# CDRL-BENCH: BENCHMARKING CROSS-DOMAIN TRANSFER IN REINFORCEMENT LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Cross-domain reinforcement learning (CDRL) aims to improve sample efficiency by reusing knowledge across tasks, environments, and embodiments. However, existing benchmarks largely cover specific subsets of CDRL, such as dynamics variation, while broader settings involving heterogeneous state and action spaces are evaluated less systematically and lack a unified framework. We introduce CDRL-Bench, a unified benchmark for evaluating transfer between heterogeneous source and target Markov decision processes. CDRL-Bench spans four transfer settings: same-task morphology transfer, cross-task sub-skill transfer, transfer from pretrained generalist models, and real-world transfer on physical robotic hardware. Across locomotion, navigation, and robotic manipulation, CDRL-Bench standardizes source–target pairs, interaction budgets, transfer protocols, and evaluation metrics, enabling fair and reproducible comparison across diverse transfer approaches. By broadening evaluation beyond morphology adaptation, CDRL-Bench provides a unified testbed for studying more general forms of cross-domain knowledge transfer.

## 1 INTRODUCTION

Reinforcement learning (RL) agents often require extensive interaction to learn effective behavior in a new environment, whereas humans routinely reuse prior knowledge across related settings. Bringing this capability to RL motivates *cross-domain reinforcement learning* (CDRL): transferring knowledge across tasks, environments, and embodiments to improve learning efficiency. This goal is increasingly relevant as generalist models demonstrate that reusable policies, value functions, and representations can be learned across diverse domains (Hansen et al., 2024). Such knowledge may serve as a shared high-level "brain" that is adapted through domain- or embodiment-specific modules, as similarly explored in morphological transfer with transferable high-level policies and embodiment-specific controllers (Hejna et al., 2020). Realizing this vision requires methods that bridge differences in observations, actions, dynamics, tasks, and embodiments.

CDRL studies how knowledge from one or more source domains can facilitate learning or adaptation in a target domain. We represent source and target domains as distinct Markov decision processes (MDPs) that may differ in their state and action spaces, rewards, or dynamics. This formulation encompasses transfer across tasks, environments, observation modalities, control interfaces, and robot morphologies. However, heterogeneous domains pose a fundamental challenge: when state or action spaces are incompatible, source experience, policies, and value functions cannot be directly reused without suitable cross-domain representations or correspondences. This diversity also creates an evaluation challenge, as progress is difficult to assess without common tasks, domain shifts, and evaluation protocols.

Standardized benchmarks have been central to progress in RL by providing common environments and evaluation protocols. General testbeds such as MuJoCo (Todorov et al., 2012), Atari (Bellemare et al., 2013), and the DeepMind Control Suite (Tassa et al., 2018), together with specialized benchmarks such as D4RL (Fu et al., 2020), SMAC (Samvelyan et al., 2019), Safety-Gymnasium (Ji et al., 2023), Procgen (Cobbe et al., 2019), and Meta-World (Yu et al., 2020), have enabled systematic and comparable evaluation across many RL settings.

CDRL, however, still lacks a unified evaluation framework. Existing methods are often evaluated using different tasks, source–target pairs, domain shifts, and protocols, making results difficult to compare. Existing benchmarks address subsets of CDRL, such as variations in environment param-



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

eters (Benjamins et al., 2023) or transition dynamics (Lyu et al., 2024b), but do not systematically evaluate transfer across heterogeneous state and action spaces. Beyond same-task morphology transfer, broader CDRL includes transferring sub-skills across tasks, combining knowledge from multiple sources, leveraging pretrained generalist models, and adapting across real-world embodiments. These settings remain insufficiently standardized, limiting our ability to assess what knowledge different methods transfer and how well they generalize.

To address this gap, we introduce `CDRL-Bench`, a unified benchmark for cross-domain transfer in RL. `CDRL-Bench` provides controlled transfer scenarios spanning locomotion and robotic manipulation, including same-task morphology transfer, cross-task sub-skill transfer, transfer from pretrained generalist models across tasks and embodiments, and real-world transfer on physical robotic hardware. By standardizing source–target pairs, transfer protocols, and evaluation metrics, `CDRL-Bench` provides common experimental ground for comparing task-specific transfer methods and generalist pretrained models across diverse CDRL settings.

To promote accessibility and reproducibility, `CDRL-Bench` adopts the one-file implementation style of prior benchmarks and libraries (Huang et al., 2022; Lyu et al., 2024b). Each baseline is implemented in a self-contained file, and we release environment specifications, pretrained source models, standardized evaluation protocols, and supporting resources for reproducible evaluation and future CDRL research.

