained task set, it is more likely a new task can benefit from the general multi-task policy.

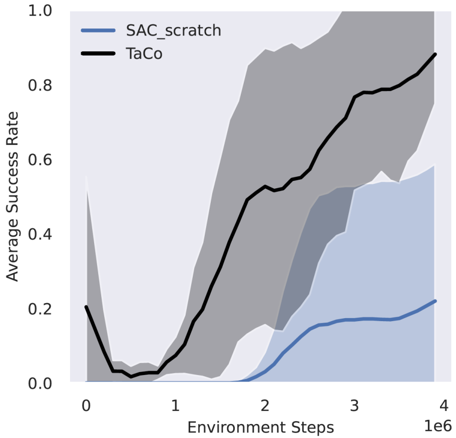


Fig. 5. Comparison of training from scratch and transfer from TaCo-MT10 on *shelf-place*, which is a hard task.

More Experiments on TaCo Transfer. We have the following observations from the results in Figure 6:

1) **reduced environment steps**: TaCo transfer can reduce the training cost by using fewer environment steps to converge on most tasks, demonstrating the first benefit of transferring;
2) **improved success rate**: TaCo also shows an improved success rate, which is more distinct for hard tasks, where training from scratch has a success rate lower than those of easy tasks, implying the second benefit;[4]
3) **reduced relative cost**: in terms of the relative transfer cost defined above, which is a metric that incorporates both the aspect of transfer cost and success rate, TaCo also shows clear improvements compared with SAC-scratch. The reduction is especially clear for difficult tasks (*e.g.* tasks with environmental steps>2M according to Figure 6(a)). For example, for the *stick-pull* task, SAC-scratch spends roughly 6M environment steps with a 0.2 success rate. TaCo on the other hand, increases the success rate to 0.8 with fewer environmental steps, leading to a significant reduction in transfer cost.
4) **transfer performance variations**: while there are clear improvements in most tasks, we do see in tasks like *handle-pull*, training from scratch is already stable in succeeding and is more efficient in environment steps. This is potentially due to its low task difficulty and the lack of task similarity between the MT10 task set and

---

[3] Task order from simple to hard based on single task training success rate and required environment steps is: *sweep-into*, *sweep*, *lever-pull*, *shelf-place*.

[4] The relative difficulty of a task can be roughly measured by the number of environment steps required for obtaining a well-performing policy.

TABLE III

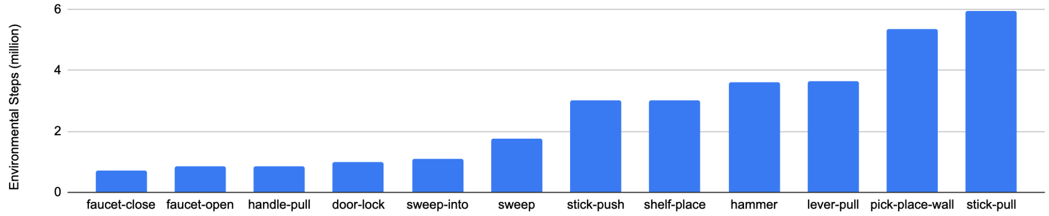
TRANSFER COST AND SUCCESS RATE BETWEEN SELECTED TASKS IN MT10.

Trained Tasks	New Task	Transfer Cost (↓) / Trans. Success (↑)	
		SAC-scratch [30]	TaCo
reach, door-open, drawer-open	*drawer-close*	1.0/1.0	**0.727/1.0**
reach, door-open, drawer-open	*window-close*	1.0/1.0	**0.879/1.0**
reach, door-open, drawer-open	*window-open*	**1.0/1.0**	1.126/1.0
reach, push, peg-insert	*door-open*	1.0/1.0	**0.528/1.0**

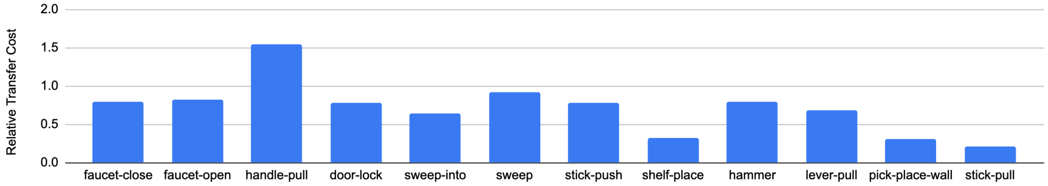
TABLE IV

TRANSFER COST AND SUCCESS RATE FROM MT10 TACO POLICY TO NEW TASKS.

