Continual Task Allocation in Meta-Policy Network via Sparse Prompting

Yijun Yang [1 2] Tianyi Zhou [3] Jing Jiang [2] Guodong Long [2] Yuhui Shi [1]

Abstract

How to train a generalizable meta-policy by continually learning a sequence of tasks? It is a natural human skill yet challenging to achieve by current reinforcement learning: the agent is expected to quickly adapt to new tasks (plasticity) meanwhile retaining the common knowledge from previous tasks (stability). We address it by "**Co**ntinual **T**ask **A**llocation via **S**parse **P**rompting (**CoTASP**)", which learns over-complete dictionaries to produce sparse masks as prompts extracting a sub-network for each task from a meta-policy network. CoTASP trains a policy for each task by optimizing the prompts and the sub-network weights alternatively. The dictionary is then updated to align the optimized prompts with tasks' embedding, thereby capturing tasks' semantic correlations. Hence, relevant tasks share more neurons in the meta-policy network due to similar prompts while cross-task interference causing forgetting is effectively restrained. Given a meta-policy and dictionaries trained on previous tasks, new task adaptation reduces to highly efficient sparse prompting and sub-network finetuning. In experiments, CoTASP achieves a promising plasticity-stability trade-off without storing or replaying any past tasks' experiences. It outperforms existing continual and multi-task RL methods on all seen tasks, forgetting reduction, and generalization to unseen tasks. Our code is available at https://github.com/stevenyangyj/CoTASP

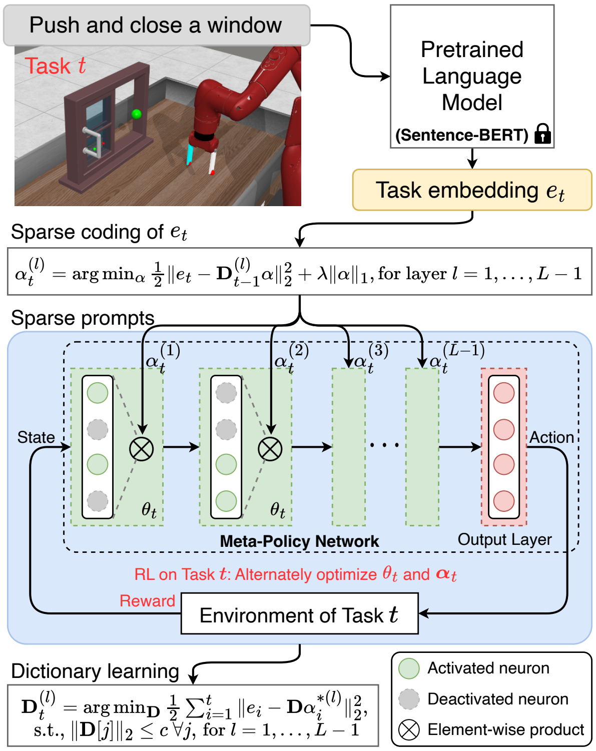

Push and close a window
Task t
Pretrained Language Model
(Sentence-BERT)
Task embedding e_t
Sparse coding of e_t
$\alpha_t^{(l)} = \arg\min_\alpha \frac{1}{2}\|e_t - \mathbf{D}_{t-1}^{(l)}\alpha\|_2^2 + \lambda\|\alpha\|_1$, for layer $l = 1, \dots, L-1$
Sparse prompts
$\alpha_t^{(1)}$
$\alpha_t^{(2)}$
$\alpha_t^{(3)}$
$\alpha_t^{(L-1)}$
State
θ_t
θ_t
Action
Meta-Policy Network
Output Layer
RL on Task t: Alternately optimize θ_t and α_t
Reward
Environment of Task t
Dictionary learning
$\mathbf{D}_t^{(l)} = \arg\min_{\mathbf{D}} \frac{1}{2}\sum_{i=1}^{t}\|e_i - \mathbf{D}\alpha_i^{*(l)}\|_2^2$,
s.t., $\|\mathbf{D}[j]\|_2 \le c\ \forall j$, for $l = 1, \dots, L-1$
Activated neuron
Deactivated neuron
Element-wise product

Figure 1: Main steps and components of CoTASP.

1. Introduction

Although reinforcement learning (RL) has demonstrated excellent performance on learning a single task, e.g., playing Go (Silver et al., 2016), robotic control (Schulman et al., 2017; Degrave et al., 2022), and offline policy optimization (Yu et al., 2020; Yang et al., 2022), it still suffers from catastrophic forgetting and cross-task interference when learning a stream of tasks on the fly (McCloskey & Cohen, 1989; Bengio et al., 2020) or a curated curriculum of tasks (Fang et al., 2019; Ao et al., 2021; 2022). So it is challenging to train a meta-policy that can generalize to all learned tasks or even unseen ones with fast adaptation, which however is an inherent skill of human learning. This problem has been recognized as continual or lifelong RL (Mendez & Eaton, 2022) and attracted growing interest in recent RL research.

A primary and long-standing challenge in continual RL is the plasticity-stability trade-off (Khetarpal et al., 2022): the RL policy on the one hand needs to retain and reuse the knowledge shared across different tasks in history (stability) while on the other hand can be quickly adapted to new tasks without interference from previous tasks (plasticity). Addressing this challenge is vital to improving the efficiency of

[1] Southern University of Science and Technology [2] University of Technology Sydney [3] University of Maryland, College Park. Correspondence to: Tianyi Zhou <tianyi@umd.edu>, Jing Jiang <jing.jiang@uts.edu.au>, Yuhui Shi <shiyh@sustech.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

continual RL and the generalization capability of its learned policy. A meta-policy with better stability can reduce the necessity of experience replay and its memory/computation cost. Moreover, the required network size can be effectively reduced if the meta-policy can manage knowledge sharing across tasks in a more compact and efficient manner. Hence, stability can greatly improve the efficiency of continual RL when the number of tasks increases (Shin et al., 2017; Li & Hoiem, 2018). Furthermore, better plasticity indicates faster adaptation and generalization to new tasks.

In order to directly address the plasticity-stability trade-off and overcome the drawbacks of prior work, we study how to train a meta-policy network in continual RL. Then, given only a textual description of a previously learned or unseen task, a specific policy can be automatically and efficiently extracted from the meta-policy. This is in the same spirit of prompting in recent large language models (Li & Liang, 2021; Liu et al., 2021) but differs from existing methods that randomly select task policies (Rajasegaran et al., 2019; Mirzadeh et al., 2020) or independently optimize a policy for each task from scratch (Serrà et al., 2018; Kang et al., 2022). To this end, we propose to learn layer-wise dictionaries along with the meta-policy network to produce sparse prompts (i.e., binary masks) for each task, which extract a sub-network from the meta-policy to be the task-specific policy. We call this approach "**Co**ntinual **T**ask **A**llocation via **S**parse **P**rompting (**CoTASP**)".

As illustrated by Fig. 1, given each task t, the prompt α_t is generated by sparse coding of its task embedding e_t under dictionary $\mathbf{D}_t$ and used to allocate a policy sub-network θ_t from the meta-policy. Then α_t and θ_t are optimized through RL. At the end of each task, the dictionary $\mathbf{D}_t$ is optimized (Mairal et al., 2009; Arora et al., 2015) for all learned tasks to provide a mapping from their task embedding to their optimized prompts/sub-networks, which exploits the task correlations in both the embedding and prompt spaces. This leads to efficient usage of the meta-policy network's capacity and automatic optimization of the plasticity-stability trade-off, i.e., relevant tasks reuse skills by sharing more neurons (plasticity and fast adaptation) while the harmful interference between irrelevant tasks is avoided by sharing less or no neurons (stability and less forgetting). Moreover, due to the dictionary, CoTASP does not need to store or replay any previous tasks' experiences and thus costs much less computation and memory than rehearsal-based methods (Rolnick et al., 2019; Wolczyk et al., 2022). Furthermore, the sparse prompting in CoTASP, as an efficient task adaptation method, can extract policy sub-networks for unseen tasks and thus leads to a more generalizable meta-policy.

On Continual World benchmarks (Wolczyk et al., 2021), CoTASP outperforms most baselines on all learned tasks, forgetting reduction, and generalization to unseen tasks (Table 1). A thorough ablation study (Table 2) demonstrates the importance of dictionary learning and sparse prompt optimization. Moreover, our empirical analysis shows that the dictionary converges fast (Fig. 5(b)) and can be generalized to future tasks, significantly reducing their adaptation cost (Fig. 5(a)), while the learned sparse prompts capture the semantic correlations between tasks (Fig. 7). In comparison with state-of-the-art methods, CoTASP achieves the best plasticity-stability trade-off (Fig. 2) and highly efficient usage of model capacity (Fig. 4).

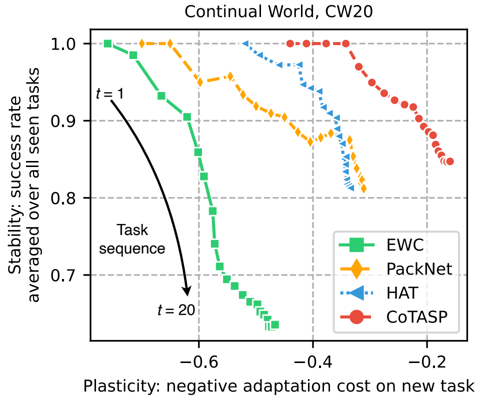


Figure 2: **Plasticity-stability trade-off** in continual RL. Adaptation cost is the number of environment steps normalized to $(0, 1]$.