We summarize our main contributions as follows: (1) We introduce `CDRL-Bench`, a unified benchmark for cross-domain RL with heterogeneous source and target MDPs, including settings with different state and action spaces. (2) We construct a diverse suite covering four forms of transfer: same-task morphology transfer, cross-task sub-skill transfer, transfer from pretrained generalist models, and real-world transfer on physical robotic hardware. (3) We provide a general benchmark formulation and standardized evaluation protocol for controlled and reproducible comparison across CDRL settings. (4) We benchmark representative CDRL methods under a unified protocol and release self-contained implementations, pretrained source models, environment specifications, and supporting resources.

## 2 Preliminaries

**Reinforcement Learning (RL).** In standard reinforcement learning, an environment is modeled as a Markov decision process (MDP), denoted by $M := (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively; $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition dynamics, where $\Delta(\cdot)$ represents the set of probability distributions over the corresponding space; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [0, 1]$ denotes the reward function; $\gamma \in [0, 1)$ is the discount factor; and $\rho \in \Delta(\mathcal{S})$ denotes the initial-state distribution.

**Cross-Domain Reinforcement Learning (CDRL).** In CDRL, we consider a set of source domains, each represented as an MDP, denoted by$M_{\text{src}} := \{M_{\text{src},i}\}_{i=1}^{N}$, together with a target domain $M_{\text{tar}}$. The objective of CDRL is to facilitate data-efficient policy learning in the target domain $M_{\text{tar}}$ by automatically exploiting transferable knowledge from the source domains $M_{\text{src}}$, thereby reducing the amount of target-domain interaction required. Notably, we make no assumptions about the structures of the source and target MDPs or their cross-domain relationships, including any predefined correspondence between their state and action spaces.

## 3 Related Work

**Same-Morphology Transfer.** Prior CDRL work often considers source and target domains with the same morphology and control interface but different environment parameters or dynamics. Domain randomization (Rajeswaran et al., 2016; Peng et al., 2018; Chebotar et al., 2019; Du et al., 2021) trains policies over environment variations for robustness. System identification and related grounding approaches (Kolev & Todorov, 2015; Yu et al., 2017; Hanna & Stone, 2017; Memmel et al., 2024) use target-domain interaction data to estimate target dynamics and reduce the dynamics mismatch between source and target domains. Off-dynamics RL (Eysenbach et al., 2021; Niu et al., 2022; Xu et al., 2023; Lyu et al., 2024a) instead reuses source-domain experience under transition dynamics shifts. These settings typically assume compatible state and action spaces.

**Cross-Morphology Transfer.** Cross-morphology transfer addresses the challenges of adapting policies across agents with varying embodiments, which often introduce mismatches in dynam-

ics, state spaces, and action spaces. To encode morphology, NerveNet (Wang et al., 2018) and TURRET (Yang et al., 2024) employ graph neural networks, with TURRET further enabling multi-source transfer via a unified state embedding. Alternatively, Hejna et al. (2020) decompose control into morphology-specific low-level controllers and a transferable high-level policy. Other approaches, such as DCC (Zhang et al., 2021), CMD (Gui et al., 2023), and ECC (Zhu et al., 2024), establish cross-domain mappings by aligning the dynamics of unpaired source and target experiences. In contrast, CAT (You et al., 2022b) and QAvatar (Chen et al., 2026) learn these mappings during target-domain adaptation by leveraging pretrained source knowledge. Specifically, CAT maximizes the mutual information between states and embeddings, whereas QAvatar minimizes cross-domain Bellman consistency to establish state-action correspondences between the source and target domains.

**Benchmark.** ODRL (Lyu et al., 2024b) benchmarks off-dynamics RL, studying how source-domain data can support learning under target-domain dynamics shifts. CARL (Benjamins et al., 2023) provides contextualized RL environments in which controllable context variables induce changes in transition dynamics and reward functions, while CausalWorld (Ahmed et al., 2021) enables controlled evaluation of robotic manipulation under configurable causal interventions. Moving beyond dynamics variations, MxT-Bench (Furuta et al., 2023) studies morphology-task generalization over procedurally generated morphology-task combinations, and Abe et al. (2026) introduce a cross-embodiment offline RL benchmark for evaluating joint policy learning under varying data quality and embodiment diversity. More general robotic learning benchmarks include Meta-World (Yu et al., 2020), which provides a suite of manipulation tasks for multi-task and meta-reinforcement learning, its standardized successor Meta-World+ (McLean et al., 2026), and RLBench (James et al., 2020), which provides large-scale robotic manipulation tasks and expert demonstrations.

