CUP: Critic-Guided Policy Reuse

Jin Zhang[1], **Siyuan Li**[2], **Chongjie Zhang**[1]
[1]Institute for Interdisciplinary Information Sciences, Tsinghua University, China
[2]School of Computer Science and Technology, Harbin Institute of Technology, China
jin-zhan20@mails.tsinghua.edu.cn
lisiyuan199511@gmail.com
chongjie@tsinghua.edu.cn

Abstract

The ability to reuse previous policies is an important aspect of human intelligence. To achieve efficient policy reuse, a Deep Reinforcement Learning (DRL) agent needs to decide when to reuse and which source policies to reuse. Previous methods solve this problem by introducing extra components to the underlying algorithm, such as hierarchical high-level policies over source policies, or estimations of source policies' value functions on the target task. However, training these components induces either optimization non-stationarity or heavy sampling cost, significantly impairing the effectiveness of transfer. To tackle this problem, we propose a novel policy reuse algorithm called Critic-gUided Policy reuse (CUP), which avoids training any extra components and efficiently reuses source policies. CUP utilizes the critic, a common component in actor-critic methods, to evaluate and choose source policies. At each state, CUP chooses the source policy that has the largest one-step improvement over the current target policy, and forms a guidance policy. The guidance policy is theoretically guaranteed to be a monotonic improvement over the current target policy. Then the target policy is regularized to imitate the guidance policy to perform efficient policy search. Empirical results demonstrate that CUP achieves efficient transfer and significantly outperforms baseline algorithms.

1 Introduction

Human intelligence can solve new tasks quickly by reusing previous policies (Guberman & Greenfield, 1991). Despite remarkable success, current Deep Reinforcement Learning (DRL) agents lack this knowledge transfer ability (Silver et al., 2017; Vinyals et al., 2019; Ceron & Castro, 2021), leading to enormous computation and sampling cost. As a consequence, a large number of works have been studying the problem of policy reuse in DRL, i.e., how to efficiently reuse source policies to speed up target policy learning (Fernández & Veloso, 2006; Barreto et al., 2018; Li et al., 2019; Yang et al., 2020b).

A fundamental challenge towards policy reuse is: how does an agent with access to multiple source policies decide when and where to use them (Fernández & Veloso, 2006; Kurenkov et al., 2020; Cheng et al., 2020)? Previous methods solve this problem by introducing additional components to the underlying DRL algorithm, such as hierarchical high-level policies over source policies (Li et al., 2018, 2019; Yang et al., 2020b), or estimations of source policies' value functions on the target task (Barreto et al., 2017, 2018; Cheng et al., 2020). However, training these components significantly impairs the effectiveness of transfer, as hierarchical structures induce optimization non-stationarity (Pateria et al., 2021), and estimating the value functions for every source policy is computationally expensive and with high sampling cost. Thus, the objective of this study is to address the question:

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Notice that actor-critic methods (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018) learn a critic that approximates the actor's Q function and serves as a natural way to evaluate policies. Based on this observation, we propose a novel policy reuse algorithm that utilizes the critic to choose source policies. The proposed algorithm, called *Critic-gUided Policy reuse (CUP)*, avoids training any additional components and achieves efficient transfer. At each state, CUP chooses the source policy that has the largest one-step improvement over the current target policy, thus forming a guidance policy. Then CUP guides learning by regularizing the target policy to imitate the guidance policy. This approach has the following advantages. First, the one-step improvement can be estimated simply by querying the critic, and no additional components are needed to be trained. Secondly, the guidance policy is theoretically guaranteed to be a monotonic improvement over the current target policy, which ensures that CUP can reuse the source policies to improve the current target policy. Finally, CUP is conceptually simple and easy to implement, introducing very few hyper-parameters to the underlying algorithm.

We evaluate CUP on Meta-World (Yu et al., 2020), a popular reinforcement learning benchmark composed of multiple robot arm manipulation tasks. Empirical results demonstrate that CUP achieves efficient transfer and significantly outperforms baseline algorithms.

2 Preliminaries

Reinforcement learning (RL) deals with Markov Decision Processes (MDPs). A MDP can be modelled by a tuple $(\mathcal{S}, \mathcal{A}, r, p, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r(s, a)$, transition function $p(s'|s, a)$, and discount factor γ (Sutton & Barto, 2018). In this study, we focus on MDPs with continuous action spaces. RL's objective is to find a policy $\pi(a|s)$ that maximizes the cumulative discounted return $R(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$.

While CUP is generally applicable to a wide range of actor-critic algorithms, in this work we use SAC (Haarnoja et al., 2018) as the underlying algorithm. The soft Q function and soft V function (Haarnoja et al., 2017) of a policy π are defined as:

$$Q_\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)} \left[V_\pi(s)\right] \tag{1}$$

$$V_\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} \left[Q_\pi(s, a) - \alpha \log \pi(a|s)\right], \tag{2}$$

where $\alpha > 0$ is the entropy weight. SAC's loss functions are defined as:

$$\begin{aligned} L_{critic}(Q_\theta) &= \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[Q_\theta(s, a) - (r + \gamma V_{\overline{\theta}}(s'))\right]^2 \\ L_{actor}(\pi_\phi) &= \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{a \sim \pi_\phi(\cdot|s)} \left[\alpha \log \pi_\phi(a|s) - Q_\theta(s, a)\right]\right] \\ L_{entropy}(\alpha) &= \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{a \sim \pi_\phi(\cdot|s)} \left[-\alpha \log \pi_\phi(a|s) - \alpha \overline{\mathcal{H}}\right]\right], \end{aligned} \tag{3}$$

where $\mathcal{D}$ is the replay buffer, $\overline{\mathcal{H}}$ is a hyper-parameter representing the target entropy, θ and ϕ are network parameters, $\overline{\theta}$ is target network's parameters, and $V_{\overline{\theta}}(s) = \mathbb{E}_{a \sim \pi(a|s)}[Q_{\overline{\theta}}(s, a) - \alpha \log \pi(a|s)]$ is the target soft value function.

We define the *soft expected advantage* of action probability distribution $\pi_i(\cdot|s)$ over policy π_j at state s as:

$$EA_{\pi_j}(s, \pi_i) = \mathbb{E}_{a \sim \pi_i(\cdot|s)} \left[Q_{\pi_j}(s, a) - \alpha \log \pi_i(a|s) - V_{\pi_j}(s)\right]. \tag{4}$$

$EA_{\pi_j}(s, \pi_i)$ measures the one-step performance improvement brought by following π_i instead of π_j at state s, and following π_j afterwards.

The field of policy reuse focuses on solving a target MDP M efficiently by transferring knowledge from a set of source policies $\{\pi_1, \pi_2, ..., \pi_n\}$. We denote the target policy learned on M at iteration t as π_{tar}^t, and its corresponding soft Q function as $Q_{\pi_{tar}^t}$. In this work, we assume that the source policies and the target policy share the same state and action spaces.

3 Critic-Guided Policy Reuse

This section presents CUP, an efficient policy reuse algorithm that does not require training any additional components. CUP is built upon actor-critic methods. In each iteration, CUP uses the critic to form a guidance policy from the source policies and the current target policy. Then CUP guides policy search by regularizing the target policy to imitate the guidance policy. Section 3.1 presents how to form a guidance policy by aggregating source policies through the critic, and proves that the guidance policy is guaranteed to be a monotonic improvement over the current target policy. We also prove that the target policy is theoretically guaranteed to improve by imitating the guidance policy. Section 3.2 presents the overall framework of CUP.

3.1 Critic-Guided Source Policy Aggregation

CUP utilizes action probabilities proposed by source policies to improve the current target policy, and forms a guidance policy. At iteration t of target policy learning, for each state s, the agent has access to a set of candidate action probability distributions proposed by the n source policies and the current target policy: $\Pi_t^s = \{\pi_1(\cdot|s), \pi_2(\cdot|s), ..., \pi_n(\cdot|s), \pi_{tar}^t(\cdot|s)\}$. The guidance policy π_g^t can be formed by combining the action probability distributions that have the largest soft expected advantage over π_{tar}^t at each state s:

$$\pi_g^t(\cdot|s) = \arg\max_{\pi(\cdot|s)\in\Pi_t^s} EA_{\pi_{tar}^t}(s,\pi) = \arg\max_{\pi(\cdot|s)\in\Pi_t^s} \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q_{\pi_{tar}^t}(s,a) - \alpha\log\pi(a|s)\right] \ for\ all\ s\in\mathcal{S}. \tag{5}$$

The second equation holds as adding $V_{\pi_{tar}^t}(s)$ to all soft expected advantages does not affect the result of the $\arg\max$ operator. Eq. 5 implies that at each state, we can choose which source policy to follow simply by querying its expected soft Q value under π_{tar}^t. Noticing that with function approximation, the exact soft Q value cannot be acquired. The following theorem enables us to form the guidance policy with an approximated soft Q function, and guarantees that the guidance policy is a monotonic improvement over the current target policy.

Theorem 1 *Let $\widetilde{Q}_{\pi_{tar}^t}$ be an approximation of $Q_{\pi_{tar}^t}$ such that*

$$|\widetilde{Q}_{\pi_{tar}^t}(s,a) - Q_{\pi_{tar}^t}(s,a)| \le \epsilon \ for\ all\ s\in\mathcal{S}, a\in A. \tag{6}$$

Define

$$\widetilde{\pi_g^t}(\cdot|s) = \arg\max_{\pi(\cdot|s)\in\Pi_t^s} \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\widetilde{Q}_{\pi_{tar}^t}(s,a) - \alpha\log\pi(a|s)\right] \ for\ all\ s\in\mathcal{S}. \tag{7}$$

Then,

$$V_{\widetilde{\pi_g^t}}(s) \ge V_{\pi_{tar}^t}(s) - \frac{2\epsilon}{1-\gamma}\ for\ all\ s\in\mathcal{S}. \tag{8}$$

Theorem 1 provides a way to choose source policies using an approximation of the current target policy's soft Q value. As SAC learns such an approximation, the guidance policy can be formed without training any additional components.

The next question is, how to incorporate the guidance policy $\widetilde{\pi_g^t}$ into target policy learning? The following theorem demonstrates that policy improvement can be guaranteed if the target policy is optimized to stay close to the guidance policy.