2. Preliminaries and Related Work

We follow the task-incremental setting adopted by prior work (Khetarpal et al., 2022; Wolczyk et al., 2022; 2021; Rolnick et al., 2019; Mendez et al., 2020; Schwarz et al., 2018; Rusu et al., 2016), which considers a sequence of tasks, each defining a Markov Decision Process (MDP) $\mathcal{M}_t = \langle \mathcal{S}_t, \mathcal{A}_t, p_t, r_t, \gamma \rangle$ such that $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition probability where $\Delta(\mathcal{S})$ is the probability simplex over $\mathcal{S}$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function so $r_t(s_{t,h}, a_{t,h})$ is the immediate reward in task t when taking action $a_{t,h}$ at state $s_{t,h}$, h indexes the environment step, and $\gamma \in [0, 1)$ is the discounted factor. Continual RL aims to achieve a policy π_θ at task $\mathcal{T}$ that performs well (with high expected return) on all seen tasks $t \leq \mathcal{T}$, with only a limited (or without) buffer of previous tasks' experiences:

$$\theta^* = \arg\max_{\theta} \sum_{t=1}^{\mathcal{T}} \mathbb{E}_{\pi_\theta} \left[\sum_{h=0}^{\infty} \gamma^h r_t(s_{t,h}, a_{t,h}) \right] \quad (1)$$

Continual learning is a natural human skill that can accumulate knowledge generalizable to new tasks without forgetting the learned ones. However, RL agents often struggle with achieving the goal in Eq. 1 due to the plasticity-stability trade-off: the policy is expected to quickly adapt to new tasks $t \geq \mathcal{T}$ (plasticity) but meanwhile to retain its performance on previous tasks $t < \mathcal{T}$ (stability).

Existing strategies for continual RL mainly focus on improving stability and reducing catastrophic forgetting. Rehearsal-based methods such as CLEAR (Rolnick et al., 2019) and P&C (Schwarz et al., 2018) repeatedly replay buffed experiences from previous tasks but their buffer memory and computational cost linearly grow with the number of tasks (Kumari et al., 2022). Regularization-based methods such as EWC (Kirkpatrick et al., 2017) and PC (Kaplanis et al., 2019) alleviate forgetting without the replay buffer by adding extra regularizers when learning new tasks, which can bias the policy optimization and lead to sub-optimal solutions (Zhao et al., 2023). Finally, structure-based methods adopt different modules, i.e., sub-networks within a fixed-capacity policy network, for each task (Mendez & Eaton, 2022). We summarize two main categories of structure-based methods in the following. We also provide a more detailed discussion of related work in Appendix B and C.

Connection-level methods. This category includes methods such as PackNet (Mallya & Lazebnik, 2018), SupSup (Wortsman et al., 2020), and WSN (Kang et al., 2022). For task t, the action a_t is drawn from $a_t \sim \pi(s_t; \theta \otimes \phi_t)$ where s_t is the state and ϕ_t is a binary mask applied to the model weights θ in an element-wise manner (i.e., $\otimes$). PackNet generates ϕ_t by iteratively pruning θ after the learning of each task, thereby preserving important weights for the task while leaving others for future tasks. SupSup fixes a randomly initialized network and finds the optimal ϕ_t for each task t. WSN jointly learns θ and ϕ_t and uses Huffman coding (Huffman, 1952) to compress ϕ_t for a sub-linear growing size of $\{\phi_t\}_{t=1}^{\mathcal{T}}$ with increasing tasks. However, these methods usually need to store the task-specific masks for each task in history, leading to additional memory costs (Lange et al., 2022). Moreover, their masks are seldom optimized for knowledge sharing across tasks, impeding the learned policy from being generalized to unseen tasks.

Neuron-level methods. Instead of extracting task-specific sub-networks by applying masks to model weights, the other category of methods (Fernando et al., 2017; Serrà et al., 2018; Ke et al., 2021; Sokar et al., 2021) produces sub-networks by applying masks to each layer's neurons/outputs of a policy network. Compared to connection-level methods, they use layer-wise masking to achieve a more flexible and compact representation of sub-networks. But the generation of masks depends on either heuristic rules or computationally inefficient policy gradient methods (Gurbuz & Dovrolis, 2022; Serrà et al., 2018). By contrast, CoTASP generates masks by highly efficient sparse coding (solving a relatively small lasso problem).

3. Methods

In this section, we introduce the main steps and components of CoTASP (see Fig. 1). Specifically, Sec. 3.1 introduces the meta-policy network in the continual RL setting. Sec. 3.2 describes our proposed sparse prompting for task policy extraction. Finally, Sec. 3.3 provides the detailed optimization procedure for each component in CoTASP, including the prompt, task policy, and dictionary.

3.1. Continual RL with a Meta-Policy Network

As discussed in Sec. 2, finetuning all weights in θ via the optimization in Eq. 1 without access to past tasks leads to harmful shift on some weights important to previous tasks and catastrophic forgetting of them. Structure-based methods address it by allocating a sub-network for each task and freezing its weights once completing the learning of the task so they are immune to catastrophic forgetting. Following prior work (Srivastava et al., 2014; Fernando et al., 2017; Serrà et al., 2018; Ke et al., 2021; Sokar et al., 2021), we represent such a sub-network by applying a binary mask to each neuron's output. Specifically, given a meta-policy network with L layers, let $l \in \{1, \ldots, L-1\}$ index the hidden layers of the network as a superscript, e.g., $\boldsymbol{y}^{(l)}$ is the output vector of layer-l and $\theta^{(l)}$ denotes layer-l's weights. The output of the sub-network on its $(l+1)$-th layer is

$$\boldsymbol{y}^{(l+1)} = f(\phi_t^{(l)} \otimes \boldsymbol{y}^{(l)}; \theta^{(l+1)}), \tag{2}$$

where $\phi_t^{(l)}$ is a binary mask generated for task t and applied to layer-l, and f is a stand-in for the neural operation, e.g., a fully-connected or convolutional layer. The element-wise product operator $\otimes$ activates a portion of neurons in layer-l of the meta-policy network according to $\phi_t^{(l)}$. These activated neurons over all layers extract a sub-network as a task-specific policy, which then interacts with the environment to collect training experiences. Training the sub-network avoids harmful interference with previous tasks on other neurons and meanwhile encourages their knowledge sharing on the shared neurons. However, allocating policies in the meta-policy network for a sequence of diverse tasks raises several challenges. For efficient usage of network capacity, each task policy should be a sparse sub-network with only a few neurons activated. Furthermore, the policy should selectively reuse neurons from previously learned policies, which can facilitate knowledge sharing between relevant tasks and reduce interference from irrelevant tasks. Inspired by prompting and in-context training for NLP (Rebuffi et al., 2017; Houlsby et al., 2019; Li & Liang, 2021; Liu et al., 2021), we propose "**Sparse Prompting**" to address the aforementioned challenge of continual task allocation, which can automatically and efficiently extract task-specific policies from the meta-policy network.

3.2. Task Policy Extraction via Sparse Prompting

In continual task allocation, the sub-network extracted for a new task is expected to reuse the knowledge learned from relevant tasks in the past and meanwhile to avoid harmful interference from irrelevant tasks. Moreover, the sub-network should be as sparse as possible for efficient usage of network capacity. Various continual RL methods (Mallya & Lazebnik, 2018; Serrà et al., 2018; Sokar et al., 2021; Kessler et al., 2022; Kang et al., 2022; Wolczyk et al., 2022) use a one-hot embedding to extract the sub-network for each learned task. They overlook semantic correlations among tasks and need delicate mechanisms to keep the sub-network sparse (Lange et al., 2022).

In CoTASP, we instead extract sparse sub-networks from a compact embedding of a task's textual description produced by Sentence-BERT (S-BERT) (Reimers & Gurevych, 2019) via sparse coding (Mairal et al., 2009; Arora et al., 2015). In particular, we learn an over-complete dictionary $\mathbf{D}^{(l)} \in \mathbb{R}^{m\times k}$ ($m \ll k$) for each layer-l of the meta-policy network, in which each column is an atom representing a neuron in the layer. Given a task embedding $e_t \in \mathbb{R}^m$, sparse prompting can produce a sparse prompt $\alpha_t^{(l)}$ for each layer-l that reconstructs e_t as a linear combination of a few neurons' representations, i.e., atoms from the dictionary. It is equal to solving the following lasso problem.

$$\alpha_t^{(l)} = \arg\min_{\alpha\in\mathbb{R}^k} \frac{1}{2}\|e_t - \mathbf{D}_{t-1}^{(l)}\alpha\|_2^2 + \lambda\|\alpha\|_1, \quad \text{for layer } l = 1, \ldots, L-1 \tag{3}$$

where λ is a regularization parameter controlling the sparsity of α. This lasso problem can be solved by a variety of provably efficient approaches, e.g., coordinate descent (Friedman et al., 2007), fast iterative shrinkage thresholding algorithm (Beck & Teboulle, 2009), and LARS algorithm (Efron et al., 2004). In this paper, we adopt a Cholesky-based implementation of the LARS algorithm (Mairal et al., 2009) for its efficiency and stability. To transform $\alpha \in \mathbb{R}^k$ to a binary mask, we apply a step function $\sigma(\cdot)$ on α, in which $\sigma(\alpha) = 1$ if $\alpha > 0$ and 0 otherwise. Then we can extract a task-specific policy sub-network by applying the mask to the meta-policy network as in Eq. 2.

3.3. Meta-Policy and Dictionary Learning in CoTASP

Alternating Optimization of Task Policy and Prompt By alternately optimizing the current task's policy and prompts $\alpha_t^{(l)}$ for $l = 1, \ldots, L-1$ using any off-the-shelf RL algorithm, CoTASP updates the sub-network weights associated with the task policy in the meta-policy network and the corresponding binary masks. However, there are two practical concerns: (1) updating the weights in θ that have already been selected by previous tasks may degrade the old tasks' performance without experience replay; and (2) the step function $\sigma(\cdot)$ has a zero gradient so optimizing α_t using such a gradient is infeasible.