## 4 Benchmark Scenarios

Building on the conventional morphology transfer setting studied in CDRL, `CDRL-Bench` progressively broadens the scope of evaluation to more challenging transfer scenarios. Our benchmark spans four settings: (1) morphology transfer evaluates knowledge transfer across agents with different morphologies while preserving the task objective; (2) sub-skill transfer considers a source task that captures only a subset of the behaviors required to accomplish the target task, requiring the target agent to identify and selectively reuse relevant source knowledge; (3) generalist model transfer draws on a large pretrained multi-task model that provides a collection of task-specific source behaviors, whose contributions are adaptively balanced to train a compact target policy for efficient deployment; (4) real-world transfer deploys reinforcement learning policies onto physical hardware. In this setting, the source and target domains share the same robotic embodiment but differ in their controllable joints, resulting in mismatched observation and action spaces.

### 4.1 Morphology transfer

Morphology transfer considers knowledge transfer between agents that share the same task objective but differ in their physical morphology. Such morphological differences can induce substantial discrepancies in state representations, action spaces, and transition dynamics. As a fundamental setting in CDRL, morphology transfer evaluates whether an agent can effectively exploit knowledge acquired from a source morphology to accelerate policy learning in a heterogeneous target domain. We adopt three categories of Morphology transfer:

**Locomotion.** Following prior work on cross-morphology locomotion (Zhang et al., 2021; Gui et al., 2023; Zhu et al., 2024; Chen et al., 2026), we consider four task families: Cheetah, Ant, Swimmer, and Centipede. For Cheetah, Ant, and Swimmer, we use the standard MuJoCo agents as the base morphologies, including 2-leg Cheetah, 4-leg Ant, and 3-link Swimmer, and construct additional variants by progressively increasing the number of legs or body links. This results in three morphologies for each family: 2-, 3-, and 4-leg Cheetah; 4-, 5-, and 6-leg Ant; and 3-, 4-, and 5-link Swimmer. For Centipede, we adopt the morphology family introduced in NerveNet (Wang et al., 2018), consisting of CentipedeFour, CentipedeSix, and CentipedeEight. Thus, each task family contains three structurally related agents with progressively increasing morphology complexity.

In this benchmark, we evaluate transfer from simpler to more complex embodiments. For example, the Cheetah family includes 2-leg $\rightarrow$ 3-leg, 3-leg $\rightarrow$ 4-leg, 2-leg $\rightarrow$ 4-leg, and 2-leg + 3-leg $\rightarrow$ 4-leg, with analogous configurations for the Swimmer, Ant, and Centipede families.

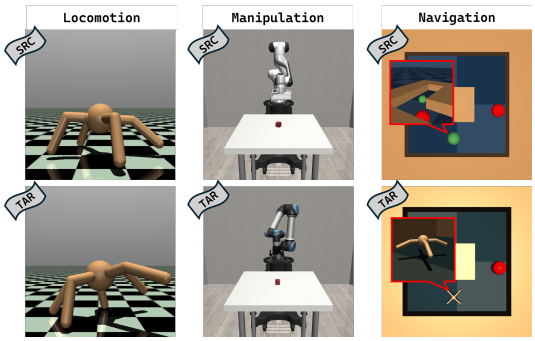



Figure 1: Morphology-transfer scenarios in CDRL-Bench. Each column shows a source domain (top) and a target domain (bottom). Locomotion transfers from simpler to more complex morphologies (e.g., 4-leg → 5-leg Ant), manipulation transfers the same task across different robot arms (e.g., Panda → UR5e), and navigation transfers across agent embodiments within the same maze (e.g., Point → Ant)

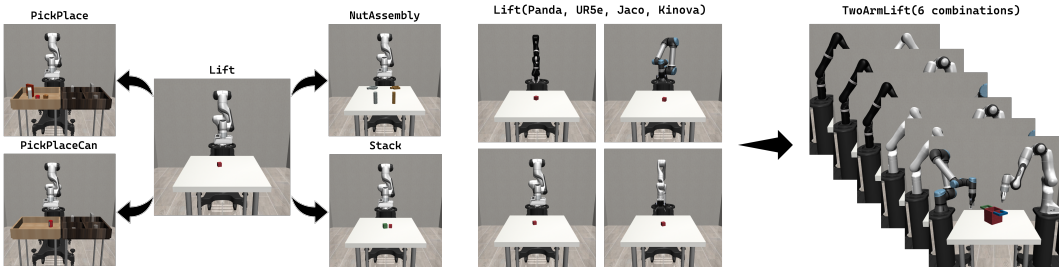
**Robot Arm Manipulation.** Robosuite (Zhu et al., 2020) provides a standardized simulation platform with diverse manipulation tasks and multiple robot-arm embodiments. Its modular design allows the same manipulation task to be configured with different robot arms while keeping the task and scene configuration fixed. This provides a controlled way to study domain shifts induced by different robot embodiments.