Theorem 2 *If*

$$D_{KL}\left(\pi_{tar}^{t+1}(\cdot|s)||\widetilde{\pi_g^t}(\cdot|s)\right) \le \delta\ for\ all\ s\in\mathcal{S}, \tag{9}$$

then

$$V_{\pi_{tar}^{t+1}}(s) \ge V_{\pi_{tar}^t}(s) - \frac{\sqrt{2\ln 2\delta}(\widetilde{R}_{max} + \alpha\mathcal{H}_{max}^{t+1})}{(1-\gamma)^2} - \frac{2\epsilon + \alpha\widetilde{\mathcal{H}}_{max}}{1-\gamma}\ for\ all\ s\in\mathcal{S}, \tag{10}$$

where $\widetilde{R}_{max} = \max_{s,a}|r(s,a)|$ is the largest possible absolute value of the reward, $\mathcal{H}_{max}^{t+1} = \max_s \mathcal{H}(\pi_{tar}^{t+1}(\cdot|s))$ is the largest entropy of π_{tar}^{t+1}, and $\widetilde{\mathcal{H}}_{max} = \max_s \left|\mathcal{H}(\pi_{tar}^t(\cdot|s)) - \mathcal{H}(\pi_{tar}^{t+1}(\cdot|s))\right|$ is the largest possible absolute difference of the policy entropy.

According to Theorem 2, the target policy can be improved by minimizing the KL divergence between the target policy and the guidance policy. Thus we can use the KL divergence as an auxiliary loss to guide target policy learning. Proofs of this section are deferred to Appendix B.1 and Appendix B.2. Theorem 1 and Theorem 2 can be extended to common "hard" value functions (deferred to Appendix B.3), so CUP is also applicable to actor-critic algorithms that uses "hard" Bellman updates, such as A3C (Mnih et al., 2016).

3.2 CUP Framework

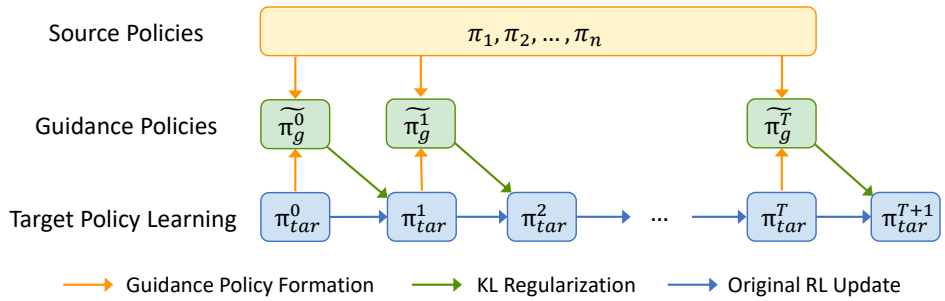


Figure 1: CUP framework. In each iteration, CUP first forms a guidance policy by querying the critic, then guides policy learning by adding a KL regularization to policy search.

In this subsection we propose the overall framework of CUP. As shown in Fig. 1, at each iteration t, CUP first forms a guidance policy $\widetilde{\pi_g^t}$ according to Eq. 7, then provides additional guidance to policy search by regularizing the target policy π_{tar}^{t+1} to imitate $\widetilde{\pi_g^t}$ (Wu et al., 2019; Fujimoto & Gu, 2021). Specifically, CUP minimizes the following loss to optimize π_{tar}^{t+1}:

$$L_{CUP}(\pi_{tar}^{t+1}) = L_{actor}(\pi_{tar}^{t+1}) + \mathbb{E}_{s\sim\mathcal{D}}\left[\beta_s D_{KL}\left(\pi_{tar}^{t+1}(\cdot|s)\,||\widetilde{\pi_g^t}(\cdot|s)\right)\right], \tag{11}$$

where L_{actor} is the original actor loss defined in Eq. (3), and $\beta_s > 0$ is a hyper-parameter controlling the weight of regularization. In practice, we find that using a fixed weight for regularization has two problems. First, it is difficult to balance the scale between L_{actor} and the regularization term, because L_{actor} grows as the Q value gets larger. Secondly, a fixed weight cannot reflect the agent's confidence on $\widetilde{\pi_g^t}$. For example, when no source policies have positive soft expected advantages, $\widetilde{\pi_g^t} = \pi_{tar}^t$. Then the agent should not imitate $\widetilde{\pi_g^t}$ anymore, as $\widetilde{\pi_g^t}$ cannot provide any guidance to further improve performance. Noticing that the soft expected advantage serves as a natural confidence measure, we weight the KL divergence with corresponding soft expected advantage at that state:

$$\beta_s = \beta_1 \min\left(\widetilde{EA}_{\pi_{tar}^t}(s, \widetilde{\pi_g^t}), \beta_2 |\widetilde{V}_{\pi_{tar}^t}(s)|\right), \tag{12}$$

where $\widetilde{EA}_{\pi_{tar}^t}(s, \widetilde{\pi_g^t}) = \mathbb{E}_{a\sim\widetilde{\pi_g^t}(\cdot|s)}\left[\widetilde{Q}_{\pi_{tar}^t}(s,a) - \alpha \log \pi_g^t(a|s) - \widetilde{V}_{\pi_{tar}^t}(s)\right]$ is the approximated soft expected advantage, $\beta_1, \beta_2 > 0$ are two hyper-parameters, and $\widetilde{V}_{\pi_{tar}^t}(s) = \mathbb{E}_{a\sim\pi_{tar}^t(\cdot|s)}\left[\widetilde{Q}_{\pi_{tar}^t}(s,a) - \alpha \log \pi_{tar}^t(a|s)\right]$ is the approximated soft value function. This adaptive regularization weight automatically balances between the two losses, and ignores the regularization term at states where $\widetilde{\pi_g^t}$ cannot improve over π_{tar}^t anymore. We further upper clip the expected advantage with the absolute value of $\beta_2 \widetilde{V}_{\pi_{tar}^t}$ to avoid the agent being overly confident about $\widetilde{\pi_g^t}$ due to function approximation error ϵ.

CUP's pseudo-code is presented in Alg. 1. The modifications CUP made to SAC are marked in red. Additional implementation details are deferred to Appendix D.1.

Algorithm 1 CUP

Require: Source policies $\{\pi_1, \pi_2, ..., \pi_n\}$, hyper-parameters $\lambda_{\theta_1}, \lambda_{\theta_2}, \lambda_\pi, \lambda_\alpha, \tau, \overline{\mathcal{H}}, \beta_1, \beta_2$
Initialize replay buffer $\mathcal{D}$
Initialize actor π_ϕ, entropy weight α, critic Q_{θ_1},Q_{θ_2}, target networks $Q_{\bar{\theta}_1} \leftarrow Q_{\theta_1}, Q_{\bar{\theta}_2} \leftarrow Q_{\theta_2}$
while not done **do**
  for each environment step **do**
    $a_t \sim \pi_\theta$
    $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, a_t, r(s_t, a_t), s_{t+1}\}$
  end for
  for each gradient step **do**
    Sample minibatch b from $\mathcal{D}$
    Query source policies' action probabilities $\{\pi_1(\cdot|s), \pi_2(\cdot|s), ..., \pi_n(\cdot|s)\}$ for states in b
    Compute expected advantages according to Eq. (4), form $\widetilde{\pi_g^t}$ according to Eq. (7)
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} L_{critic}(Q_{\theta_i})$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi L_{CUP}(\pi_\phi)$
    $\alpha \leftarrow \alpha - \lambda_\alpha \hat{\nabla}_\alpha L_{entropy}(\alpha)$
    $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$
  end for
end while

4 Experiments

We evaluate on Meta-World (Yu et al., 2020), a popular reinforcement learning benchmark composed of multiple robot manipulation tasks. These tasks are both correlated (performed by the same Sawyer robot arm) and distinct (interacting with different objects and having different reward functions), and serve as a proper evaluation benchmark for policy reuse. The source policies are achieved by training on three representative tasks: Reach, Push, and Pick-Place. We choose several complex tasks as target tasks, including Hammer, Peg-Insert-Side, Push-Wall, Pick-Place-Wall, Push-Back, and Shelf-Place. Among these target tasks, Hammer and Peg-Insert-Side require interacting with objects unseen in the source tasks. In Push-Wall and Pick-Place-Wall, there is a wall between the object and the goal. In Push-Back, the goal distribution is different from Push. In Shelf-Place, the robot is required to put a block on a shelf, and the shelf is unseen in the source tasks. Video demonstrations of these tasks are available at `https://meta-world.github.io/`. Similar to the settings in Yang et al. (2020a), in our experiments the goal position is randomly reset at the start of every episode. Codes are available at `https://github.com/NagisaZj/CUP`.

4.1 Transfer Performance on Meta-World

We compare against several representative baseline algorithms, including HAAR (Li et al., 2019), PTF (Yang et al., 2020b), MULTIPOLAR (Barekatain et al., 2021), and MAMBA (Cheng et al., 2020). Among these algorithms, HAAR and PTF learn hierarchical high-level policies over source policies. MAMBA aggregates source policies' V functions to form a baseline function, and performs policy improvement over the baseline function. MULTIPOLAR learns a weighted sum of source policies' action probabilities, and learns an additional network to predict residuals. We also compare against the original SAC algorithm. All the results are averaged over six random seeds. As shown in Figure 2, CUP is the only algorithm that achieves efficient transfer on all six tasks, significantly outperforming the original SAC algorithm. HAAR has a jump-start performance on Push-Wall and Pick-Pick-Wall, but fails to further improve due to optimization non-stationarity induced by jointly training high-level and low-level policies. MULTIPOLAR achieves comparable performance on Push-Wall and Peg-Insert-Side, because the Push source policy is useful on Push-Wall (implied by HAAR's good jump-start performance), and learning residuals on Peg-Insert-Side is easier (implied by SAC's fast learning). In Pick-Place-Wall, the Pick-Place source policy is useful, but the residual is difficult to learn, so MULTIPOLAR does not work. For the remaining three tasks, the source policies are less useful, and MULTIPOLAR fails on these tasks. PTF fails as its hierarchical policy only gets updated when the agent chooses similar actions to one of the source policies, which is quite rare when

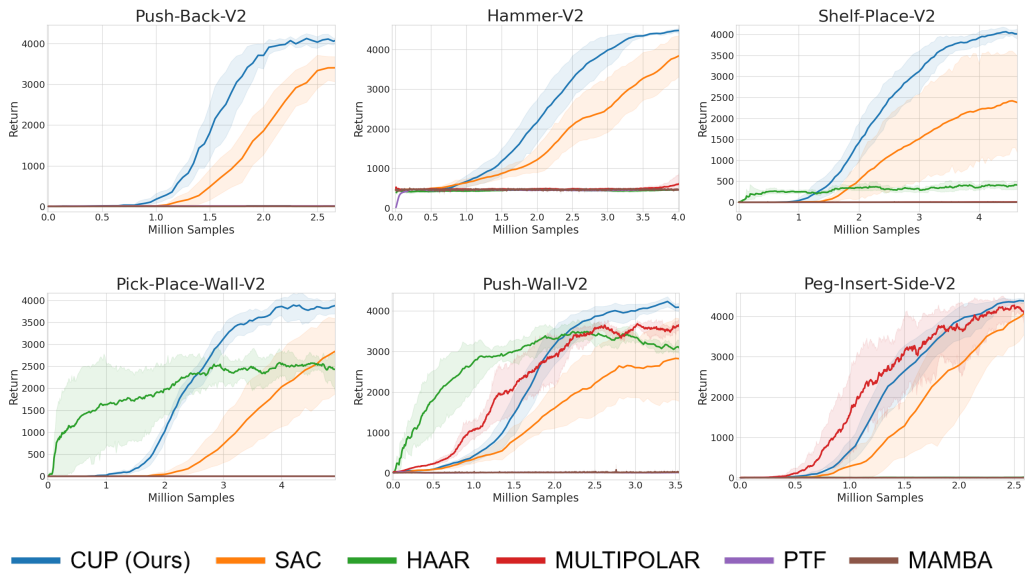


Figure 2: Evaluation of CUP and several baselines on various Meta-World tasks. Dashed areas represent 95% bootstrapped confidence intervals. CUP achieves substantially better performance than baseline algorithms.

the source and target tasks are distinct. MAMBA fails as estimating all source policies' V functions accurately is sampling inefficient. Algorithm performance evaluated by success rate is deferred to Appendix E.1.