To address the first concern, we update the weights selectively by only allowing updates of weights that have never been allocated for any previous task. For this purpose, we accumulate the binary masks for all learned tasks by $\hat{\phi}_{t-1}^{(l)} = \vee_{i=1}^{t-1}\phi_{t-1}^{(l)}$ and update θ when learning task t by

$$\theta \leftarrow \theta - \eta\hat{\mathbf{g}}_t$$

$$\hat{g}_t^{(l)} = \begin{cases} \left(1 - \hat{\phi}_{t-1}^{(l)}\right) g_t^{(l)}, & l = 1 \\ \left(1 - \hat{\phi}_{t-1}^{(l-1)}\right) g_t^{(l)}, & l = L \\ \left[1 - \min(\hat{\phi}_{t-1}^{(l-1)}, \hat{\phi}_{t-1}^{(l)})\right] g_t^{(l)}, & l > 1 \end{cases} \tag{4}$$

where η is the learning rate and $g_t^{(l)}$ denotes negative gradients of the expected return w.r.t. θ for layer-l on the task t. In Eq. 4, we modify each weight's gradient according to the accumulated mask associated with its input and output layer. This effectively avoids overwriting the weights selected by policies of previous tasks and thus mitigates forgetting.

Algorithm 1 Dictionary Learning

```
1: input: $\mathbf{D}^{(l)}$ for hidden layer-$l$, $\mathbf{A}^{(l)} = [a_1^{(l)}, \ldots, a_k^{(l)}] \in \mathbb{R}^{k\times k} = \sum_{i=1}^t \alpha_i^{*(l)}\alpha_i^{*(l)\mathrm{T}}$, $\mathbf{B}^{(l)} = [b_1^{(l)}, \ldots, b_k^{(l)}] \in \mathbb{R}^{m\times k} = \sum_{i=1}^t e_i\alpha_i^{*(l)\mathrm{T}}$, and constant $c$
2: while until convergence do
3:     for $j = 1$ to $k$ do
4:         $z = {}^1\!/\mathbf{A}_{jj}^{(l)}(b_j^{(l)} - \mathbf{D}^{(l)}a_j^{(l)}) + \mathbf{D}^{(l)}[j]$
5:         $\mathbf{D}^{(l)}[j] = \min\{\frac{c}{\|z\|_2}, 1\}z$ ▷ $\ell_2$-norm constraint
6: output: updated $\mathbf{D}^{(l)}$;
```

To address the second concern, we use the straight-through estimator (STE) (Bengio et al., 2013), i.e., clip$(\alpha, 0, 1)$, in the backward pass so that the α can be directly optimized using the same gradient descent algorithm applied to the meta-policy weights.

Dictionary Learning Given the previous tasks' optimized prompts α_i^* and their embedding, we further update the dictionary per layer so that sparse prompting will be improved to produce the optimized prompts for each previous task., i.e.,

$$\mathbf{D}_t^{(l)} = \arg\min_{\mathbf{D}\in\mathbb{R}^{m\times k}} \frac{1}{2}\sum_{i=1}^t \|e_i - \mathbf{D}\alpha_i^{*(l)}\|_2^2, \text{ s.t., } \|\mathbf{D}[j]\|_2 \le c \quad \forall j = 1, \ldots, k, \text{ for layer } l = 1, \ldots, L-1 \tag{5}$$

where we constrain the ℓ_2 norm of each atom $\mathbf{D}[j]$ to prevent the scale of $\mathbf{D}$ from growing arbitrarily large, which would lead to arbitrarily small entries in α.

To solve the optimization with inequality constraints in Eq. 5, we use block-coordinate descent with $\mathbf{D}_{t-1}^{(l)}$ as warm restart, as described in Alg. 1. Specifically, we sequentially update each atom of $\mathbf{D}_{t-1}^{(l)}$ under the constraint $\|\mathbf{D}[j]\|_2 \le c$

Algorithm 2 Training Procedure of CoTASP

```
initialize: replay buffer $\mathcal{B} = \varnothing$, meta-policy network $\pi_\theta$ with $L$ layers, critic $Q$, dictionaries $\{\mathbf{D}_0^{(l)}\}_{l=1}^{L-1}$, $\mathbf{A}_0^{(l)}, \mathbf{B}_0^{(l)}, \hat{\phi}_0^{(l)} \leftarrow \mathbf{0}, \mathbf{0}, \mathbf{0}$, and constant $c$ for Alg. 1
input: training budget $I_\theta$, $I_\alpha$, and step function $\sigma(\cdot)$
for $t = 1$ to $\mathcal{T}$ do
    $e_t = f_{\text{S-BERT}}$(textual description of task $t$)
    Initialize $\{\alpha_t^{(l)}\}_{l=1}^{L-1}$ by solving Eq. 3
    Extract task-specific $\tilde{\pi}$ by Eq. 2 with $\{\sigma(\alpha_t^{(l)})\}_{l=1}^{L-1}$
    for each iteration do    ▷ Learning task $t$ with SAC
        for $i = 1$ to $I_\theta$ do    ▷ Optimizing $\theta$
            Collect $\tau = \{s_t, a_t, r_t, s_t'\}$ with $\tilde{\pi}$
            Update $\mathcal{B}$ and sample a mini-batch $\tau$
            Gradient descent on $Q$
            Update $\theta$ by Eq. 4 with $\{\hat{\phi}_{t-1}^{(l)}\}_{l=1}^{L-1}$
        for $i = 1$ to $I_\alpha$ do    ▷ Optimizing $\alpha$
            Collect $\tau = \{s_t, a_t, r_t, s_t'\}$ with $\tilde{\pi}$
            Update $\mathcal{B}$ and sample a mini-batch $\tau$
            Gradient descent on $Q$
            Gradient descent on $\{\alpha_t^{(l)}\}_{l=1}^{L-1}$ by STE
    for $l = 1$ to $L - 1$ do    ▷ Dictionary learning
        $\hat{\phi}_t^{(l)} \leftarrow \hat{\phi}_{t-1}^{(l)} \vee \sigma(\alpha_t^{*(l)})$
        $\mathbf{A}_t^{(l)} \leftarrow \mathbf{A}_{t-1}^{(l)} + \alpha_t^{*(l)} \alpha_t^{*(l)\mathrm{T}}$
        $\mathbf{B}_t^{(l)} \leftarrow \mathbf{B}_{t-1}^{(l)} + e_t \alpha_t^{*(l)\mathrm{T}}$
        Get updated $\mathbf{D}_t^{(l)}$ by Alg. 1 with $\mathbf{D}_{t-1}^{(l)}$
output: $\theta^*$ and $\{\mathbf{D}^{*(l)}\}_{l=1}^{L-1}$;
```

while fixing the rest of atoms. Since this optimization admits separable constraints to the updated atoms, convergence to a global optimum is guaranteed (Bertsekas, 1997; Lee et al., 2006; Mairal et al., 2009). Moreover, $\mathbf{D}_{t-1}^{(l)}$ as a warm start for $\mathbf{D}_t^{(l)}$ significantly reduces the required optimization steps: we empirically found one step suffices. The complete training procedure of CoTASP is detailed in Alg. 2.

4. Experiments

4.1. Experimental Setups

Benchmarks. To evaluate CoTASP, we follow the same settings as prior work (Wolczyk et al., 2022) and perform thorough experiments. Specifically, we primarily use CW10, a benchmark in the Continual World (CW) (Wolczyk et al., 2021), which consists of 10 representative manipulation tasks from MetaWorld (Yu et al., 2019). To make the benchmark more challenging, we rank these tasks according to a pre-computed transfer matrix so that there is a high variation of forward transfer both in the whole sequence and locally. We also use CW20, which repeats CW10 twice, to measure the transferability of the learned policy when encountering the same task. For a fair comparison between different tasks, the number of environment interaction steps is limited to 1M per task.

Evaluation metrics. Following a widely-used evaluation protocol in continual learning literature (Lopez-Paz & Ranzato, 2017; Rolnick et al., 2019; Chaudhry et al., 2019; Wolczyk et al., 2021; 2022), we adopt three metrics. (1) *Average Performance* (higher is better): the average performance at time t is defined as $P(t) = \frac{1}{\mathcal{T}} \sum_{i=1}^{\mathcal{T}} p_i(t)$ where $p_i(t) \in [0, 1]$ denotes the success rate of task i at time t. This is a canonical metric used in the continual learning community. (2) *Forgetting* (lower is better): it measures the average degradation across all tasks at the end of learning, denoted by $F = \frac{1}{\mathcal{T}} \sum_{i=1}^{\mathcal{T}} p_i(i \cdot \delta) - p_i(\mathcal{T} \cdot \delta)$, where δ is the allowed environment steps for each task. (3) *Generalization* (lower is better): it equals to the average number of steps needed to reach a success threshold across all tasks. Note that we stop the training when the success rate in two consecutive evaluations reaches the threshold (set to 0.9). Moreover, the metric is divided by δ to normalize its scale to $[0, 1]$.