Trained Tasks	New Task	Transfer Cost (↓) / Trans. Success(↑)		
		SAC-scratch [30]	ANIL [5]	TaCo
MT10 (*reach, push, drawer-open, drawer-close, window-open, window-close, button-press, door-open, pick-place, peg-insert-side*)	*sweep*	1.0/1.0	**0.826/1.0**	0.922/1.0
	shelf-place	1.0/0.4	1.868/0.4	**0.809/1.0**
	sweep-into	1.0/1.0	1.758/1.0	**0.654/1.0**
	lever-pull	1.0/0.8	1.232/0.6	**0.849/1.0**



SAC-scratch



TaCo

Fig. 6. More transfer results. (a) The number of environmental steps required for SAC-scratch for 12 different tasks. Note that only successfully trained seeds are used. (b) Relative Transfer Cost of TaCo over SAC-scratch. Training success rate and step cost for SAC and TaCo are both considered.

handle-pull. How to characterize the similarities between trained and new tasks is another open research question and an interesting topic for future work.

C. Additional Example: Transferring with Fixed Φ

An interesting attempt in TaCo framework is to verify if we can achieve successful transfer with the *fixed* task-agnostic policy parameters learned from pre-trained multiple tasks. In this setting, we fix the learned $\mathbf{\Phi}$ parameters and learn only a new $\mathbf{w}$ vector for the unseen task, meaning the new policy lies in the trained multi-task subspace and uses no additional policy parameters except a new $\mathbf{w}$ parameter representing its position in the subspace. The results are summarized in Table V. The performance under this setting depends more heavily on the relationships between trained and extension/new tasks. The relation analysis of tasks and sequencing during training is an open problem for robotic learning, and because of this, we report the effectiveness of TaCo's no-cost extension ability as an additional example, acknowledging the non-negligible efforts required to fully investigate this orthogonal direction in the future.

TABLE V

TACO TRANSFERRING WITH FIXED Φ.

Trained Tasks	Extension Task	Success Rate
reach, door-open, drawer-open	*drawer-close*	75 ± 9
window-open, window-close, door-open	*door-close*	90 ± 5

VI. CONCLUSIONS AND FUTURE WORK

In this work, we present TaCo, an approach that leverages MTRL for transfer reinforcement learning. With TaCo, we can achieve better MTRL performance and effective transfer learning of new tasks on top of it. We show encouraging results along the direction of leveraging multi-task learning for transferring [4] under a number of different settings in the RL domain. Successful transfer hinges on the relationships between the source tasks and the new task. How to quantify the diversity of the source tasks and the relationship with the new task is a challenging and interesting future work.

In this work, we have used task id to represent different tasks following protocols in standard MTRL environments [3]. While this setting has minimal assumptions on the tasks, it is limited by its power in encoding task properties and their mutual relationships. Moreover, it incurs additional labeling costs. It is possible to improve by using language-based task description [12] or unsupervised task relationship discovery. Another limitation is the evaluation protocol. To the best of our knowledge, there is no standardized evaluation protocols available for MTRL-based transferring in the community, including the selection of the base trained task and new task. In this work, we used an existing MTRL benchmark with a reasonable diversity, with the hope to retain some relevance to the new task. How to achieve this in a more principled way requires further investigation.

While this work mainly focuses on the algorithmic aspects of pre-training and transferring schemes, it is important to note that transfer learning is a very challenging task that involves many other aspects, including task relations, imbalances, and long-term dependencies. Multi-task and transfer learning in task space needs to combine with sim-to-real techniques considering vision representation [33] and task-dependent constraints [34] to work on real robots. We leave the investigations on these aspects as future work.

TABLE VI

General MTRL hyper-parameters on MT10

Hyper-parameter	Value
batch size	1280
number of parallel env	10
MLP hidden layer size	[400, 400, 400]
policy learning rate	3e-4
Q learning rate	3e-4
discount	0.99
episode length	150
exploration steps	1500
replay buffer size	1e6

TABLE VII

TaCo specific hyper-parameters

Hyper-parameter	Value
extreme loss threshold ϵ	3e3
param-set size K	5
compositional vector w learning rate	3e-4
transfer stage exploration step n_e	20000

References

[1] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, “Multi-goal reinforcement learning: Challenging robotics environments and request for research,” *CoRR*, vol. abs/1802.09464, 2018.

[2] Q. Gallouédec, N. Cazin, E. Dellandréa, and L. Chen, “panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning,” *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*, 2021.

[3] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning,” in *Conference on Robot Learning*, 2019.

[4] H. Wang, H. Zhao, and B. Li, “Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation,” in *International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., 2021.