4.2 Analyzing the Guidance Policy

This subsection provides visualizations of CUP's source policy selection. Fig. 3 shows the percentages of each source policy being selected throughout training on Push-Wall. At early stages of training, the source policies are selected more frequently as they have positive expected advantages, which means that they can be used to improve the current target policy. As training proceeds and the target policy becomes better, the source policies are selected less frequently. Among these three source policies, Push is chosen more frequently than the other two source policies, as it is more related to the target task. Figure 4 presents the source policies' expected advantages over an episode at convergence in Pick-Place-Wall. The Push source policy and Reach source policy almost always have negative expected advantages, which implies that these two source policies can hardly improve the current target policy anymore. Meanwhile, the Pick-Place source policy has expected advantages close to zero after 100 environment steps, which implies that the Pick-Place source policy is close to the target policy at these steps. Analyses on all six tasks as well as analyses on HAAR's source policy selection are deferred to Appendix E.2 and Appendix E.6, respectively.

4.3 Ablation Study

This subsection evaluates CUP's sensitivity to hyper-parameter settings and the number of source policies. We also evaluate CUP's robustness against random source policies, which do not provide meaningful candidate actions for solving target tasks.

4.3.1 Hyper-Parameter Sensitivity

For all the experiments in Section 4.1, we use the same set of hyper-parameters, which indicates that CUP is generally applicable to a wide range of tasks without particular fine-tuning. CUP introduces only two additional hyper-parameters to the underlying SAC algorithm, and we further test CUP's sensitivity to these additional hyper-parameters. As shown in Fig. 5, CUP is generally robust to the choice of hyper-parameters and achieves stable performance.

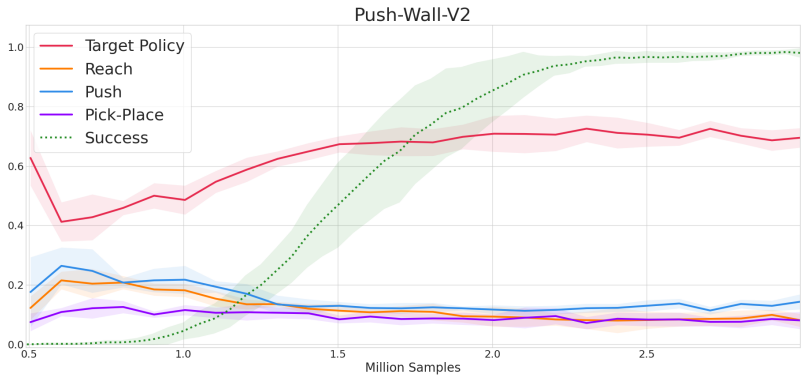


Figure 3: Percentages of source policies being selected by CUP during training on Push-Wall. The green dashed line represents the target policy's success rate on the task.

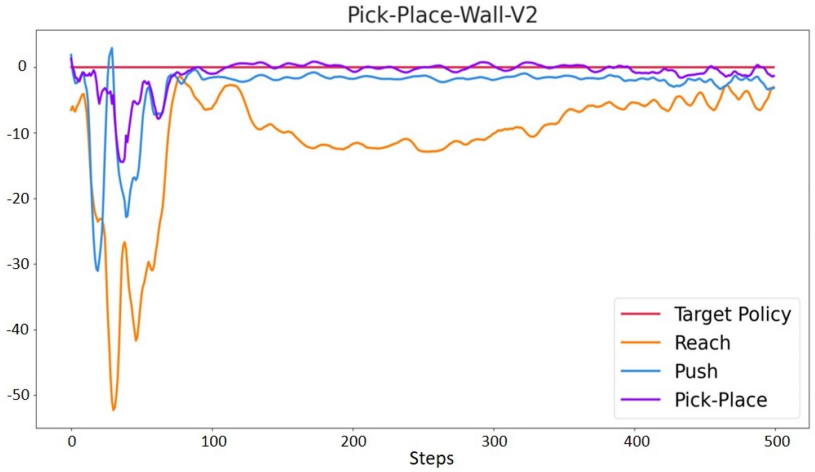


Figure 4: Expected advantages of source policies at convergence on Pick-Place-Wall. The horizontal axis represents the environment steps of an episode.

4.3.2 Number of Source Policies

We evaluate CUP as well as baseline algorithms on a larger source policy set. We add three policies to the original source policy set, which solve three simple tasks including Drawer-Close, Push-Wall, and Coffee-Button. This forms a source policy set composed of six policies. As shown in Fig. 6, CUP is still the only algorithm that solves all the six target tasks efficiently. MULTIPOLAR suffers from a decrease in performance, which indicates that learning the weighted sum of source policies' actions becomes more difficult as the number of source policies grows. The rest of the baseline algorithms have similar performance to those using three source policies. Fig. 7 provides a more direct comparison of CUP's performance with different number of source policies. CUP is able to utilize the additional source policies to further improve its performance, especially on Pick-Place-Wall and Peg-Insert-Side. Further detailed analysis is deferred to Appendix E.3.

4.3.3 Interference of Random Source Policies

In order to evaluate the efficiency of CUP's critic-guided source policy aggregation, we add random policies to the set of source policies. As shown in Fig. 8(a), adding up to 3 random source policies does not affect CUP's performance. This indicates that CUP can efficiently choose which source policy to follow even if there exist many source policies that are not meaningful. Adding 4 and 5 random source policies leads to a slight drop in performance. This drop is because that as the number

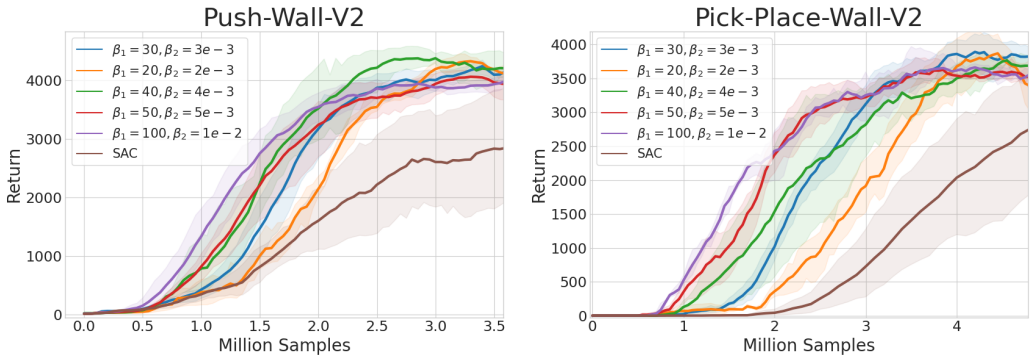


Figure 5: Ablation studies on a wide range of hyper-parameters. CUP performs well on a wide range of hyper-parameters.

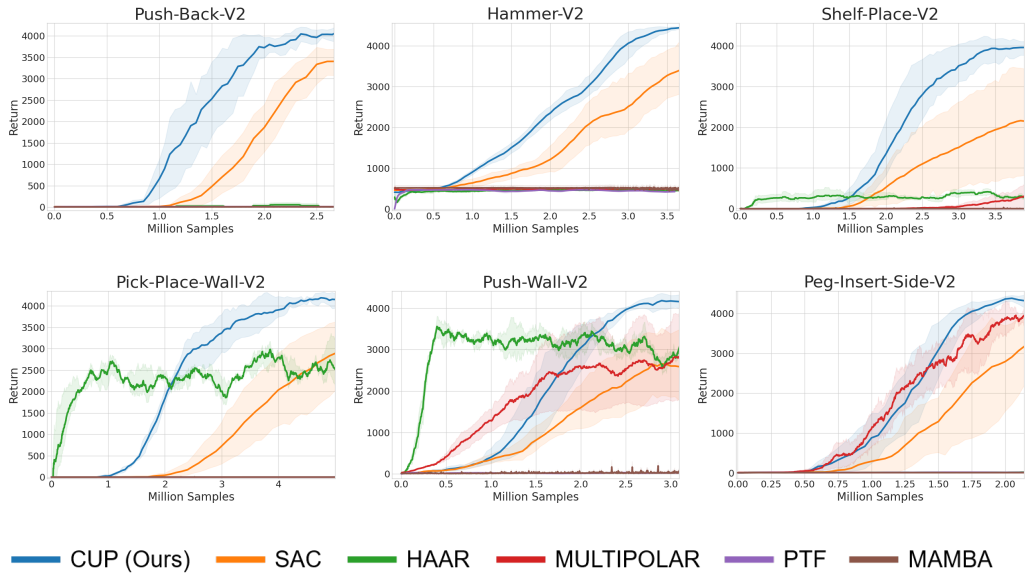


Figure 6: Performance of CUP and baseline algorithms on various Meta-World tasks, with a set of six source policies.

of random policies grows, more random actions are sampled, and taking argmax over these actions' expected advantages is more likely to be affected by errors in value estimation.

To further investigate CUP's ability to ignore unsuitable source policies, we design another transfer setting that consists of another two source policy sets. The first set consists of three random policies that are useless for the target task, and the second set adds the Reach policy to the first set. As demonstrated in Fig. 8(b), when none of the source policies are useful, CUP performs similarly to the original SAC, and its sample efficiency is almost unaffected by the useless source policies. When there exists a useful source policy, CUP can efficiently utilize it to improve performance, even if there are many useless source policies.