Comparing methods. We compare CoTASP with several baselines and state-of-the-art (SoTA) continual RL methods. According to (Lange et al., 2022), these methods can be divided into three categories: regularization-based, structure-based, and rehearsal-based methods. Concretely, regularization-based methods include L2, Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), Memory-Aware Synapses (MAS) (Aljundi et al., 2018), and Variational Continual Learning (VCL) (Nguyen et al., 2018). Structure-based methods include PackNet (Mallya & Lazebnik, 2018), Hard Attention to Tasks (HAT) (Serrà et al., 2018), and TaDeLL (Rostami et al., 2020). Rehearsal-based methods include Reservoir, Average Gradient Episodic Memory (A-GEM) (Chaudhry et al., 2019), and ClonEx-SAC (Wolczyk et al., 2022). For completeness, we also include a naive sequential training method (i.e., Finetuning) and representative multi-task RL baselines (MTL (Yu et al., 2019) and MTL+PopArt (Hessel et al., 2019)), which are usually regarded as the soft upper bound a continual RL method can achieve. For a fair comparison, we refer to the Continual World repository[1] for implementation and hyper-parameter selection. We re-run these methods to ensure the best possible performance. In addition, we adopt author-reported results for ClonEx-SAC due to the lack of open-sourced implementation. An extended description and discussion of these methods are provided in Appendix C.

Training details. In order to ensure the reliability and comparability of our experiments, we follow the training details described in (Wolczyk et al., 2021; 2022) and implement all of the baseline methods based on Soft Actor-Critic (SAC) (Haarnoja et al., 2018), a SoTA off-policy actor-critic

[1] https://github.com/awarelab/continual_world

Benchmarks		CW 10			CW 20		
Metrics		P (↑)	F (↓)	G (↓)	P (↑)	F (↓)	G (↓)
Reg	L2	0.44 ±0.12	0.00 ±0.06	0.51 ±0.07	0.52 ±0.07	-0.10 ±0.05	0.58 ±0.06
	EWC	0.64 ±0.14	0.02 ±0.05	0.34 ±0.04	0.60 ±0.07	0.02 ±0.03	0.39 ±0.06
	MAS	0.60 ±0.14	-0.06 ±0.04	0.44 ±0.07	0.48 ±0.06	0.02 ±0.02	0.49 ±0.03
	VCL	0.48 ±0.10	-0.02 ±0.06	0.43 ±0.06	0.50 ±0.11	-0.04 ±0.08	0.52 ±0.06
	Finetuning	0.12 ±0.04	0.70 ±0.04	0.25 ±0.06	0.05 ±0.00	0.72 ±0.03	0.30 ±0.05
Struc	PackNet	0.80 ±0.09	0.00 ±0.00	0.28 ±0.07	0.78 ±0.07	0.00 ±0.00	0.32 ±0.04
	HAT	0.68 ±0.12	0.00 ±0.00	0.44 ±0.07	0.67 ±0.08	0.00 ±0.00	0.46 ±0.04
	TaDeLL	0.75 ±0.04	0.00 ±0.00	0.68 ±0.01	0.66 ±0.03	0.01 ±0.02	0.67 ±0.01
Reh	Reservoir	0.32 ±0.12	0.04 ±0.05	0.79 ±0.02	0.08 ±0.09	0.14 ±0.05	0.87 ±0.01
	A-GEM	0.14 ±0.05	0.68 ±0.04	0.23 ±0.02	0.08 ±0.02	0.72 ±0.07	0.29 ±0.04
	ClonEx-SAC*	0.86	0.02	–	0.87	0.02	–
MT	MTL	0.52 ±0.10	–	–	0.50 ±0.11	–	–
	MTL+PopArt	0.70 ±0.14	–	–	0.66 ±0.17	–	–
	CoTASP (ours)	0.92 ±0.04	0.00 ±0.00	0.24 ±0.03	0.88 ±0.02	0.00 ±0.00	0.27 ±0.03

Table 1: **Evaluation (mean±std of 3 metrics over 5 random seeds) on Continual World.** ∗-reported in previous work. Reg = Regularization-based, Struc = Structure-based, Reh = Rehearsal-based, MT = Multi-task, P = Average Performance, F = Forgetting, G = Generalization. A detailed description of baselines and metrics can be found in Sec. 4.1. The best result for each metric is highlighted.

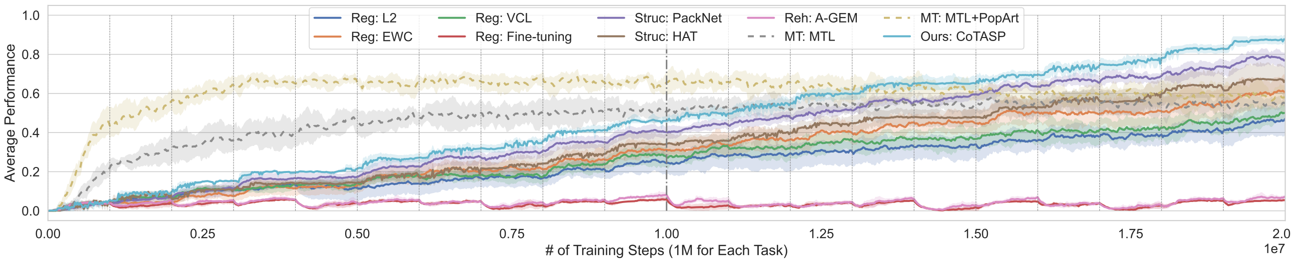


Figure 3: **Performance (mean±std over 5 random seeds) of all methods on CW20 sequence.** CoTASP outperforms all the continual RL methods and all the multi-task RL baselines. We separately plot the curve per method in Fig. 9 of Appendix.

algorithm. The actor and the critic are implemented as two separate multi-layer perceptron (MLP) networks, each with 4 hidden layers of 256 neurons. For structure-based methods (PackNet, HAT) and our proposed CoTASP, a wider MLP network with 1024 neurons per layer is used as the actor. We refer to these hidden layers as the backbone and the last output layer as the head. Unlike other continual RL methods (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Serrà et al., 2018; Kessler et al., 2022) which rely on using a separate head for each new task, CoTASP uses a single-head setting where only one head is used for all tasks. In this case, CoTASP does not require selecting the appropriate head for each task and enables the reuse of parameters between similar tasks. According to (Wolczyk et al., 2021), regularizing the critic often leads to a decline in performance. Therefore, we completely ignore the forgetting issue in the critic network and retrain it for each new task. More details on the hyperparameters used in training can be found in the Appendix D.

4.2. Main Results

This section presents the comparison between CoTASP and ten representative continual RL methods on CW benchmarks. We focus on the *stability* (retain performance on seen tasks) and the *plasticity* (quickly adapt to unseen tasks) and keep the constraints on computation, memory, number of samples, and neural network architecture constant. Table 1 summarizes our main results on CW10 and CW20 sequences. CoTASP consistently outperforms all the compared methods across different lengths of task sequences, in terms of both average performance (measures *stability*) and generalization (measures *plasticity*). We observe that when the hidden-layer size is the same as other structure-based methods (PackNet and HAT), CoTASP outperforms them by a large margin, especially in the generalization metric, indicating the advantage of CoTASP in improving the adaptation to new tasks. A more detailed analysis of the reasons for CoTASP's effectiveness is presented in Sec. 4.3 and 4.4. Moreover, we find that most continual

Benchmark	CW 20	
Metrics	P (↑)	G (↓)
CoTASP (ours)	0.88 ±0.02	0.27 ±0.03
with $\mathbf{D}$ frozen	0.73 ±0.06	0.47 ±0.03
with $\boldsymbol{\alpha}$ frozen	0.79 ±0.06	0.34 ±0.02
with both frozen	0.62 ±0.05	0.52 ±0.03
lazily update $\mathbf{D}$	**0.85** ±0.03	**0.29** ±0.05
EWC	0.60 ±0.07	0.39 ±0.06
PackNet	**0.78** ±0.07	0.32 ±0.04
A-GEM	0.08 ±0.02	**0.29** ±0.04
Finetuning	0.05 ±0.00	0.30 ±0.05

Table 2: **Ablation study.** Performance of CoTASP variants on CW20 sequence. Please refer to Sec. 4.3 for a detailed explanation.

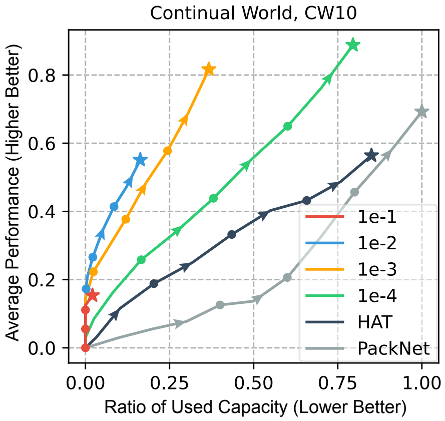


Figure 4: **Model Capacity Usage.** Comparison of CoTASP with different sparsity λ and two baselines on the efficiency of model capacity usage, i.e., ratio of used parameters vs. performance.

RL methods fail to achieve positive backward transfer (i.e., $F < 0$) except for VCL, suggesting the ability to improve previous tasks' performance by learning new ones is still a significant challenge. We leave this for future work. Finally, the results in Fig. 3 show that CoTASP is the only method performing comparably to the multi-task learning baselines on the first ten tasks of CW20 sequence, and it exhibits superior performance over these baselines after learning the entire CW20 sequence. One possible explanation is that the knowledge accumulated by CoTASP's meta-policy network and dictionaries leads to improved generalization.