In this benchmark, we consider the *Block Lifting* and *Door Opening* tasks, using the Panda arm as the source embodiment and the UR5e, Jaco, and Kinova3 arms as target embodiments.

**Navigation.** Navigation tasks have been widely adopted in reinforcement learning benchmarks (Fu et al., 2020; Lyu et al., 2024b) and provide a challenging setting for goal-directed control, where an agent must navigate from an initial position to a designated goal through maze environments of varying complexity. Morphology transfer from simple point agents to more complex legged agents has also been studied in prior work (Hejna et al., 2020), demonstrating the potential of transferring navigation knowledge across substantially different embodiments. To maintain a consistent Gymnasium-based interface across our benchmark, we adopt the Minari (Younis et al., 2024) implementation of the D4RL navigation tasks. The navigation environments include four maze layouts of increasing complexity: *Open*, *UMaze*, *Medium*, and *Large*. Each layout is further provided with both sparse- and dense-reward variants, resulting in navigation tasks that vary in both environmental complexity and reward structure.

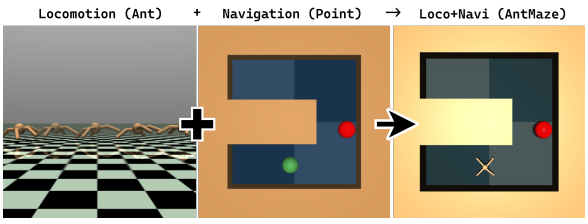
In this benchmark, we evaluate morphology transfer from the Point agent to the Ant agent across all four layouts under both sparse- and dense-reward settings.

### 4.2 Sub-skills transfer




(a) Single-Source Manipulation

(c) Heterogeneous-Embodiment Multi-Source




(b) Complementary Multi-Source

Figure 2: Overview of the three sub-skill transfer paradigms evaluated in `CDRL-Bench`. The settings scale in complexity from (a) single-source foundational extraction, to (b) complementary policy composition for hierarchical environments, and (c) morphological adaptations across diverse robotic embodiments.

As illustrated in Figure 2, complex tasks can often be decomposed into simpler behaviors, and knowledge acquired from these behaviors may remain useful when learning a more complex objective. Motivated by this observation, we consider a transfer setting in which each source task captures only part of the knowledge required to solve the target task. Unlike morphology transfer, where the source and target share the same task objective, sub-skill transfer allows their objectives to differ and requires the target agent to determine which aspects of the source knowledge are relevant and how they should be reused. This setting therefore evaluates whether an agent can identify and exploit transferable sub-skills when learning a more complex target behavior. In this benchmark, we consider three sub-skill transfer scenarios.

**Single-Source Manipulation Sub-Skills**. We transfer knowledge from `Lift` with a Panda arm to four more complex manipulation tasks: `NutAssembly`, `Stack`, `PickPlaceCan`, and `PickPlace`. The source policy provides a reusable behavior that constitutes only one component of the target task.

**Complementary Multi-Source Sub-Skill Transfer.** We uses `PointNavigate` and 4-leg Ant locomotion as two source tasks and transfers their knowledge to `AntMaze`. The two sources provide complementary capabilities: `PointNavigate` captures goal-directed navigation, while the Ant locomotion task captures embodiment-specific control. Neither source alone is sufficient to solve the target task; instead, the target agent must identify and integrate knowledge from both. This setting therefore represents complementary multi-source transfer, where distinct source tasks jointly provide the key capabilities required by the target.

**Heterogeneous-Embodiment Multi-Source Sub-Skill Transfer.** We uses `Lift` policies from Panda, UR5e, Jaco, and Kinova3 as sources for `TwoArmLift` targets composed of different pairs of robot embodiments. For each target, two source policies correspond to the embodiments present in the target domain, while the remaining sources perform the same `Lift` task with different embodiments. This setting combines multi-source transfer with embodiment heterogeneity, requiring the target agent to distinguish directly relevant source knowledge from knowledge that may be less transferable.