5 Related Work

Policy reuse. A series of works on policy reuse utilize source policies for exploration in value-based algorithms (Fernández & Veloso, 2006; Li & Zhang, 2018; Gimelfarb et al., 2021), but they are not applicable to policy gradient methods due to the off-policyness problem (Fujimoto et al., 2019). AC-Teach (Kurenkov et al., 2020) mitigates this problem by improving the actor over behavior policy's

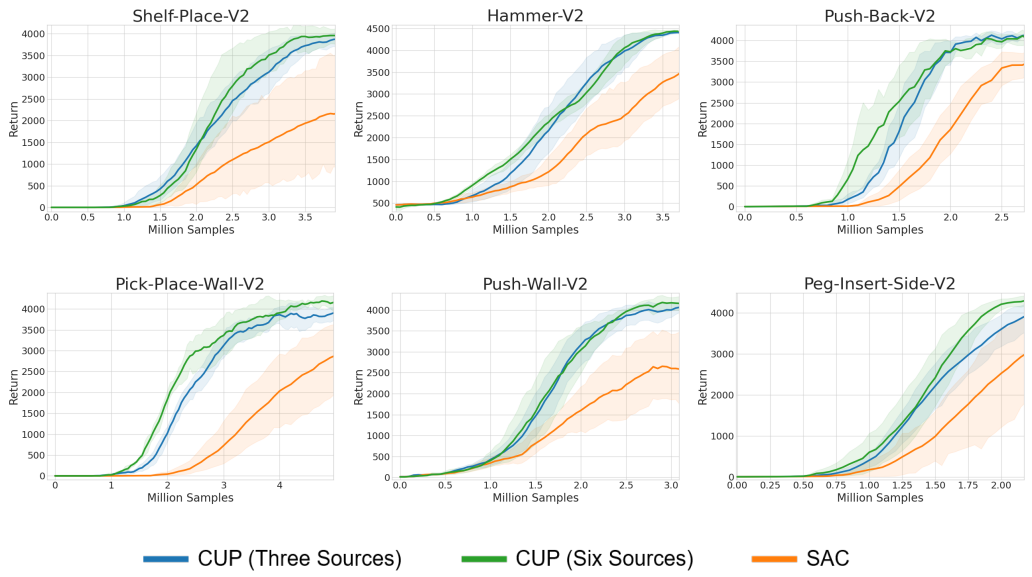
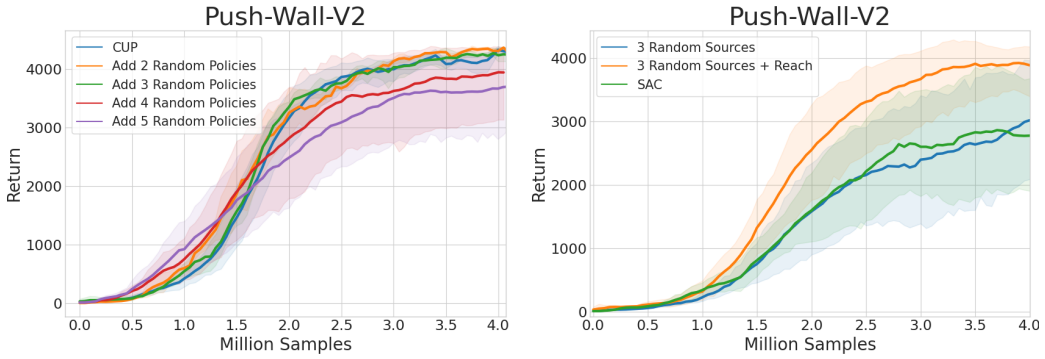


Figure 7: Comparison of CUP's performance with different number of source policies.



(a) (b)

Figure 8: Ablation studies on CUP's sensitivity to useless source policies. (a) Adding up to 3 random policies to the source policy set does not affect CUP's performance. (b) Ablation study in a setting where most source policies are useless. If none of the source policies are useful (3 Random Sources), CUP performs similarly to the original SAC. Even if only one of the four source policies is useful (3 Random Sources+Reach), CUP is still able to efficiently utilize the useful source policy to improve learning performance.

value estimation, but still fails in more complex tasks. One branch of methods train hierarchical high-level policies over source policies. CAPS (Li et al., 2018) guarantees the optimality of the hierarchical policies by adding primitive skills to the low-level policy set, but is inapplicable to MDPs with continuous action spaces. HAAR (Li et al., 2019) fine-tunes low-level policies to ensure optimality, but joint training of high-level and low-level policies induce optimization non-stationarity (Pateria et al., 2021). PTF (Yang et al., 2020b) trains a hierarchical policy, which is imitated by the target policy. However, the hierarchical policy only gets updated when the target policy chooses similar actions to one of the source policies, so PTF fails in complex tasks with large action spaces. Another branch of works aggregate source policies via their Q functions or V functions on the target task. Barreto et al. (2017) and Barreto et al. (2018) focus on the situation where source tasks and target tasks share the same dynamics, and aggregate source policies by choosing the policy that has the largest Q at each state. They use successor features to mitigate the heavy computation cost

brought by estimating Q functions for all source policies. MAMBA (Cheng et al., 2020) forms a baseline function by aggregating source policies' V functions, and guides policy search by improving the policy over the baseline function. Finally, MULTIPOLAR (Barekatain et al., 2021) learns a weighted sum over source policies' actions, and learns an auxiliary network to predict residuals around the aggregated actions. MULTIPOLAR is computationally expensive, as it requires querying all the source policies at every sampling step. Our proposed method, CUP, focuses on the setting of learning continuous-action MDPs with actor-critic methods. CUP is both computationally and sampling efficient, as it does not require training any additional components.

Policy regularization. Adding regularization to policy optimization is a common approach to induce prior knowledge into policy learning. Distral (Teh et al., 2017) achieves inter-task transfer by imitating an average policy distilled from policies of related tasks. In offline RL, policy regularization serves as a common technique to keep the policy close to the behavior policy used to collect the dataset (Wu et al., 2019; Nair et al., 2020; Fujimoto & Gu, 2021). CUP uses policy regularization as a means to provide additional guidance to policy search with the guidance policy.

6 Conclusion

In this study, we address the problem of reusing source policies without training any additional components. By utilizing the critic as a natural evaluation of source policies, we propose CUP, an efficient policy reuse algorithm without training any additional components. CUP is conceptually simple, easy to implement, and has theoretical guarantees. Empirical results demonstrate that CUP achieves efficient transfer on a wide range of tasks. As for future work, CUP assumes that all source policies and the target policy share the same state and action spaces, which limits CUP's application to more general scenarios. One possible future direction is to take inspiration from previous works that map the state and action spaces of an MDP to another MDP with similar high-level structure (Wan et al., 2020; Zhang et al., 2020; Heng et al., 2022; van der Pol et al., 2020b,a). Another interesting direction is to incorporate CUP into the continual learning setting (Rolnick et al., 2019; Khetarpal et al., 2020), in which an agent gradually enriches its source policy set in an online manner.

Acknowledgements

This work is supported in part by Science and Technology Innovation 2030 – "New Generation Artificial Intelligence" Major Project (No. 2018AAA0100904), National Natural Science Foundation of China (62176135), and China Academy of Launch Vehicle Technology (CALT2022-18).

References

Barekatain, M., Yonetani, R., and Hamaya, M. Multipolar: multi-source policy aggregation for transfer reinforcement learning between diverse environmental dynamics. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 3108–3116, 2021.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., and Munos, R. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pp. 501–510. PMLR, 2018.

Ceron, J. S. O. and Castro, P. S. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning*, pp. 1373–1383. PMLR, 2021.

Cheng, C.-A., Kolobov, A., and Agarwal, A. Policy improvement via imitation of multiple oracles. *Advances in Neural Information Processing Systems*, 33:5587–5598, 2020.

Fedotov, A. A., Harremoës, P., and Topsoe, F. Refinements of pinsker's inequality. *IEEE Transactions on Information Theory*, 49(6):1491–1498, 2003.

Fernández, F. and Veloso, M. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 720–727, 2006.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Gimelfarb, M., Sanner, S., and Lee, C.-G. Contextual policy transfer in reinforcement learning domains via deep mixtures-of-experts. In *Uncertainty in Artificial Intelligence*, pp. 1787–1797. PMLR, 2021.

Guberman, S. R. and Greenfield, P. M. Learning and transfer in everyday cognition. *Cognitive Development*, 6(3):233–260, 1991.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pp. 1352–1361. PMLR, 2017.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Heng, Y., Yang, T., ZHENG, Y., Jianye, H., and Taylor, M. E. Cross-domain adaptive transfer reinforcement learning based on state-action correspondence. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Khetarpal, K., Riemer, M., Rish, I., and Precup, D. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

Kurenkov, A., Mandlekar, A., Martin-Martin, R., Savarese, S., and Garg, A. Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. In *Conference on Robot Learning*, pp. 717–734. PMLR, 2020.

Li, S. and Zhang, C. An optimal online method of selecting source policies for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Li, S., Gu, F., Zhu, G., and Zhang, C. Context-aware policy reuse. *arXiv preprint arXiv:1806.03793*, 2018.

Li, S., Wang, R., Tang, M., and Zhang, C. Hierarchical reinforcement learning with advantage-based auxiliary rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Nair, A., Gupta, A., Dalal, M., and Levine, S. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Ostrovski, G., Castro, P. S., and Dabney, W. The difficulty of passive learning in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:23283–23295, 2021.

Pateria, S., Subagdja, B., Tan, A.-h., and Quek, C. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550 (7676):354–359, 2017.

Sodhani, S., Zhang, A., and Pineau, J. Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning*, pp. 9767–9779. PMLR, 2021.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

van der Pol, E., Kipf, T., Oliehoek, F. A., and Welling, M. Plannable approximations to mdp homomorphisms: Equivariance under actions. *arXiv preprint arXiv:2002.11963*, 2020a.

van der Pol, E., Worrall, D., van Hoof, H., Oliehoek, F., and Welling, M. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020b.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Wan, M., Gangwani, T., and Peng, J. Mutual information based knowledge transfer under state-action dimension mismatch. *arXiv preprint arXiv:2006.07041*, 2020.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yang, R., Xu, H., Wu, Y., and Wang, X. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020a.

Yang, T., Hao, J., Meng, Z., Zhang, Z., Hu, Y., Chen, Y., Fan, C., Wang, W., Wang, Z., and Peng, J. Efficient deep reinforcement learning through policy transfer. In *AAMAS*, pp. 2053–2055, 2020b.

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 *Conference on Robot Learning*, pp. 1094–1100. PMLR, 2020.

Zhang, Q., Xiao, T., Efros, A. A., Pinto, L., and Wang, X. Learning cross-domain correspondence for control with dynamics cycle-consistency. *arXiv preprint arXiv:2012.09811*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 2.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix A.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.1.