4.3. Ablation Studies

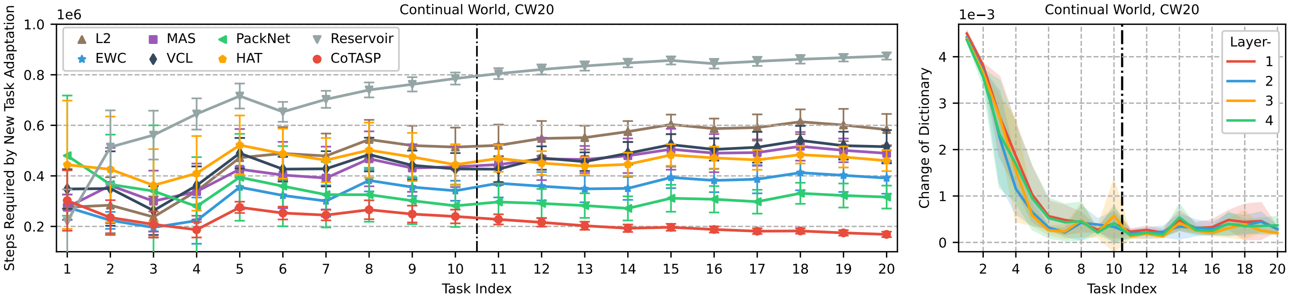
Effectiveness of core designs. To show the effectiveness of each of our components, we conduct an ablation study on four variants of CoTASP, each of which removes or changes a single design choice made in the original CoTASP. Table 2 presents the results of the ablation study on CW20 sequence, using two representative evaluation metrics. Among the four variants of CoTASP, "$\mathbf{D}$ frozen" replaces the learnable dictionary with a fixed, randomly initialized one; "$\boldsymbol{\alpha}$ frozen" removes the prompt optimization proposed in Sec. 3.3; "both frozen" neither updates the dictionary nor optimizes the prompt; "lazily update $\mathbf{D}$" stops the dictionary learning after completing the first ten tasks of CW20 sequence. According to the results in Table 2, we give the following conclusions: (1) The use of a fixed, randomly initialized dictionary degrades the performance of CoTASP on two evaluation metrics, highlighting the importance of the learnable dictionary in capturing semantic correlations among tasks. (2) The "$\boldsymbol{\alpha}$ frozen" variant performs comparably to our CoTASP but outperforms the results achieved by EWC and PackNet. This indicates that optimizing the prompt can improve CoTASP's performance but is not crucial to our appealing results. (3) The "both frozen" variant exhibits noticeable degradation in performance, supporting the conclusion that the combination of core designs proposed in CoTASP is essential for achieving strong results. (4) The "lazily update $\mathbf{D}$" variant only slightly degrades from the original CoTASP on the performance but still outperforms all baselines by a large margin, indicating that the learned dictionary has accumulated sufficient knowledge in the first ten tasks so that CoTASP can achieve competitive results without updating the dictionary for repetitive tasks.

Effect of key hyperparameters. CoTASP introduces the sparsity parameter λ, a hyperparameter that controls the trade-off between the used network capacity and the performance of the resulting policy. A larger value of λ results in a more sparse policy sub-network, improving the usage efficiency of the meta-policy network's capacity. But the cost is decreased performance on each task due to the loss of expressivity of the over-sparse task policy. According to the results in Fig. 4, CoTASP with λ=1e-3 or 1e-4 achieves better trade-off between performance and usage efficiency than other structure-based methods (HAT and PackNet) on CW10 sequence.

4.4. Why does CoTASP work? An Empirical Study

In this section, we answer the following questions based on the phenomena observed from our empirical results: (1) Is the learned dictionary generalizable? (2) Does the sparse prompts generated by CoTASP capture the semantic correlations between tasks?

To answer the first question, we measure the change of dictionaries, visualize their dynamics, and compare the adaptation cost of CoTASP with other continual/meta RL methods on CW20 sequence. The results in Fig 5(b) show that these dictionaries converge quickly with the increasing number of tasks, leading to a stationary mapping from task embedding to the optimized prompt. In next new task, CoTASP



(a) Results of Task Adaptation

(b) Results of Dictionary Learning

Figure 5: **(a):** Steps (mean±std over 5 random seeds) required by new task adaptation after learning t tasks (fewer is better) on CW20 sequence. **CoTASP spends the least steps on the new task adaptation and the steps decrease for later tasks**, indicating a benefit of knowledge reuse. **(b):** The change of dictionary (averaged over 5 random seeds) of each hidden layer on CW20 sequence. The change is computed by $1/|\mathbf{D}_t^{(l)}|\|\mathbf{D}_t^{(l)} - \mathbf{D}_{t-1}^{(l)}\|_2^2$. The black dash-dotted line splits the x-axis in two parts, the first CW10 sequence (left part) and the repeated one (right part). It shows **a fast convergence of the dictionary learning**. Please refer to Sec. 4.4 for a detailed discussion.

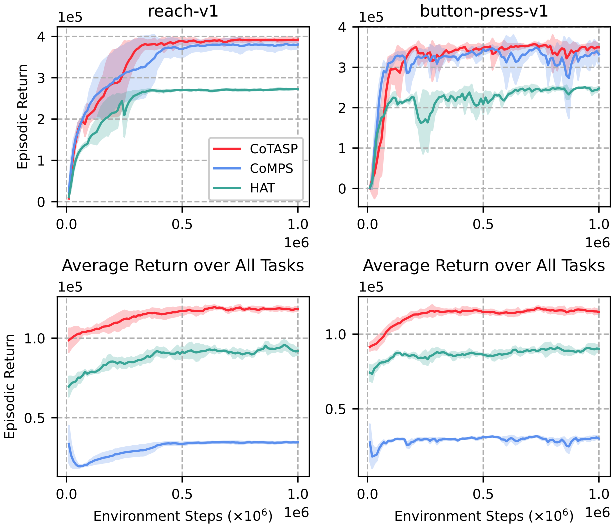
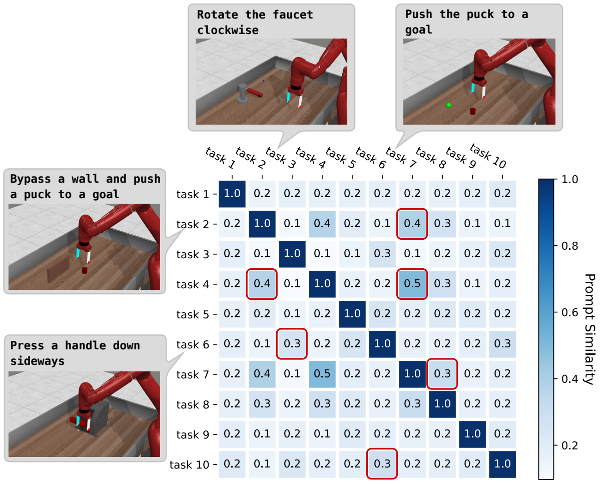


Figure 6: **Evaluation (mean±std over 6 random seeds) in Meta-World environments** (Yu et al., 2019). The complete results in 5 environments are reported in Fig. 11 of Appendix.

will produce a "good" initial prompts by sparse coding of its task embedding. This significantly reduces the number of training steps needed to reach the success threshold, as demonstrated by Fig 5(a). Furthermore, we compare CoTASP with a SoTA meta-RL algorithm, CoMPS (Berseth et al., 2022), to show its superiority. Specifically, we pretrain CoTASP and CoMPS on CW10 sequence and then use the learned meta-policy as the initial policy to fine-tune unseen tasks, e.g., reach-v1 and button-press-v1. According to the results shown in Fig 6, CoTASP performs comparably to CoMPS but significantly better than the continual RL baseline (HAT) in terms of adaptation cost. However, due to the lack of mechanism against catastrophic forgetting, CoMPS adapts to the new task while its performance on previous tasks quickly degrades, resulting in worse average return across all tasks.



	task 1	task 2	task 3	task 4	task 5	task 6	task 7	task 8	task 9	task 10
task 1	1.0	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
task 2	0.2	1.0	0.1	0.4	0.2	0.1	0.4	0.3	0.1	0.1
task 3	0.2	0.1	1.0	0.1	0.1	0.3	0.1	0.2	0.2	0.2
task 4	0.2	0.4	0.1	1.0	0.2	0.2	0.5	0.3	0.1	0.2
task 5	0.2	0.2	0.1	0.2	1.0	0.2	0.2	0.2	0.2	0.2
task 6	0.2	0.1	0.3	0.2	0.2	1.0	0.2	0.2	0.2	0.3
task 7	0.2	0.4	0.1	0.5	0.2	0.2	1.0	0.3	0.2	0.2
task 8	0.2	0.3	0.2	0.3	0.2	0.2	0.3	1.0	0.2	0.2
task 9	0.2	0.1	0.2	0.1	0.2	0.2	0.2	0.2	1.0	0.2
task 10	0.2	0.1	0.2	0.2	0.2	0.3	0.2	0.2	0.2	1.0

Figure 7: **Prompt (sub-network mask) similarity between two tasks** i and j computed by $\frac{1}{L-1}\sum_l \|\phi_i^{(l)} \wedge \phi_j^{(l)}\|_1 / \|\phi_i^{(l)} \vee \phi_j^{(l)}\|_1$. A layer-wise version is provided in Fig. 10 of Appendix.

To answer the second question, we visualize the similarity (i.e., the overlap between two binary prompts) of the prompts generated by CoTASP between every two tasks over CW10 sequence in Fig. 7. The blue heatmap summarizes the similarity values averaged over all hidden layers. Specifically, the element on row i and column j is the averaged similarity value computed between task i and task j. For task 2 and task 7, their task descriptions share the same manipulation primitive, i.e., *pushing a puck*. Hence, the prompts generated by solving the lasso problem in Eq. 3 are highly correlated. By contrast, for task 2 and task 7 with irrelevant task descriptions, CoTASP produces different prompts, reducing cross-task interference and improving plasticity.