### 4.3 Generalist Model transfer

Large-scale multi-task models provide a rich source of reusable knowledge by capturing diverse task-specific behaviors across multiple control domains (Hansen et al., 2024). However, directly adapting such models to a new target domain is not always feasible. In particular, when the target state or action space exceeds the dimensionality supported by the pretrained model's original input or output interface, the source model cannot be directly fine-tuned without modifying its architecture. Moreover, deploying the full generalist model may be unnecessarily expensive when only a subset of its knowledge is relevant to the target task. These limitations motivate transferring useful knowledge from a pretrained generalist model into a compact target policy.

In this benchmark, we use the publicly released 317M-parameter TD-MPC2 model (Hansen et al., 2024), pretrained jointly on 30 tasks from the DeepMind Control Suite (Tassa et al., 2018), as the generalist source model. We treat the behaviors associated with its pretrained tasks as candidate sources of task-specific knowledge. As target domains, we consider four locomotion environments: 4-limb Swimmer, 5-leg Ant, and CentipedeSix from Section 4.1, together with the standard MuJoCo Humanoid-v5 environment. For each target, we train a compact SAC policy that selectively exploits and adaptively balances source knowledge rather than directly fine-tuning the generalist model. This setting evaluates whether knowledge from a large multi-task model can be transferred to a compact policy when the target domain may have state or action spaces beyond those supported by the pretrained source model.

### 4.4 Real2Real Experiments

Real-world transfer extends cross-domain reinforcement learning to physical hardware, where sensing noise, system latency, imperfect actuation, noisy rewards, and costly target-domain interaction make robust and sample-efficient adaptation particularly important. This setting evaluates whether existing methods can effectively leverage source-domain knowledge under these practical constraints.

As show in Figure 3, we use the SO-ARM101, a low-cost 3D-printed robotic arm driven through LeRobott (Cadene et al., 2024), for a visual reaching task. The agent controls the arm to move its wrist-mounted camera toward a visual target displayed on a screen, with rewards computed from

| Task/Alg | QAvatar-SAC | CAT-SAC | FT-SAC | SAC | CAT-PPO | FT-PPO | PPO |
|---|---|---|---|---|---|---|---|
| C+S | $0.39 \pm 0.13$ | $0.34 \pm 0.14$ | $0.37 \pm 0.11$ | $0.34 \pm 0.12$ | $0.54 \pm 0.25$ | $0.57 \pm 0.27$ | $0.54 \pm 0.27$ |
| A+C | $0.52 \pm 0.13$ | $0.63 \pm 0.09$ | $0.70 \pm 0.07$ | $0.59 \pm 0.07$ | $0.74 \pm 0.06$ | $0.83 \pm 0.04$ | $0.82 \pm 0.09$ |
| D+L | $0.63 \pm 0.18$ | $0.69 \pm 0.17$ | $0.67 \pm 0.14$ | $0.62 \pm 0.19$ | $0.77 \pm 0.08$ | $1.00 \pm 0.01$ | $0.79 \pm 0.11$ |
| Robo SS | $0.89 \pm 0.06$ | $0.95 \pm 0.03$ | $0.96 \pm 0.03$ | $0.95 \pm 0.03$ | $0.98 \pm 0.03$ | $1.00 \pm 0.00$ | $0.99 \pm 0.02$ |
| Maze:M+S | $1.00 \pm 0.01$ | $1.00 \pm 0.01$ | $1.00 \pm 0.01$ | $1.00 \pm 0.01$ | $0.99 \pm 0.01$ | $0.99 \pm 0.01$ | $0.99 \pm 0.01$ |
| Real2Real | $0.78 \pm 0.02$ | $0.84 \pm 0.02$ | $0.86 \pm 0.08$ | $0.70 \pm 0.20$ | – | – | – |

Table 1: Regret computed over expert-normalized learning curves. Lower is better. Forabbreviations, C+S = Cheetah+Swimmer, A+C = Ant+Centipede, D+L = Door+Lift, Robo SS = Robotic Sub-skills, Maze:M+S = Maze Morphology+ Maze Sub-skills, Real2Real = real-world transfer.

the target position in the camera image. To keep the real-world setup tractable, the source and target domains share the same robot, task, and reward function, but differ in their control interfaces. The source controls four joints, whereas the target fixes the wrist-flex joint at its home position, inducing different state and action spaces without providing explicit cross-domain correspondence. We evaluate four SAC-based methods, namely SAC, FT-SAC, CAT-SAC, and QAvatar-SAC, motivated by SAC's stronger performance than PPO in our simulated benchmark. Detailed settings are provided in Appendix D.