(b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.1 and Appendix B.2.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplemental materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4
   (b) Did you mention the license of the assets? [Yes] See the supplemental materials.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Broader Social Impact

We believe policy reuse serves as a promising way to transfer knowledge among AI agents. This ability will enable AI agents to master new skills efficiently. However, we are also aware of possible negative social impacts, such as plagiarizing other AI products by querying and reusing their policies.

B Proofs

B.1 Proof for Theorem 1

Proof. As $|\widetilde{Q}_{\pi^t_{tar}}(s,a) - Q_{\pi^t_{tar}}(s,a)| \leq \epsilon$ for all $s \in \mathcal{S}, a \in A$, we have that for all $s_i \in \mathcal{S}$, the difference between the true value function $V_{\pi^t_{tar}}$ and the approximated value function $\widetilde{V}_{\pi^t_{tar}}$ is bounded:

$$
\begin{aligned}
&V_{\pi^t_{tar}}(s_i) \\
&= \mathbb{E}_{a_i \sim \pi^t_{tar}(\cdot|s_i)} \left[Q_{\pi^t_{tar}}(s_i, a_i) - \alpha \log \pi^t_{tar}(a_i|s_i) \right] \\
&\leq \mathbb{E}_{a_i \sim \pi^t_{tar}(\cdot|s_i)} \left[\widetilde{Q}_{\pi^t_{tar}}(s_i, a_i) - \alpha \log \pi^t_{tar}(a_i|s_i) + \epsilon \right] \\
&= \widetilde{V}_{\pi^t_{tar}}(s_i) + \epsilon.
\end{aligned}
$$

As $\pi^t_{tar}(\cdot|s)$ is contained in Π^s_t, with $\widetilde{\pi^t_g}$ defined in Eq. (7), it is obvious that for all $s \in \mathcal{S}$, $\mathbb{E}_{a \sim \widetilde{\pi^t_g}(\cdot|s)} \left[\widetilde{Q}_{\pi^t_{tar}}(s,a) - \alpha \log \widetilde{\pi^t_g}(a|s) \right] \geq \mathbb{E}_{a \sim \pi^t_{tar}(\cdot|s)} \left[\widetilde{Q}_{\pi^t_{tar}}(s,a) - \alpha \log \pi^t_{tar}(a|s) \right] =$

$\widetilde{V}_{\pi^t_{tar}}(s)$. Then for all $s_i \in \mathcal{S}$,

$$
\begin{aligned}
&V_{\pi^t_{tar}}(s_i) \\
&\leq \widetilde{V}_{\pi^t_{tar}}(s_i) + \epsilon \\
&\leq \mathbb{E}_{a_i \sim \widetilde{\pi^t_g}(\cdot|s_i)}[\widetilde{Q}_{\pi^t_{tar}}(s_i, a_i) - \alpha \log \widetilde{\pi^t_g}(a_i|s_i)] + \epsilon \\
&\leq \mathbb{E}_{a_i \sim \widetilde{\pi^t_g}(\cdot|s_i)}[Q_{\pi^t_{tar}}(s_i, a_i) - \alpha \log \widetilde{\pi^t_g}(a_i|s_i)] + 2\epsilon \\
&= \mathbb{E}_{a_i \sim \pi^t_g(a|s_i)}[r(s_i, a_i) - \alpha \log \widetilde{\pi^t_g}(a_i|s_i) + \gamma V_{\pi^t_{tar}}(s_{i+1})] + 2\epsilon \\
&\vdots \\
&\leq \mathbb{E}_{\widetilde{\pi^t_g}}[\sum_{\tau=0}^{\infty} \gamma^\tau (r(s_{i+\tau}, a_{i+\tau}) - \alpha \log \widetilde{\pi^t_g}(a_{i+\tau}|s_{i+\tau}))] + 2\sum_{\tau=0}^{\infty} \gamma^\tau \epsilon \\
&= V_{\widetilde{\pi^t_g}}(s_i) + \frac{2\epsilon}{1-\gamma}.
\end{aligned}
$$

B.2 Proof of Theorem 2

Proof. According to Pinsker's inequality (Fedotov et al., 2003), $D_{KL}(\pi^{t+1}_{tar}(\cdot|s)||\widetilde{\pi^t_g}(\cdot|s)) \geq \frac{1}{2\ln 2}||\pi^{t+1}_{tar}(\cdot|s) - \widetilde{\pi^t_g}(\cdot|s)||_1^2$, where $||\cdot||_1$ is the L1 norm. So we have that for all $s \in \mathcal{S}$, $||\pi^{t+1}_{tar}(\cdot|s) - \widetilde{\pi^t_g}(\cdot|s)||_1 \leq \sqrt{2\ln 2\delta}$. According to the Performance Difference Lemma (Kakade & Langford, 2002), we have that for all $s \in \mathcal{S}$:

$$
\begin{aligned}
&V_{\widetilde{\pi^t_g}}(s) - V_{\pi^{t+1}_{tar}}(s) \\
&= \frac{1}{1-\gamma} \mathbb{E}_{s' \sim \mu_s^{\widetilde{\pi^t_g}}(s')} \\
&\qquad \left[\mathbb{E}_{a \sim \widetilde{\pi^t_g}(\cdot|s')}[Q_{\pi^{t+1}_{tar}}(s', a) - \alpha \log \widetilde{\pi^t_g}(a|s)] - \mathbb{E}_{a \sim \widetilde{\pi^{t+1}_{tar}}(\cdot|s')}[Q_{\pi^{t+1}_{tar}}(s', a) - \alpha \log \widetilde{\pi^{t+1}_{tar}}(a|s)]\right] \\
&\leq \frac{1}{1-\gamma} \max_{s' \in \mathcal{S}} \left[\mathbb{E}_{a \sim \widetilde{\pi^t_g}(\cdot|s')}[Q_{\pi^{t+1}_{tar}}(s', a)] - \mathbb{E}_{a \sim \pi^{t+1}_{tar}(\cdot|s')}[Q_{\pi^{t+1}_{tar}}(s', a)]\right] \\
&\qquad\qquad + \frac{\alpha}{1-\gamma} \max_{s'' \in \mathcal{S}} \left|\mathcal{H}(\widetilde{\pi^t_g}(\cdot|s'')) - \mathcal{H}(\pi^t_{tar}(\cdot|s''))\right| \\
&= \frac{1}{1-\gamma} \max_{s' \in \mathcal{S}} \int \left(\widetilde{\pi^t_g}(\cdot|s) - \pi^{t+1}_{tar}(a|s)\right) Q_{\pi^{t+1}_{tar}}(s', a) da + \frac{\alpha}{1-\gamma} \widetilde{\mathcal{H}}_{max} \\
&\leq \frac{1}{1-\gamma} \max_{s' \in \mathcal{S}} \int \left|\widetilde{\pi^t_g}(a|s) - \pi^{t+1}_{tar}(a|s)\right| \cdot \left|Q_{\pi^{t+1}_{tar}}(s', a)\right| da + \frac{\alpha}{1-\gamma} \widetilde{\mathcal{H}}_{max} \\
&\leq \frac{1}{1-\gamma} \max_{s' \in \mathcal{S}} \int \left|\widetilde{\pi^t_g}(a|s) - \pi^{t+1}_{tar}(a|s)\right| \cdot \frac{\widetilde{R}_{max} + \alpha \mathcal{H}^{t+1}_{max}}{1-\gamma} da + \frac{\alpha}{1-\gamma} \widetilde{\mathcal{H}}_{max} \\
&= \frac{\widetilde{R}_{max} + \alpha \mathcal{H}^{t+1}_{max}}{(1-\gamma)^2} \max_{s' \in \mathcal{S}} ||\widetilde{\pi^t_g}(\cdot|s) - \pi^{t+1}_{tar}(\cdot|s)||_1 + \frac{\alpha}{1-\gamma} \widetilde{\mathcal{H}}_{max} \\
&\leq \frac{\sqrt{2\ln 2\delta}(\widetilde{R}_{max} + \alpha \mathcal{H}^{t+1}_{max}) + \alpha(1-\gamma)\widetilde{\mathcal{H}}_{max}}{(1-\gamma)^2},
\end{aligned}
\tag{13}
$$

where $\mu_s^{\widetilde{\pi_g^t}}(s') = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p(s_t = s'|s_0 = s, \widetilde{\pi_g^t})$[1] is the normalized discounted state occupancy distribution. Note that

$$
\begin{aligned}
&|Q_{\pi_{tar}^{t+1}}(s,a)| \\
&= \left|\mathbb{E}_{\pi_{tar}^{t+1}}\left[\sum_{i=0}^{\infty}\gamma^i(r(s_{\tau+i},a_{\tau+i}) - \alpha\log\pi_{tar}^{t+1}(\cdot|s))|s_\tau = s, a_\tau = a\right]\right| \\
&\leq \mathbb{E}_{\pi}[\sum_{i=0}^{\infty}\gamma^i(\widetilde{R}_{max} + \gamma\mathcal{H}_{max}^{t+1})] \qquad (14) \\
&= \frac{\widetilde{R}_{max} + \alpha\mathcal{H}_{max}^{t+1}}{1-\gamma}. \qquad (15)
\end{aligned}
$$

Eventually, we have $V_{\pi_{tar}^{t+1}}(s) \geq V_{\widetilde{\pi_g^t}}(s) - \frac{\sqrt{2\ln 2\delta}(\widetilde{R}_{max}+\alpha\mathcal{H}_{max}^{t+1})+\alpha(1-\gamma)\widetilde{\mathcal{H}}_{max}}{(1-\gamma)^2} \geq V_{\pi_{tar}^t}(s) - \frac{\sqrt{2\ln 2\delta}(\widetilde{R}_{max}+\alpha\mathcal{H}_{max}^{t+1})}{(1-\gamma)^2} - \frac{2\epsilon+\alpha\widetilde{\mathcal{H}}_{max}}{1-\gamma}$.

B.3 Critic-Guided Source Policy Aggregation under "Hard" Value Functions

In this section we override the notation Q, V to represent "hard" value functions, and override the notation EA to represent the *expected advantage*, which is defined as $EA_{\pi_j}(s,\pi_i) = \mathbb{E}_{a\sim\pi_i(\cdot|s)}\left[Q_{\pi_j}(s,a) - V_{\pi_j}(s)\right]$. Then Theorem 1 and Theorem 2 can be extended as below.