5. Conclusions

We propose CoTASP to address two key challenges in continual RL: (1) training a meta-policy generalizable to all seen and even unseen tasks, and (2) efficiently extracting a task policy from the meta-policy. CoTASP learns a dictionary to produce sparse masks (prompts) to extract each task's policy as a sub-network of the meta-policy and optimizes the sub-network via RL. This encourages knowledge sharing/reusing among relevant tasks while reducing harmful cross-task interference that causes forgetting and poor new-task adaptation. Without any experience replay, CoTASP achieves a significantly better plasticity-stability trade-off and more efficient network capacity allocation than baselines. Its extracted policies outperform all baselines on both previous and new tasks.

Acknowledgements

Yijun Yang and Yuhui Shi are partially supported by the Science and Technology Innovation Committee Foundation of Shenzhen under the Grant No. JCYJ20200109141235597 and ZDSYS201703031748284, National Science Foundation of China under grant number 61761136008, Shenzhen Peacock Plan under Grant No. KQTD2016112514355531, and Program for Guangdong Introducing Innovative and Entrepreneurial Teams under grant number 2017ZT07X386. We would like to thank ICML area chairs and anonymous reviewers for their efforts in reviewing this paper and their constructive comments!

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

Ao, S., Zhou, T., Long, G., Lu, Q., Zhu, L., and Jiang, J. CO-PILOT: collaborative planning and reinforcement learning on sub-task curriculum. In *NeurIPS*, 2021.

Ao, S., Zhou, T., Jiang, J., Long, G., Song, X., and Zhang, C. EAT-C: environment-adversarial sub-task curriculum for efficient reinforcement learning. In *ICML*, 2022.

Arora, S., Ge, R., Ma, T., and Moitra, A. Simple, efficient, and neural algorithms for sparse coding. In *COLT*, 2015.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Bengio, E., Pineau, J., and Precup, D. Interference and generalization in temporal difference learning. In *ICML*, 2020.

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, 2013.

Berseth, G., Zhang, Z., Zhang, G., Finn, C., and Levine, S. Comps: Continual meta policy search. In *ICLR*, 2022.

Bertsekas, D. P. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

Beyer, H. and Schwefel, H. Evolution strategies - A comprehensive introduction. *Nat. Comput.*, 1(1):3–52, 2002.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with A-GEM. In *ICLR*, 2019.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B. D., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., Donner, C., Fritz, L., Galperti, C., Huber, A., Keeling, J., Tsimpoukelli, M., Kay, J., Merle, A., Moret, J., Noury, S., Pesamosca, F., Pfau, D., Sauter, O., Sommariva, C., Coda, S., Duval, B., Fasoli, A., Kohli, P., Kavukcuoglu, K., Hassabis, D., and Riedmiller, M. A. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nat.*, 602(7897):414–419, 2022.

Duan, Q., Zhou, G., Shao, C., Yang, Y., and Shi, Y. Collective learning of low-memory matrix adaptation for large-scale black-box optimization. In *PPSN*, 2022.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

Fang, M., Zhou, T., Du, Y., Han, L., and Zhang, Z. Curriculum-guided hindsight experience replay. In *NeurIPS*, 2019.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, 2017.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. Pathwise coordinate optimization. *The annals of applied statistics*, 1(2):302–332, 2007.

Gurbuz, M. B. and Dovrolis, C. NISPA: neuro-inspired stability-plasticity adaptation for continual learning in sparse networks. In *ICML*, 2022.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H. Multi-task deep reinforcement learning with popart. In *AAAI*, 2019.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In *ICML*, 2019.

Huffman, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

Kang, H., Mina, R. J. L., Madjid, S. R. H., Yoon, J., Hasegawa-Johnson, M., Hwang, S. J., and Yoo, C. D. Forget-free continual learning with winning subnetworks. In *ICML*, 2022.

Kaplanis, C., Shanahan, M., and Clopath, C. Policy consolidation for continual reinforcement learning. In *ICML*, 2019.

Ke, Z., Liu, B., Ma, N., Xu, H., and Shu, L. Achieving forgetting prevention and knowledge transfer in continual learning. In *NeurIPS*, 2021.

Kessler, S., Parker-Holder, J., Ball, P. J., Zohren, S., and Roberts, S. J. Same state, different task: Continual reinforcement learning without interference. In *AAAI*, 2022.

Khetarpal, K., Riemer, M., Rish, I., and Precup, D. Towards continual reinforcement learning: A review and perspectives. *J. Artif. Intell. Res.*, 75:1401–1476, 2022.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526, 2017.

Kostrikov, I. JAXRL: Implementations of Reinforcement Learning algorithms in JAX, 10 2021. URL https://github.com/ikostrikov/jaxrl.

Kumari, L., Wang, S., Zhou, T., and Bilmes, J. A. Retrospective adversarial replay for continual learning. In *NeurIPS*, 2022.

Lange, M. D., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G. G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44 (7):3366–3385, 2022.

Lee, H., Battle, A. J., Raina, R., and Ng, A. Y. Efficient sparse coding algorithms. In *NIPS*, 2006.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947, 2018.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control. *CoRR*, 2022.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, 2021.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

Mairal, J., Bach, F. R., Ponce, J., and Sapiro, G. Online dictionary learning for sparse coding. In *ICML*, 2009.

Mallya, A. and Lazebnik, S. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

Mallya, A., Davis, D., and Lazebnik, S. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, 2018.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. 1989.

Mendez, J. A. and Eaton, E. How to reuse and compose knowledge for a lifetime of tasks: A survey on continual learning and functional composition. *CoRR*, 2022.

Mendez, J. A., Wang, B., and Eaton, E. Lifelong policy gradient learning of factored policies for faster training without forgetting. In *NeurIPS*, 2020.

Mirzadeh, S., Farajtabar, M., and Ghasemzadeh, H. Dropout as an implicit gating mechanism for continual learning. In *CVPR*, 2020.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. In *ICLR*, 2018.

Rajasegaran, J., Hayat, M., Khan, S. H., Khan, F. S., and Shao, L. Random path selection for continual learning. In *NeurIPS*, 2019.

Rebuffi, S., Bilen, H., and Vedaldi, A. Learning multiple visual domains with residual adapters. In *NeurIPS*, 2017.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., and Wayne, G. Experience replay for continual learning. In *NeurIPS*, 2019.

Rostami, M., Isele, D., and Eaton, E. Using task descriptions in lifelong machine learning for improved performance and zero-shot transfer. *J. Artif. Intell. Res.*, 67:673–704, 2020.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *CoRR*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, 2017.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & compress: A scalable framework for continual learning. In *ICML*, 2018.

Serrà, J., Suris, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, 2018.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. In *NeurIPS*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016.

Sokar, G., Mocanu, D. C., and Pechenizkiy, M. Spacenet: Make free space for continual learning. *Neurocomputing*, 439:1–11, 2021.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

Wen, Y., Tran, D., and Ba, J. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *ICLR*, 2020.

Wolczyk, M., Zajac, M., Pascanu, R., Kucinski, L., and Milos, P. Continual world: A robotic benchmark for continual reinforcement learning. In *NeurIPS*, 2021.

Wolczyk, M., Zajac, M., Pascanu, R., Kucinski, L., and Milos, P. Disentangling transfer in continual reinforcement learning. *CoRR*, 2022.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. Supermasks in superposition. In *NeurIPS*, 2020.

Yang, Y., Jiang, J., Zhou, T., Ma, J., and Shi, Y. Pareto policy pool for model-based offline reinforcement learning. In *ICLR*, 2022.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019.

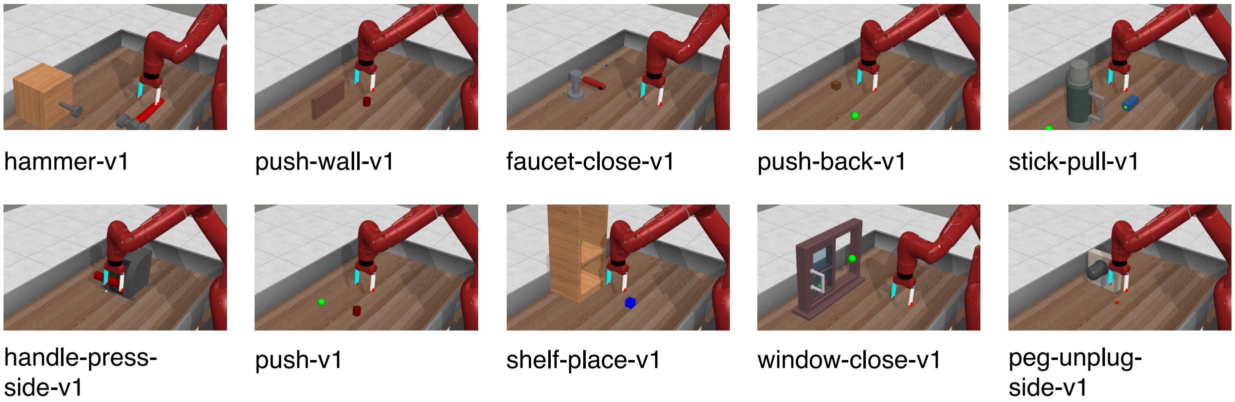
Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. MOPO: model-based offline policy optimization. In *NeurIPS*, 2020.

Zhao, H., Zhou, T., Long, G., Jiang, J., and Zhang, C. Does continual learning equally forget all parameters? In *ICML*, 2023.

Appendix

A. Continual World benchmark

We visualize all of the tasks in Continual World in Fig. 8, and provide a description of these tasks in Table 3.



hammer-v1

push-wall-v1

faucet-close-v1

push-back-v1

stick-pull-v1

handle-press-side-v1

push-v1

shelf-place-v1

window-close-v1

peg-unplug-side-v1

Figure 8: Continual World benchmark adopts robotic manipulation tasks developed by Meta-World (Yu et al., 2019). Presented above is CW10 sequence.