### 4.5 Evaluation Protocol

In `CDRL-Bench`, we first pretrain a high-quality source model for each environment in the source-domain set $M_{\text{src}}$. The pretrained source models are frozen before target-domain training and are shared across all applicable transfer methods. Source-model pretraining is performed once and is not counted toward the target-domain interaction budget. All methods are evaluated using the same target-domain online-interaction budget.

We report two complementary evaluation metrics. First, the expert-normalized score (NS) is defined as $\text{NS} = \frac{J_\pi - J_{\text{rand}}}{J_{\text{expert}} - J_{\text{rand}}}$, where $J_\pi$ denotes the return of the learned target policy, and $J_{\text{rand}}$ and $J_{\text{expert}}$ denote the returns of a random policy and an expert policy in the target domain, respectively. Second, we report Average Normalized Regret, $\bar{R} = \frac{1}{K}\sum_{k=1}^{K}(1 - S_k)$, where $S_k$ is the expert-normalized score at the $k$-th evaluation checkpoint and $K$ is the total number of evaluation checkpoints.



Figure 3: A real-robot domain: an SO-ARM101 arm reaching for a green dot on a screen, scored by what its wrist camera sees.

## 5 Experimental Results

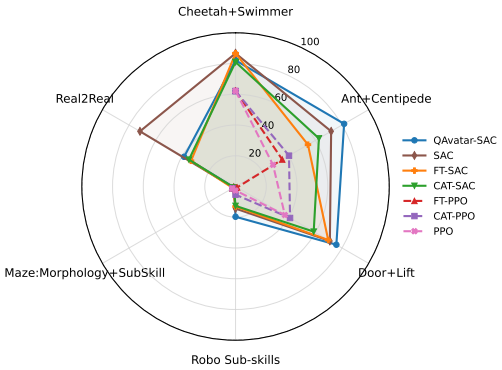



Figure 4: Expert-normalized best performance across `CDRL-Bench` benchmark groups.

In this section, we present the overall benchmark results and systematically analyze the empirical findings through a series of research questions. Our goal is twofold. First, we examine whether the conclusions established by prior CDRL studies continue to hold under a broader, more diverse, and unified evaluation protocol. Second, we investigate whether existing methods can deliver reliable and consistent transfer in more challenging scenarios that go beyond the morphology-transfer settings predominantly considered in prior work. Through this analysis, we aim to provide a more comprehensive assessment of the current capabilities and limitations of CDRL methods, while also highlighting the potential of CDRL as a more general framework for enabling effective knowledge transfer across heterogeneous tasks.

We first introduce the baseline algorithms used for comparison, and then present and analyze the benchmark results with respect to our research questions.

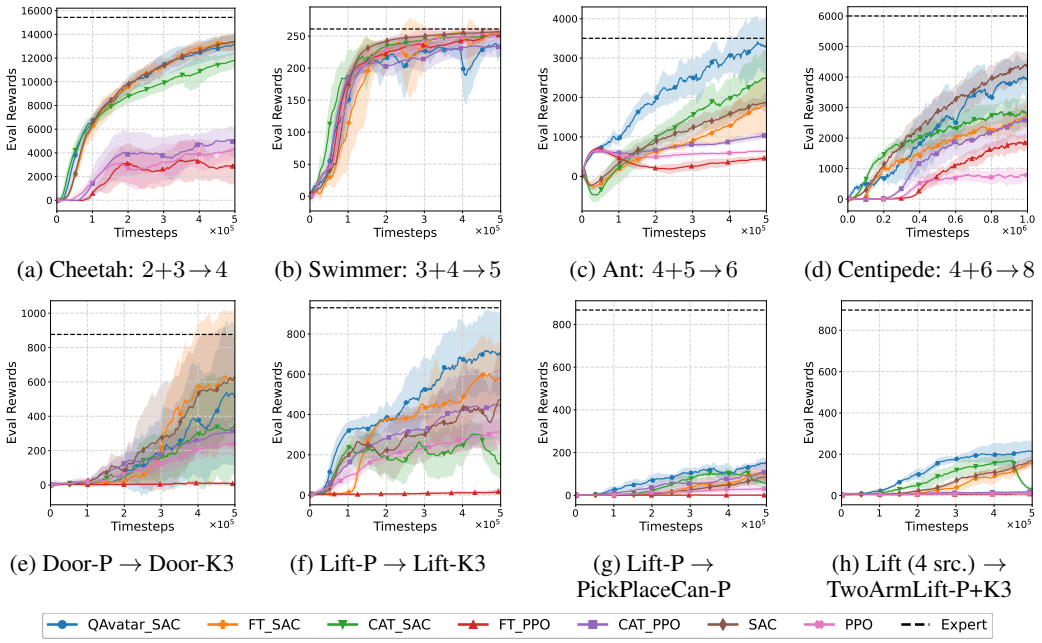



Figure 5: Learning curves across representative `CDRL-Bench` transfer settings. (a)–(f) evaluate morphology transfer across locomotion and robotic manipulation tasks, while (g)–(h) evaluate sub-skill transfer, including single-source and heterogeneous-embodiment multi-source transfer. For abbreviations, P = Panda and K3 = Kinova3.