Theorem 3 *Let $\widetilde{Q}_{\pi_{tar}^t}$ be an approximation of $Q_{\pi_{tar}^t}$ such that*

$$|\widetilde{Q}_{\pi_{tar}^t}(s,a) - Q_{\pi_{tar}^t}(s,a)| \leq \epsilon \textit{ for all } s\in\mathcal{S}, a\in A. \qquad (16)$$

Define

$$\widetilde{\pi_g^t}(\cdot|s) = \arg\max_{\pi(\cdot|s)\in\Pi_t^s}\mathbb{E}_{a\sim\pi(\cdot|s)}\left[\widetilde{Q}_{\pi_{tar}^t}(s,a)\right] \textit{ for each } s\in\mathcal{S}. \qquad (17)$$

Then,

$$V_{\widetilde{\pi_g^t}}(s) \geq V_{\pi_{tar}^t}(s) - \frac{2\epsilon}{1-\gamma} \textit{ for all } s\in\mathcal{S}. \qquad (18)$$

Theorem 4 *If*

$$D_{KL}\left(\pi_{tar}^{t+1}(\cdot|s)||\widetilde{\pi_g^t}(\cdot|s)\right) \leq \delta \; for \; all \; s\in\mathcal{S}, \qquad (19)$$

then

$$V_{\pi_{tar}^{t+1}}(s) \geq V_{\pi_{tar}^t}(s) - \frac{\sqrt{2\ln 2\delta}\widetilde{R}_{max}}{(1-\gamma)^2} - \frac{2\epsilon}{1-\gamma} \; for \; all \; s\in\mathcal{S}, \qquad (20)$$

where $\widetilde{R}_{max} = \max_{s,a}|r(s,a)|$ is the largest possible absolute value of the reward.

Theorem 3 and Theorem 4 implies that CUP can still guarantee policy improvement under hard Bellman updates. Proofs are given below.

Proof for Theorem 3. As $|\widetilde{Q}_{\pi_{tar}^t}(s,a) - Q_{\pi_{tar}^t}(s,a)| \leq \epsilon$ for all $s\in\mathcal{S}, a\in A$, we have that for all $s_i\in\mathcal{S}$, the difference between the true value function $V_{\pi_{tar}^t}$ and the approximated value function $\widetilde{V}_{\pi_{tar}^t}$ is bounded:

$$
\begin{aligned}
&V_{\pi_{tar}^t}(s_i) \\
&= \mathbb{E}_{a\sim\pi_{tar}^t(\cdot|s)}\left[Q_{\pi_{tar}^t}(s_i,a_i)\right] \\
&\leq \mathbb{E}_{a\sim\pi_{tar}^t(\cdot|s)}\left[\widetilde{Q}_{\pi_{tar}^t}(s_i,a_i) + \epsilon\right] \\
&= \widetilde{V}_{\pi_{tar}^t}(s_i) + \epsilon.
\end{aligned}
$$

[1] We slightly abuse the notation s_0 here to indicate that the agent start deterministically from state s.

As $\pi^t_{tar}(\cdot|s)$ is contained in Π^s_t, with π^t_g defined in Eq. (17), it is obvious that for all $s \in \mathcal{S}$, $\mathbb{E}_{a\sim\widetilde{\pi^t_g}(\cdot|s)}\left[\widetilde{Q}_{\pi^t_{tar}}(s,a)\right] \geq \mathbb{E}_{a\sim\pi^t_{tar}(\cdot|s)}\left[\widetilde{Q}_{\pi^t_{tar}}(s,a)\right] = \widetilde{V}_{\pi^t_{tar}}(s)$. Then for all $s_i \in \mathcal{S}$,

$$
\begin{aligned}
&V_{\pi^t_{tar}}(s_i)\\
&\leq \widetilde{V}_{\pi^t_{tar}}(s_i) + \epsilon\\
&\leq \mathbb{E}_{a_i\sim\widetilde{\pi^t_g}(\cdot|s_i)}[\widetilde{Q}_{\pi^t_{tar}}(s_i,a_i)] + \epsilon\\
&\leq \mathbb{E}_{a_i\sim\widetilde{\pi^t_g}(\cdot|s_i)}[Q_{\pi^t_{tar}}(s_i,a_i)] + 2\epsilon\\
&= \mathbb{E}_{a_i\sim\pi^t_g(a|s_i)}[r(s_i,a_i) + \gamma V_{\pi^t_{tar}}(s_{i+1})] + 2\epsilon\\
&\leq \mathbb{E}_{a_i\sim\widetilde{\pi^t_g}(\cdot|s_i)}[r(s_i,a_i) + \gamma(\widetilde{V}_{\pi^t_{tar}}(s_{i+1}) + \epsilon)] + 2\epsilon\\
&\leq \mathbb{E}_{a_i\sim\widetilde{\pi^t_g}(\cdot|s_i),a_{i+1}\sim\widetilde{\pi^t_g}(\cdot|s_{i+1})}[r(s_i,a_i) + \gamma\widetilde{Q}_{\pi^t_{tar}}(s_{i+1},a_{i+1})] + (2+\gamma)\epsilon\\
&\leq \mathbb{E}_{a_i\sim\widetilde{\pi^t_g}(\cdot|s_i),a_{i+1}\sim\widetilde{\pi^t_g}(\cdot|s_{i+1})}[r(s_i,a_i) + \gamma(Q_{\pi^t_{tar}}(s_{i+1},a_{i+1}) + \epsilon)] + (2+\gamma)\epsilon\\
&= \mathbb{E}_{a_i\sim\widetilde{\pi^t_g}(\cdot|s_i),a_{i+1}\sim\widetilde{\pi^t_g}(\cdot|s_{i+1})}[r(s_i,a_i) + \gamma r(s_{i+1},a_{i+1}) + \gamma^2 V_{\pi^t_{tar}}(s_{i+2})] + (2+2\gamma)\epsilon\\
&\vdots\\
&\leq \mathbb{E}_{\widetilde{\pi^t_g}}[\sum_{\tau=0}^{\infty}\gamma^\tau r(s_{i+\tau},a_{i+\tau})] + 2\sum_{\tau=0}^{\infty}\gamma^\tau\epsilon\\
&= V_{\widetilde{\pi^t_g}}(s_i) + \frac{2\epsilon}{1-\gamma}.
\end{aligned}
$$

Proof for Theorem 4. According to Pinsker's inequality (Fedotov et al., 2003), $D_{KL}(\pi^{t+1}_{tar}(\cdot|s)||\widetilde{\pi^t_g}(\cdot|s)) \geq \frac{1}{2\ln 2}||\pi^{t+1}_{tar}(\cdot|s) - \widetilde{\pi^t_g}(\cdot|s)||_1^2$, where $||\cdot||_1$ is the L1 norm. So we have that for all $s \in \mathcal{S}$, $||\pi^{t+1}_{tar}(\cdot|s) - \widetilde{\pi^t_g}(\cdot|s)||_1 \leq \sqrt{2\ln 2\delta}$. According to the Performance Difference Lemma (Kakade & Langford, 2002), we have that for all $s \in \mathcal{S}$:

$$
\begin{aligned}
&V_{\widetilde{\pi^t_g}}(s) - V_{\pi^{t+1}_{tar}}(s)\\
&= \frac{1}{1-\gamma}\mathbb{E}_{s'\sim\mu_s^{\widetilde{\pi^t_g}}(s')}\left[EA_{\pi^{t+1}_{tar}}(s,\widetilde{\pi^t_g})\right]\\
&\leq \frac{1}{1-\gamma}\max_{s'\in\mathcal{S}}\left[\mathbb{E}_{a\sim\widetilde{\pi^t_g}(\cdot|s')}[Q_{\pi^{t+1}_{tar}}(s',a)] - \mathbb{E}_{a\sim\pi^{t+1}_{tar}(\cdot|s')}[Q_{\pi^{t+1}_{tar}}(s',a)]\right]\\
&= \frac{1}{1-\gamma}\max_{s'\in\mathcal{S}}\int\left(\widetilde{\pi^t_g}(\cdot|s) - \pi^{t+1}_{tar}(a|s)\right)Q_{\pi^{t+1}_{tar}}(s',a)da\\
&\leq \frac{1}{1-\gamma}\max_{s'\in\mathcal{S}}\int\left|\widetilde{\pi^t_g}(a|s) - \pi^{t+1}_{tar}(a|s)\right|\cdot\left|Q_{\pi^{t+1}_{tar}}(s',a)\right|da\\
&\leq \frac{1}{1-\gamma}\max_{s'\in\mathcal{S}}\int\left|\widetilde{\pi^t_g}(a|s) - \pi^{t+1}_{tar}(a|s)\right|\cdot\frac{\widetilde{R}_{max}}{1-\gamma}da\\
&= \frac{\widetilde{R}_{max}}{(1-\gamma)^2}\max_{s'\in\mathcal{S}}||\widetilde{\pi^t_g}(\cdot|s) - \pi^{t+1}_{tar}(\cdot|s)||_1\\
&\leq \frac{\sqrt{2\ln 2\delta}\widetilde{R}_{max}}{(1-\gamma)^2}.
\end{aligned}
\tag{21}
$$

Note that $|Q_\pi(s,a)| = |\mathbb{E}_\pi[\sum_{i=0}^{\infty}\gamma^t r(s_{t+i},a_{t+i})|s_t = s, a_t = a]| \leq \mathbb{E}_\pi[\sum_{t=0}^{\infty}\gamma^t\widetilde{R}_{max}] = \frac{\widetilde{R}_{max}}{1-\gamma}$. Eventually, we have $V_{\pi^{t+1}_{tar}}(s) \geq V_{\widetilde{\pi^t_g}}(s) - \frac{\sqrt{2\ln 2\delta}\widetilde{R}_{max}}{(1-\gamma)^2} \geq V_{\pi^t_{tar}}(s) - \frac{\sqrt{2\ln 2\delta}\widetilde{R}_{max}}{(1-\gamma)^2} - \frac{2\epsilon}{1-\gamma}$.

C Discussion on the Influence of Over-Estimation

As CUP takes an argmax over expected Q values, it may suffer from the value over-estimation issue in DRL (Ostrovski et al., 2021). Although CUP may over-estimate values on rarely selected actions, this over-estimation serves as a kind of exploration mechanism, encouraging the agent to explore actions suggested by the source policies and potentially improving the learning target policy. If the source policies give unsuitable actions, then after exploration this over-estimation is resolved and these unsuitable actions will not be selected again. Results in Figure 8(b) suggest that even if all source policies are random and do not give useful actions, CUP still performs similarly to the original SAC, and is almost unaffected by the over-estimation issue, as over-estimation is addressed after exploring these actions.

D Experimental Settings

D.1 Additional Implementation Details

To improve CUP's computation efficiency, we store the source polices' output $\{\pi_1(\cdot|s), \pi_2(\cdot|s), ..., \pi_n(\cdot|s)\}$ in the replay buffer. As we can query source policies with batches of states, and each state in the buffer only need to be queried for once, CUP is computationally efficient. Empirically, CUP only takes about 30% more wall-clock time than SAC to run the same number of environment steps. All experiments are run on GeForce GTX 2080 GPUs. The policy regularization is added after 0.5M environment environment steps to achieve more stable learning.