Index	Task	Description
1	hammer-v1	Hammer a screw on the wall.
2	push-wall-v1	Bypass a wall and push a puck to a goal.
3	faucet-close-v1	Rotate the faucet clockwise.
4	push-back-v1	Pull a puck to a goal.
5	stick-pull-v1	Grasp a stick and pull a box with the stick.
6	handle-press-side-v1	Press a handle down sideways.
7	push-v1	Push the puck to a goal.
8	shelf-place-v1	Pick and place a puck onto a shelf.
9	window-close-v1	Push and close a window.
10	peg-unplug-side-v1	Unplug a peg sideways.

Table 3: A list of all of the Continual-World tasks and a description of each task. Listed above is CW10 sequence. CW20 sequence contains tasks from CW10 repeated twice. Tasks are learned sequentially, and 1M environment interaction steps are allowed per task.

B. Detailed Related Work

A promising strategy to address the stability-plasticity trade-off is using different modules, i.e., sub-networks within a fixed-capacity policy network, for each task. The modules of a new task are flexible to learn (for high plasticity), while the modules of past tasks are fixed (for high stability). In addition, there may exist shared modules that are trained across similar tasks, encouraging knowledge transfer. We introduce existing continual RL methods that adopt the strategy in two groups: connection-level and neuron-level methods, and also discuss their limitations.

Connection-level methods. Continusal RL methods in this category, such as PackNet (Mallya & Lazebnik, 2018), Piggyback (Mallya et al., 2018), SupSup (Wortsman et al., 2020), BatchE (Wen et al., 2020), WSN (Kang et al., 2022) and NISPA (Gurbuz & Dovrolis, 2022), sequentially learn and select an optimal sub-network for each task. Specifically, they alternately optimize the model weights and the binary masks of sub-networks associated with each task while attempting to select a sparse set of weights to be activated by reusing weights of the prior sub-networks. Formally, given the state s_t from task t, the action a_t is computed as $a_t \sim \pi(s_t, \theta \otimes \phi_t)$ where ϕ_t denotes the binary masks for task t, $\otimes$ denotes element-wise product. PackNet generates ϕ via iterative pruning after learning each task, which preserves important weights for current task while leaving others available for the future tasks. Piggyback learns task-specific binary masks on the weights given a pre-trained model. SupSup uses a randomly initialized, fixed network and finds the optimal ϕ for each task. BatchE learns a shared weight matrix on the first task and then produces only a rank-one element-wise ϕ for each new task. WSN jointly learns the weights and task-adaptive ϕ and uses Huffman coding (Huffman, 1952) to compress the resulting masks for a sub-linear increase in memory with respect to the number of tasks. NISPA draws inspiration from the sparse connectivity in the human brain and proposes a heuristic mechanism to generate sparse ϕ for each task.

Neuron-level methods. Instead of extracting sub-networks by applying masks to every model weight as methods in the first category, another group of methods produces sub-networks for each task by applying masks to each layer's output of a policy network. Specifically, given the output h_l of neurons of layer-l, the neurons to be activated are selected by $h'_l = h_l \otimes \phi_l$, in which ϕ_l denotes the binary masks with the same shape as h_l. These selected neurons together constitute a task-specific policy, which then interacts with the environment to collect training experiences. This category includes PathNet (Fernando et al., 2017), HAT (Serrà et al., 2018), CTR (Ke et al., 2021) and SpaceNet (Sokar et al., 2021). PathNet first uses Evolution Strategy (Beyer & Schwefel, 2002; Duan et al., 2022) to produce masks $\{\phi_l\}_{l=1}^{L}$ and then learns a sub-network according to the masks as the optimal policy. On the contrary, HAT jointly learns the policy weights and the binary masks through a gradient-based optimization. CTR borrows the idea from Adapter-BERT (Houlsby et al., 2019), which adds adapters in BERT for parameter-efficient transfer learning. The key distinction between CTR and Adapter-BERT is that it employs task masks to avoid forgetting. SpaceNet obtains a sub-network by compressing the sparse connections between a selected number of neurons in each layer via the proposed sparse training. Although these methods use layer-wise masking mechanisms to achieve a more flexible and compact representation of sub-networks than connection-level methods, they have a limitation in that the generation of the masks requires either heuristic rules or computationally inefficient policy gradient methods. By contrast, our CoTASP, which also falls into this category, generates masks by encoding a accurate and compact task embedding, which is produced by LLMs, as linear combinations of a small number of atoms chosen from an over-complete dictionary. This procedure can be done efficiently with classical optimization tools (Mairal et al., 2009).

Most of the aforementioned methods assume the availability of task labels, in the form of one-hot task embeddings, to identify and utilize the appropriate sub-networks during the training and inference phases. However, this assumption can lead to poor performance in real-world continual RL scenarios where tasks are diverse and prohibitive to obtain labels. CoTASP addresses this issue by learning an online dictionary that automatically generates proper masks based on the textual task descriptions, which are typically much easier to acquire from human commands (Liang et al., 2022). Furthermore, this approach captures the semantic correlation among tasks, leading to more efficient knowledge transfer and improved generalizability.

C. An Extended Description of Compared Methods

We now provide a detailed description of those baseline methods compared with CoTASP. Most of them are developed in the supervised continual learning setting, and require task-specific adaption to the continual RL setting. We refer to a groundbreaking work (Wolczyk et al., 2021) for the implementation of these methods. Concretely,

Regularization-based methods. This line of work reduces forgetting by restricting policy parameters important for the learned tasks. The most basic method is L2. It applies a ℓ_2 regularization to the objective function, which keeps the parameters close to the previously optimized ones. In this method, each parameter is considered to be equally important for the previous tasks. In effect, prior work demonstrates that only a few parameters are essential for retaining performance on previous tasks. Based on this observation, EWC (Kirkpatrick et al., 2017) adopts the Fisher information as a metric to select important parameters for previous tasks. MAS (Aljundi et al., 2018) uses a parameter-wise regularizer and computes the regularization weights for each parameter by estimating their impact on the output of the policy. VCL (Nguyen et al., 2018) interprets the above methods from a Bayesian perspective and adopts variational inference to minimize the KL divergence between the posterior (current distribution of parameters) and the prior (distribution of the previously optimized parameters).

Structure-based methods preserve a set of parameters (i.e., sub-network), which is important for previous tasks, as a task-specific policy from a large "super-network". PackNet (Mallya & Lazebnik, 2018) preserves a single sub-network for each task via iterative pruning after learning the task. HAT (Serrà et al., 2018) jointly learns the policy parameters and the corresponding sub-network masks through a gradient-based optimization. TaDeLL (Rostami et al., 2020) first integrates task descriptors into continual RL and then uses coupled dictionary learning to model the inter-task relationships between the task descriptions and the task-specific policies. However, dictionary learning in TaDeLL is defined on and applied to the policy parameters instead of task embeddings in CoTASP. A linear combination of policy parameters does not hold for nonlinear neural networks (Mendez et al., 2020) and the high dimensionality of the parameters will make the dictionary learning highly inefficient. So TaDeLL is not practical (in terms of both the efficiency and effectiveness) for nonlinear policy networks.

Rehearsal-based methods repeatedly replay buffered experiences from previous tasks and use them for retraining or as constraints to alleviate catastrophic forgetting. We choose a simple reservoir strategy-based baseline, which replays all the data from the past when learning a new task, to investigate whether the naive method can increase the performance of continual RL agents. Moreover, A-GEM (Chaudhry et al., 2019) projects policy gradients from the current experiences to the closest gradients guaranteeing the average performance at previous tasks does not decrease. ClonEX-SAC (Wolczyk et al., 2022) retains some samples from previous tasks and performs behavior cloning based on them to reduce forgetting. It achieved SoTA results on Continual World benchmarks.

D. Hyperparameter Details

We carefully tune the hyperparameters for a JAX implementation of the SAC algorithm (Bradbury et al., 2018; Kostrikov, 2021), and they are common for all baseline methods. Moreover, we tune the hyperparameters used for CoTASP using the final average performance on CW10 sequence as the objective. The search space and selected hyperparameters are presented in Table 4.