### 5.1 Baseline Algorithms

For our baseline evaluation, we exclude methods that require additional structural information or task-specific structural assumptions, such as known morphology graphs or predefined hierarchical decompositions (Wang et al., 2018; Hejna et al., 2020; Yang et al., 2024), as these requirements make such methods difficult to extend to more general transfer settings.

We consider seven baselines in three categories: **(1) Learning from scratch:** The target policy is randomly initialized and trained only with target-domain interactions, serving as a non-transfer reference. We use **PPO** (Schulman et al., 2017) and **SAC** (Haarnoja et al., 2018). **(2) Direct fine tuning:** The target policy is initialized from source-domain parameters and fine-tuned on the target domain, a common baseline in prior cross-domain transfer studies (Wang et al., 2018; Wan et al., 2020; Chen et al., 2026). We denote the PPO- and SAC-based variants as **FT-PPO** and **FT-SAC**. **(3) Cross-domain transfer:** These methods explicitly learn relationships between heterogeneous source and target domains. **CAT-PPO** (You et al., 2022a) and its SAC-based adaptation, **CAT-SAC**, used as a baseline by Chen et al. (Chen et al., 2026), learn state–action correspondences using mutual-information and cycle-consistency objectives, while **QAvatar-SAC** (Chen et al., 2026) learns inter-domain mappings through cross-domain Bellman consistency and uses a hybrid critic to adaptively combine source- and target-domain Q-functions.

### 5.2 Results and Analysis

All experiments use five random seeds, with results aggregated across seeds. Experimental settings, implementation details, and hyperparameters are provided in Appendix B. Results for the settings introduced in Sections 4.1, 4.2, and 4.4 are summarized in Figure 4 and Table 1, with representative learning curves shown in Figure 5. Full learning curves are reported in Appendix C, and complete average-regret results in Table 6.

Based on these results, we examine whether prior CDRL conclusions remain robust under a broader unified protocol and how reliably existing methods transfer knowledge across diverse and challenging settings, revealing both their strengths and limitations.

**RQ1: Do conclusions from prior CDRL evaluations hold under a broader, unified evaluation?**

Prior CDRL studies evaluate different tasks, source-target pairs, and source-policy qualities, making it difficult to assess whether their conclusions generalize. `CDRL-Bench` addresses this by evaluating representative methods under a unified protocol against their corresponding non-transfer baselines. As summarized in Figure 4 and Table 1, the benefits of dedicated transfer methods remain highly setting-dependent. CAT-PPO consistently matches or outperforms PPO across benchmark groups, while QAvatar-SAC improves over SAC in most, but not all, settings. Thus, dedicated CDRL mechanisms can provide substantial gains, but these gains do not generalize uniformly across transfer scenarios.

Direct fine-tuning, however, shows behavior consistent with prior evaluations. FT-SAC provides only limited improvements over SAC, while FT-PPO can underperform PPO; under our broader evaluation, fine-tuning likewise generally matches or underperforms the corresponding scratch baseline. Overall, our results reinforce that direct fine-tuning offers limited and inconsistent benefits, while the effectiveness of dedicated CDRL methods also depends strongly on the evaluation setting.

**RQ2: What does Average Normalized regret reveal beyond best performance?**

CDRL methods are commonly evaluated using time-to-threshold and best performance. However, time-to-threshold depends on a manually specified threshold, and varying this threshold across studies can lead to different conclusions, making cross-study comparisons less consistent. best performance, in contrast, captures only the strongest policy obtained within a fixed interaction budget and does not reflect performance throughout training. We therefore also report average normalized regret in Table 1, which measures the average gap to expert performance over the full target-domain interaction budget. By accounting for the entire learning trajectory, regret captures early, intermediate, and late-stage performance in a single normalized measure and can therefore yield different orderings from best performance. For example, among SAC-based methods, CAT-SAC shows relatively low best performance on Cheetah and Swimmer in Figure 4, yet achieves the lowest average normalized regret. This indicates that CAT-SAC remains closer to expert performance for a larger portion of training, even without attaining the strongest peak performance. Thus, source knowledge may improve the overall learning process while still limiting the best performance eventually reached. For CDRL, transfer should therefore be evaluated not only by peak policy quality, but also by learning performance throughout target-domain training.