SAC utilizes two Q functions to mitigate the overestimation error. When CUP forms the guidance policy, we use the max value of the two target Q functions to estimate the expected advantage, which contributes to bolder exploration. Using target networks contributes to more stable training.

Equation 5 requires estimating expectations over Q values. In practice, to be efficient, we estimate the expectation by sampling a few actions (e.g., 3 actions) from each action probability distribution proposed by the source policies, and find it sufficient to achieve stable performance.

As for HAAR, we fix the source policies, and train a high-level policy as well as an additional low-level policy with HAAR's auxiliary rewards.

D.2 Hyper-Parameter Details

All hyper-parameters used in our experiments are listed in Table 1. We use the same set of hyper-parameters for all six tasks. We also use the same set of hyper-parameters for both CUP and the SAC baseline. Most hyper-parameters are adopted from Sodhani et al. (2021).

D.3 Discussions on Hyper-Parameter Design

CUP has two additional hyper-parameters compared to SAC, β_1 and β_2. We provide some insight on choosing β_1 and β_2. Note that the maximum weight for the KL regularization is $\beta_1 * \beta_2 |\widetilde{V}_{\pi^t_{tar}}(s)|$, and the original actor loss L_{actor} has roughly the same magnitude as $|\widetilde{V}_{\pi^t_{tar}}(s)|$. So $\beta_1 * \beta_2$ roughly determines the maximum regularization weight. Following previous works on regularization [1,2], (0.1, 1) is a reasonable range for $\beta_1 * \beta_2$. As a consequence, we choose (0.04, 1) as the range of $\beta_1 * \beta_2$ for our hyper-parameter ablation studies. What's more, as β_2 upper bounds the maximum confidence on the expected advantage estimation (Section 3.2), β_2 should be decreased if a large variance in performance is observed. These two insights efficiently guide the design of β_1 and β_2. As shown in Section 4.3, CUP achieves stable performance on a large range of β_1s and β_2s.

E Additional Experiment Results

E.1 Success Rate Evaluation

Fig. 9 and Fig.10 shows performance evaluated by success rates. The performance is consistent with performance evaluated by cumulative return.

Table 1: Detailed hyper-parameter settings for CUP.

Hyper-Parameter	Hyper-Parameter Values
batch size	1280
non-linearity	ReLU
actor/critic network structure	fully connected networks, three fully connected layers with 400 units
policy initialization	standard Gaussian
exploration parameters	run a uniform exploration policy 50k steps
learning rates for all networks	3e-4
# of samples / # of train steps per iteration	10 env steps / 1 training step
optimizer	adam
Episode length (horizon)	500
beta for all optimizers	(0.9, 0.999)
discount	0.99
reward scale	1.0
temperature	learned
# of environment steps before adding KL regularization	500k
$beta_1$	30
β_2	3e-3

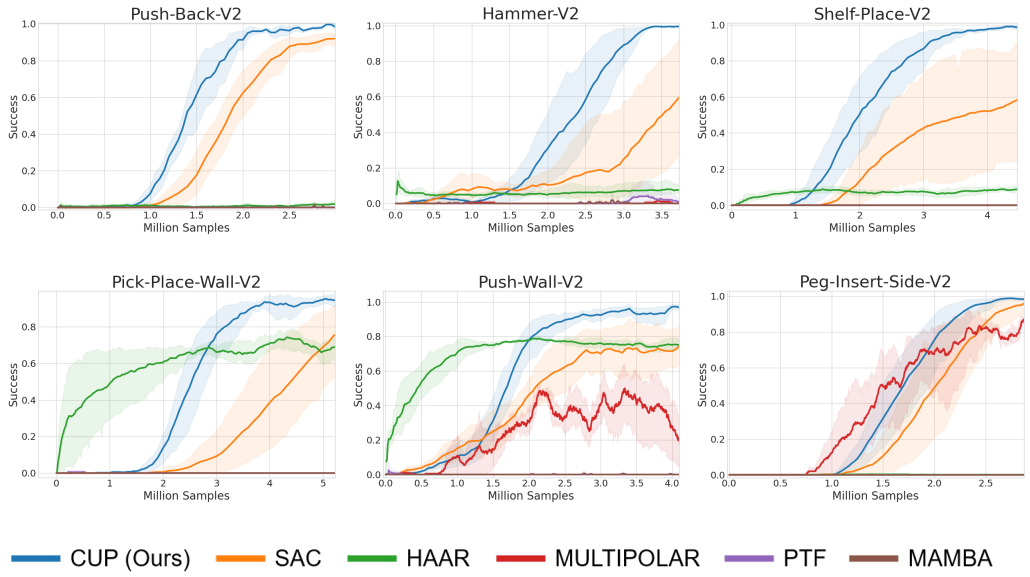


Figure 9: Algorithm performance evaluated by success rate (three source policies).

E.2 Guidance Policy Analysis for All Tasks

This subsection provides analyses of guidance policies on all six tasks, as shown in Fig. 11 and Fig. 12. Results demonstrate that the Pick-Place source policy is the most useful in Shelf-Place, Hammer, and Pick-Place-Wall, while the Push source policy is the most useful in Push-Back and Push-Wall. In Peg-Insert-Side, both source policies are of similar usefulness.

E.3 CUP's Ability to Use Additional Source Policies

In Fig. 6 the improvement brought by additional source policies is mild, because the original three source policies have already provided sufficient support for policy reuse, as demonstrated in Fig. 13. To evaluate CUP's ability to utilize additional source policies, we design another two sets of source policies. Set 1 consists of three source policies that solve Reach, Peg-Insert-Side, and Hammer, while

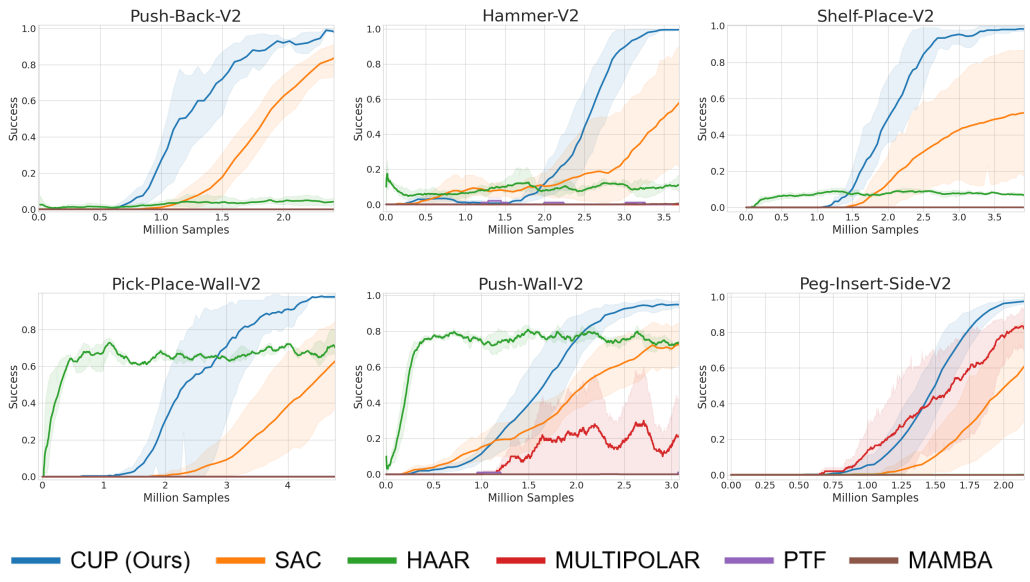


Figure 10: Algorithm performance evaluated by success rate (six source policies).

Set 2 adds source policies trained on Push-Back, Pick-Place-Wall, and Shelf-Place to Set 1. As Set 1 is less related to our target task Push-Wall, CUP must utilize the additional source policies in Set 2 to improve its performance. As demonstrated in Fig. 14, CUP can efficiently take advantage of the additional source policies to achieve efficient learning.

E.4 CUP's Source Policy Selection on the Source Task

To further investigate CUP's formation of the guidance policy, we train CUP on one of the source tasks, Push. As shown in Fig. 15, the corresponding source policy Push is selected frequently. After the target policy converges, CUP selects the target policy and the Push policy for roughly the same frequency, as they can both solve the task.

E.5 Experiments on Additional Tasks

We evaluate CUP on Bin-Picking and Stick-Pull, two tasks less related to the source policies. As demonstrated in Fig. 16, in this harder setting, CUP's performance improvement over SAC is smaller. To investigate this, we provide an analysis on CUP's source policy selection. As shown in Fig. 17, the smaller improvement is because that source policies are less related to the target tasks, as they are selected less frequently.

E.6 Analyzing Non-Stationarity in HRL Methods

To further analyze the advantages of CUP and demonstrate the non-stationarity problem of HRL methods, we illustrate the percentages of each low-level policy being selected by HAAR's high-level policy. HAAR's low-level policy set consists of the three source policies and an additional trainable low-level policy, which is expected to be selected at states where no source policies give useful actions. As demonstrated in Fig. 18(a) and Fig. 18(b), HAAR's low-level policy selection suffers from a large variance over different random seeds, and oscillates over time. This is because that as the low-level policy keeps changing, the high-level transition becomes non-stationary and leads to unstable learning. In comparison, as shown in Fig. 18(c) and Fig. 18(d), CUP's source policy selection is much more stable and achieves superior performance, as it selects source policies according to expected advantages instead of high-level policies, and avoids the non-stationarity problem.