Table 4: Hyperparameters of CoTASP for Continual World experiments

Hyperparameter	**Search Space**	**Selected Value**
SAC Hyperparameters		
Actor hidden size	$\{256, 512, 1024, 2048\}$	1024
Critic hidden size	$\{128, 256, 512, 1024\}$	256
# of hidden layers	$\{2, 3, 4\}$	4
Activation function	{Tanh, ReLU, LeakyReLU}	LeakyReLU
Batch size	$\{128, 256\}$	256
Discount factor	-	0.99
Target entropy	$\{-4.0, -2.0, 0.0\}$	-2.0
Target interpolation	-	5×10^{-3}
Replay buffer size	-	10^6
Exploratory steps	-	10^4
Optimizer	-	Adam
Learning rate	$\{3 \times 10^{-4}, 1 \times 10^{-3}\}$	3×10^{-4}
CoTASP-specific Hyperparameters		
Sparsity parameter λ	$\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$	10^{-3}
Coding algorithm	-	LARS-Lasso
Constant c	-	1.0

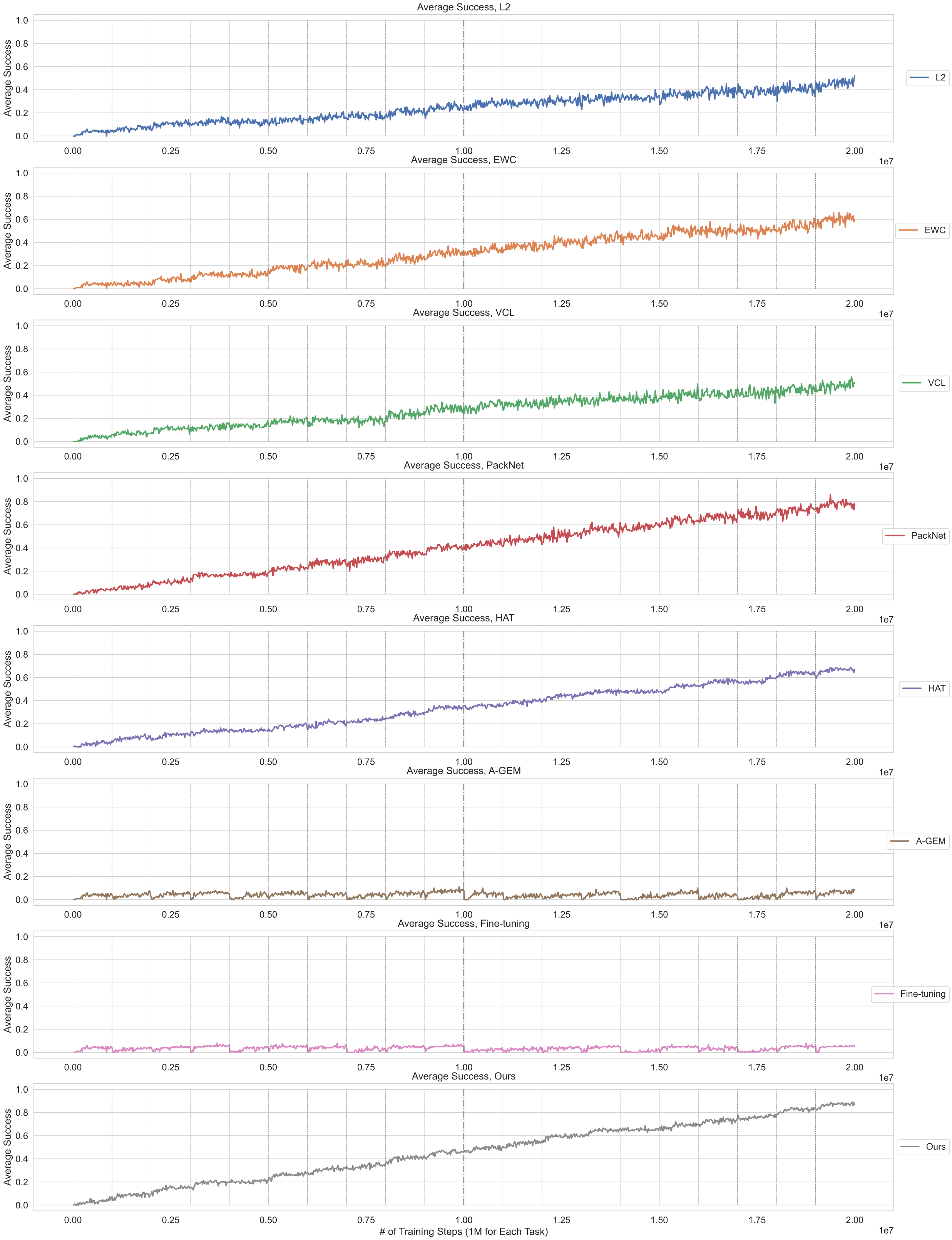


Figure 9: Average success rate over CW20 sequence for each tested method.

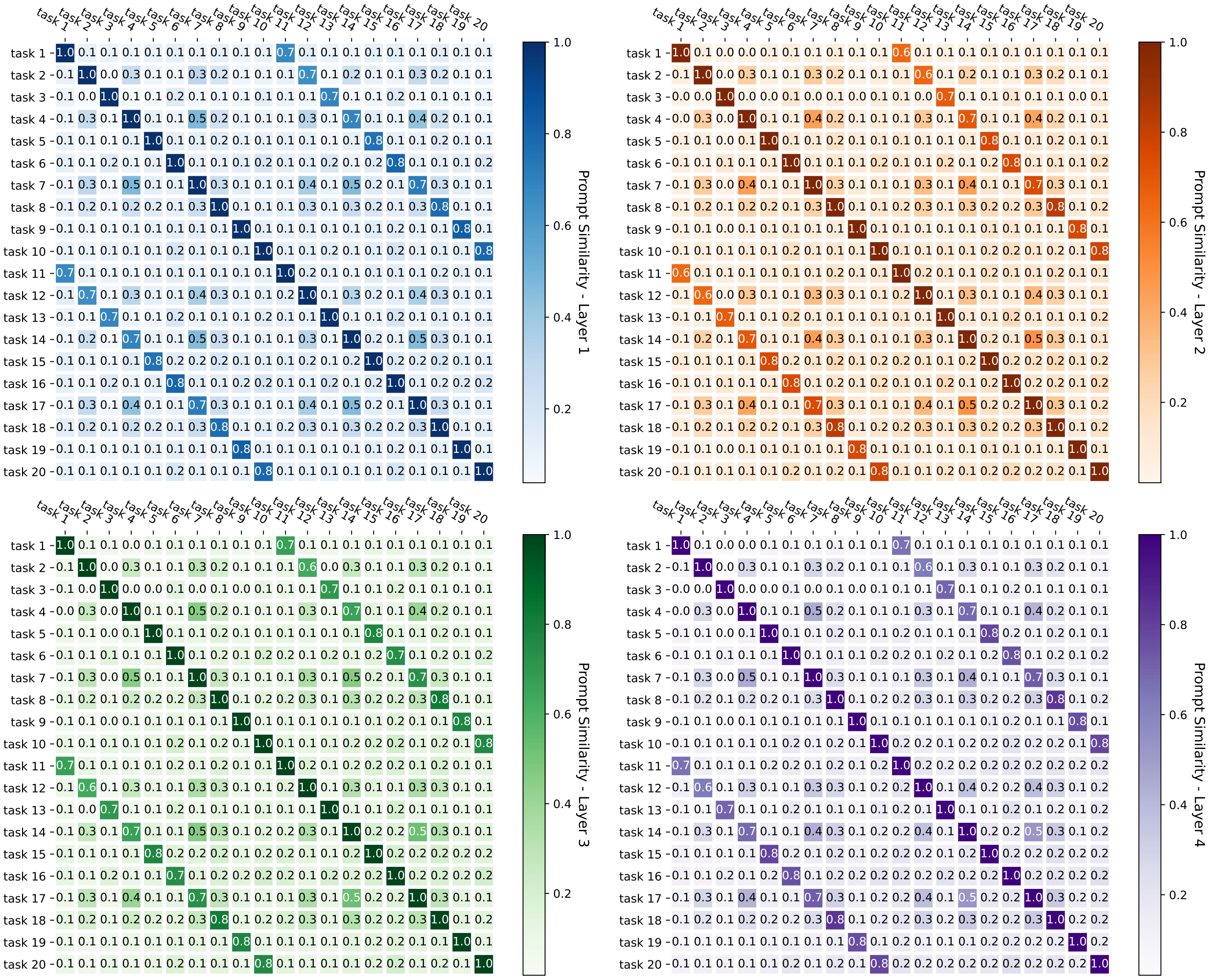


Figure 10: The similarity of each layer's prompts between every two tasks over CW20 sequence. Note that the task sequence 11-20 are repetitive CW10 sequence. We observe that the prompts generated by CoTASP are highly correlated among repetitive tasks (e.g., tasks 1 and 11), reflecting the inherent capabilities of CoTASP for knowledge transfer and task inference.

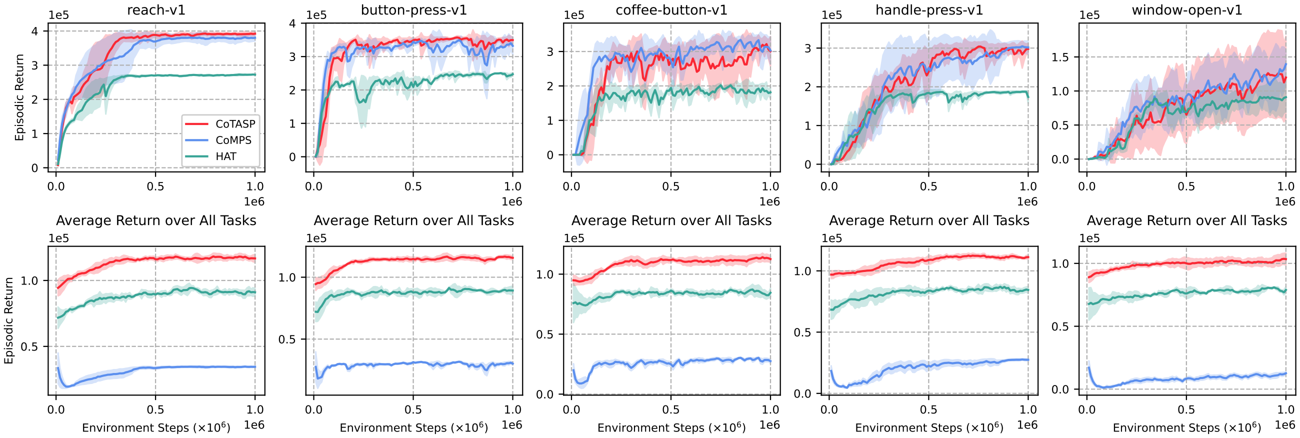


Figure 11: Results of different methods in environments from Meta-World (Yu et al., 2019). All curves are the average of 6 runs with different seeds, and the shaded areas are standard errors of the mean.