[5] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, “Rapid learning or feature reuse? towards understanding the effectiveness of MAML,” in *International Conference on Learning Representations*, 2020.

[6] L. Sun, H. Zhang, W. Xu, and M. Tomizuka, “PaCo: Parameter-compositional multi-task reinforcement learning,” in *Advances in Neural Information Processing Systems*, 2022.

[7] R. Yang, H. Xu, Y. WU, and X. Wang, “Multi-task reinforcement learning with soft modularization,” in *Advances in Neural Information Processing Systems*, 2020.

[8] J. Andreas, D. Klein, and S. Levine, “Modular multitask reinforcement learning with policy sketches,” in *ICML*, 2017.

[9] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, “Gradient surgery for multi-task learning,” in *Advances in Neural Information Processing Systems*, 2020.

[10] C. D’Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, “Sharing knowledge in multi-task deep reinforcement learning,” in *International Conference on Learning Representations*, 2020.

[11] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, “Distral: Robust multitask reinforcement learning,” in *Advances in Neural Information Processing Systems*, 2017.

[12] S. Sodhani, A. Zhang, and J. Pineau, “Multi-task reinforcement learning with context-based representations,” in *International Conference on Machine Learning*, 2021.

[13] Dmitry Kalashnikov and Jacob Varley and Yevgen Chebotar and Benjamin Swanson and Rico Jonschkowski and Chelsea Finn and Sergey Levine and Karol Hausman, “MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale,” *CoRR*, 2021.

[14] E. Parisotto, L. J. Ba, and R. Salakhutdinov, “Actor-mimic: Deep multitask and transfer reinforcement learning,” in *International Conference on Learning Representations*, 2016.

[15] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, “Learning modular neural network policies for multi-task and multi-robot transfer,” *CoRR*, vol. abs/1609.07088, 2016.

[16] Julian, Ryan and Swanson, Benjamin and Sukhatme, Gaurav and Levine, Sergey and Finn, Chelsea and Hausman, Karol, “Never Stop Learning: The Effectiveness of Fine-Tuning in Robotic Reinforcement Learning,” in *Conference on Robot Learning*, 2021.

[17] W. Ge and Y. Yu, “Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[18] S. Kornblith, J. Shlens, and Q. V. Le, “Do better imagenet models transfer better?” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[19] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018.

[20] A. Radford and K. Narasimhan, “Improving language understanding by generative pre-training,” 2018.

[21] V. Campos, P. Sprechmann, S. Hansen, A. Barreto, S. Kapturowski, A. Vitvitskyi, A. P. Badia, and C. Blundell, “Coverage as a principle for discovering transferable behavior in reinforcement learning,” *CoRR*, vol. abs/2102.13515, 2021.

[22] K. R. Zentner, R. Julian, U. Puri, Y. Zhang, and G. S. Sukhatme, “A simple approach to continual learning by transferring skill parameters,” *CoRR*, vol. abs/2110.10255, 2021.

[23] M. E. Taylor and P. Stone, “Transfer learning for reinforcement learning domains: A survey,” *Journal of Machine Learning Research*, vol. 10, no. 56, pp. 1633–1685, 2009.

[24] J. Oh, S. P. Singh, H. Lee, and P. Kohli, “Zero-shot task generalization with multi-task deep reinforcement learning,” in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[25] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver, “Successor features for transfer in reinforcement learning,” in *Advances in Neural Information Processing Systems*, 2017.

[26] C. Ma, D. R. Ashley, J. Wen, and Y. Bengio, “Universal successor features for transfer reinforcement learning,” *CoRR*, 2020.

[27] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, “Hindsight experience replay,” in *Advances in Neural Information Processing Systems*, 2017.

[28] Se-Wook Yoo and Seung-Woo Seo, “Learning Multi-Task Transferable Rewards via Variational Inverse Reinforcement Learning,” in *International Conference on Robotics and Automation*.

[29] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International Conference on Machine Learning*, 2017.

[30] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International Conference on Machine Learning*, 2018.

[31] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *International Conference on Knowledge Discovery and Data Mining*, 1996.

[32] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, “FiLM: Visual reasoning with a general conditioning layer,” *CoRR*, vol. abs/1709.07871, 2017.

[33] X. Zhu, L. Sun, Y. Fan, and M. Tomizuka, “6-dof contrastive grasp proposal network,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6371–6377.

[34] J.-W. Wang, L. Sun, X. Zhu, Q. Qian, and M. Tomizuka, “A simple approach for general task-oriented picking using placing constraints,” 2023.