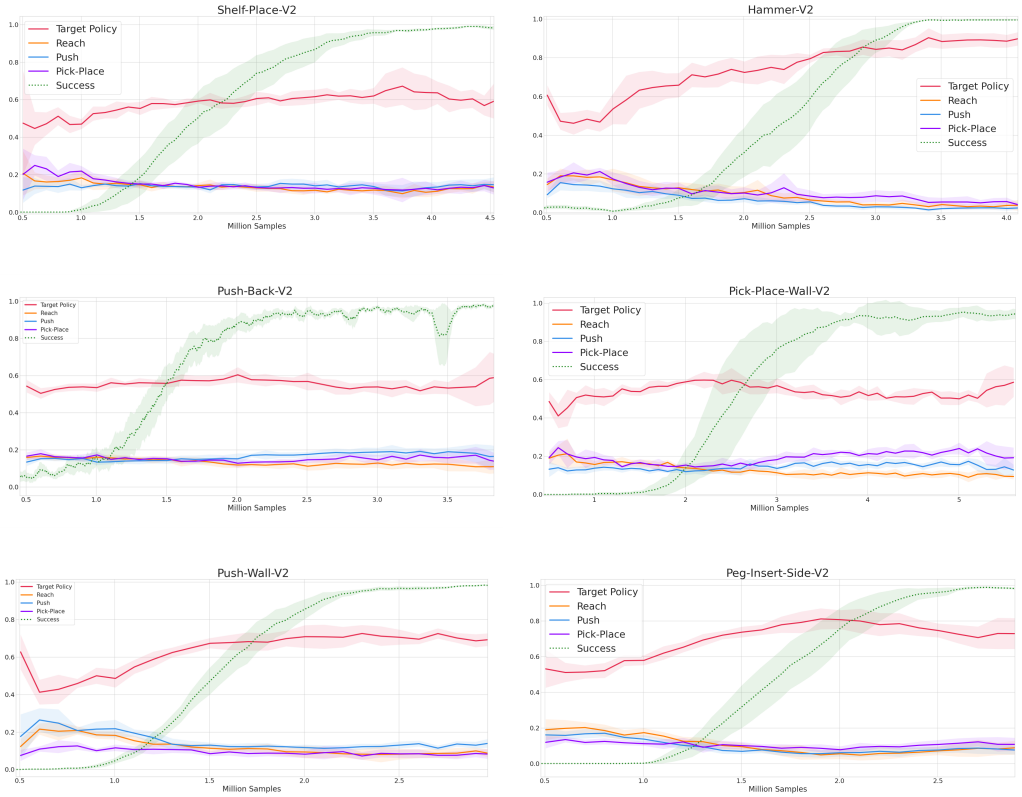


Figure 11: Percentages of source policies being selected by CUP during training on all six tasks. The green dashed line represents the target policy's success rate on the task.

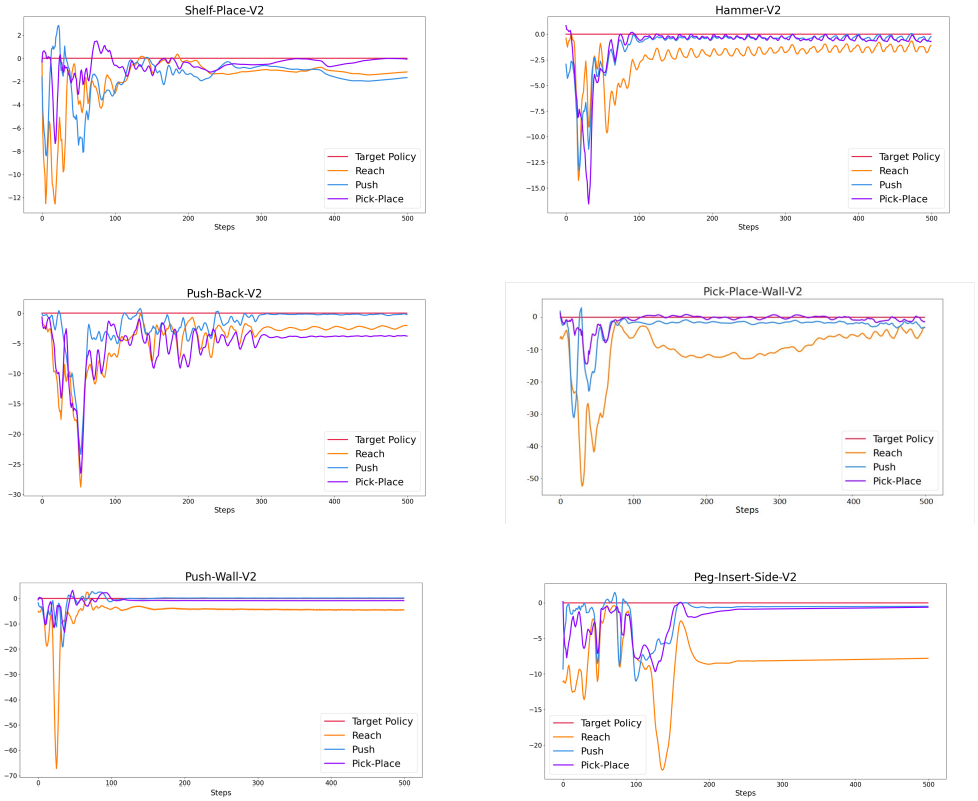


Figure 12: Expected advantages of source policies at convergence on all six tasks. The horizontal axis represents the environment steps of an episode.

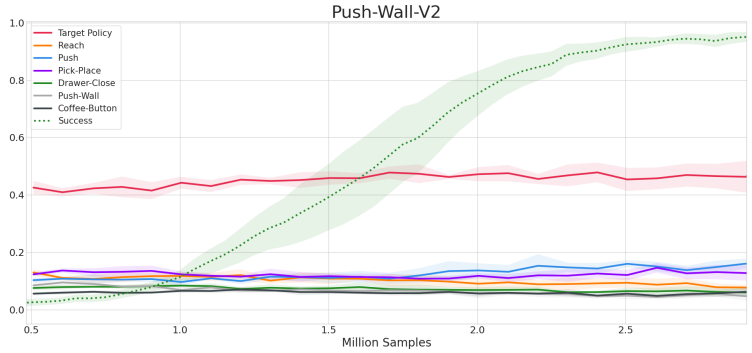


Figure 13: Percentages of source policies being selected by CUP during training on Push-Wall with the original set of six source policies. The additional source policies are seldom selected, which suggests that the first three source policies already provide sufficient support for policy reuse.

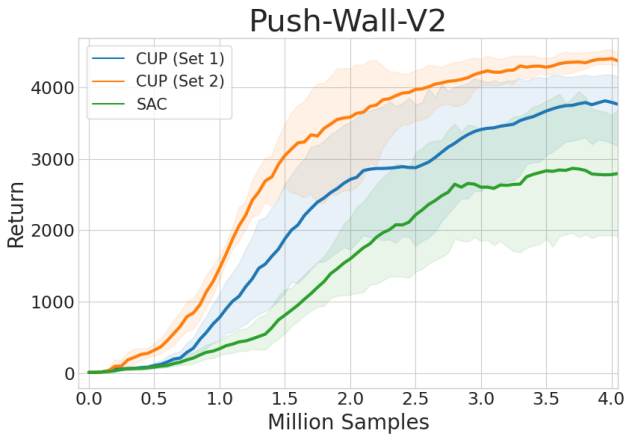


Figure 14: CUP's performance on another two sets of source policies. CUP can efficiently utilize the additional source policies contained in Set 2.

Figure 15: Percentages of source policies being selected by CUP during training on Push. The green dashed line represents the target policy's success rate on the task. The Push source policy is selected far more frequently than the other two source policies, and is selected for roughly the same frequency as the target policy at convergence.

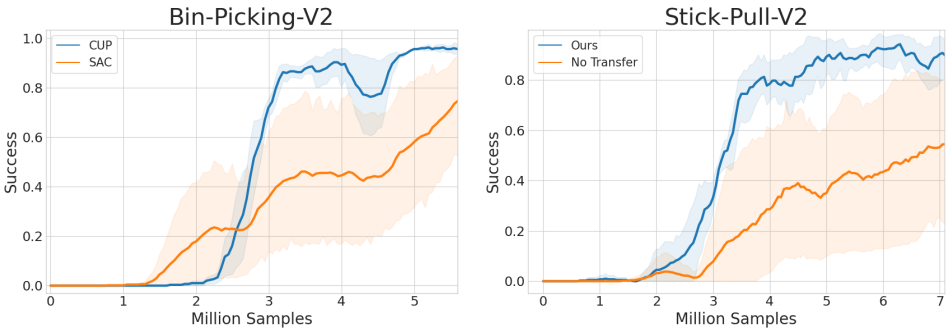


Figure 16: Performance of CUP and SAC on two harder Meta-World tasks that require more environment steps to converge.

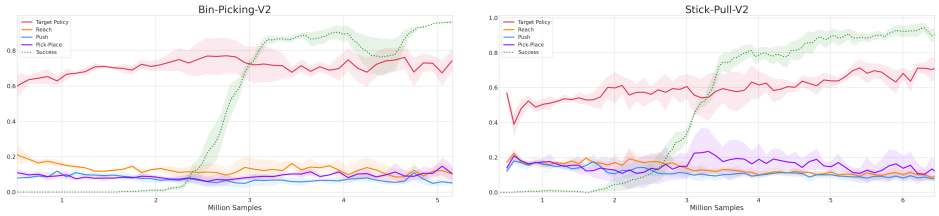


Figure 17: Percentages of source policies being selected by CUP during training. In these two tasks, the source policies are chosen less frequently, which implies that source policies are less related to these tasks.

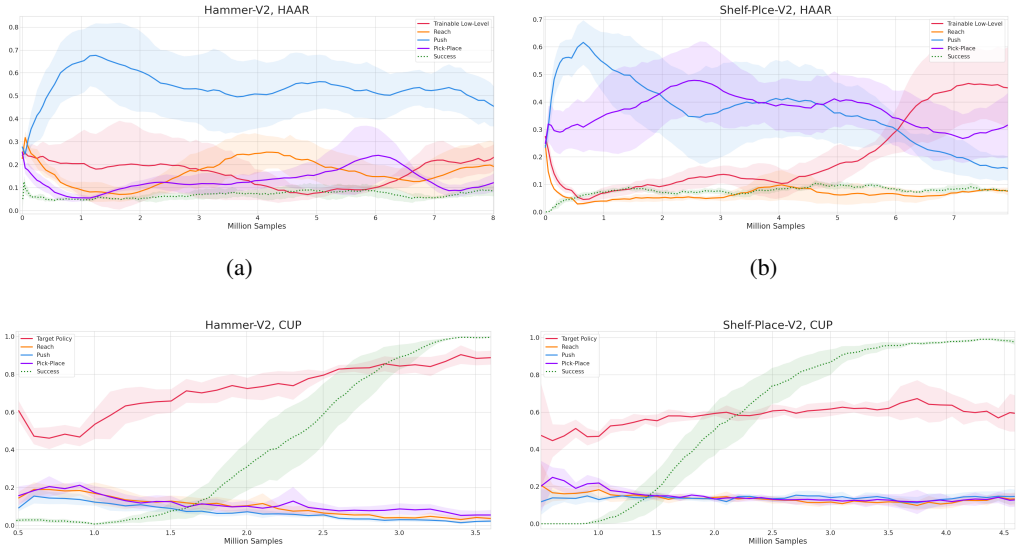


Figure 18: Comparison between HAAR and CUP's source policy selection on two representative tasks. Results are averaged over six random seeds. (a) and (b) demonstrates the percentages of each low-level policy being selected by HAAR's high-level policy. "Trainable Low-Level" is HAAR's additional trainable low-level policy, as mentioned in Appendix E.6. (c) and (d) demonstrates the the percentages of each source policy being selected by CUP. While HAAR suffers from the non-stationarity problem and has a large variance in source policy selection, CUP is much more stable and achieves superior performance, as CUP avoids the non-stationarity problem by avoiding training high-level policies.