**RQ3: How well do existing methods solve morphology transfer?**

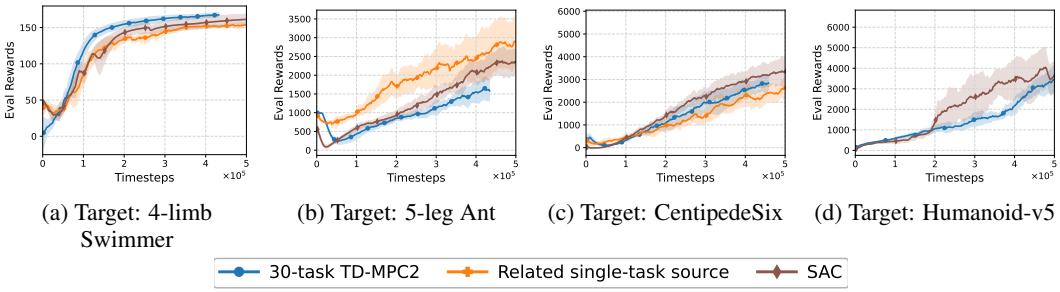
Morphology transfer has been a primary focus of prior CDRL research, yet our results in Figure 4 show that its effectiveness remains highly setting-dependent. When source and target morphologies are relatively similar, several methods successfully reuse source knowledge and achieve competitive performance, but no single method consistently dominates across morphology families. On the easier Cheetah and Swimmer tasks, most SAC-based methods eventually achieve strong performance. Figure 18 further shows this target dependence on the Door task, where changing the target robot alters which transfer strategy is most effective despite using the same source embodiment and task. This challenge becomes more pronounced in Point-to-Ant navigation in Figure 19, where all methods struggle to learn transferable knowledge and make little to no progress across maze layouts and reward settings. Overall, morphology transfer remains sensitive to embodiment changes, highlighting the need to evaluate methods across diverse source-target pairs.

**RQ4: How effectively can existing CDRL methods identify and reuse relevant knowledge across tasks?**

Beyond same-task morphology transfer, methods must identify source knowledge relevant to different target objectives. Our sub-skill experiments, reported as Robo Sub-skills in both Figure 4 and Table 1, show that cross-task reuse is possible, but remains difficult even without an embodiment gap, with all methods far from expert performance. In the multi-source TwoArmLift setting, Figure 21 shows that QAvatar-SAC consistently performs best across target embodiment pairs, demonstrating benefits from multiple heterogeneous source policies. In contrast, AntMaze in Figure 22 remains challenging: although PointNavigate and Ant locomotion provide complementary capabilities required by the target task, existing methods still make little meaningful progress, highlighting the difficulty of integrating complementary skills.

For generalist model transfer, the benchmark further requires the target policy to be substantially smaller than the source model. Under this constraint, QAvatar-SAC is the only applicable evaluated

method because it does not require reusing source feature representations, whereas FT and CAT require architectural compatibility. Figure 6 further shows that broader source knowledge does not consistently improve transfer. The 30-task TD-MPC2 source performs best on 4-limb Swimmer, while the more closely related 4-leg Ant source performs substantially better on 5-leg Ant, and SAC from scratch outperforms transfer methods on Humanoid and CentipedeSix. These results suggest that source relevance can matter more than source diversity, and that current transfer mechanisms do not always suppress harmful source influence.




(a) Target: 4-limb Swimmer (b) Target: 5-leg Ant (c) Target: CentipedeSix (d) Target: Humanoid-v5

Figure 6: Learning curves for generalist-model transfer across four target domains. In the legend, "30-task TD-MPC2" denotes transfer from the pretrained generalist model, "Morphology transfer" denotes the corresponding morphology-transfer setting, and "SAC" denotes training from scratch.

**RQ5: Can existing CDRL methods effectively transfer between real-world domains?**

Real-world transfer is more challenging than the simulated settings above because abundant source-domain data cannot be assumed. As shown in Figure 7 and the Real2Real results in Table 1, none of the evaluated transfer methods outperforms SAC trained from scratch. QAvatar-SAC and CAT-SAC are more consistent across seeds but plateau at low performance, whereas SAC from scratch reaches substantially higher peaks despite greater variability, indicating limited benefit from current CDRL methods in the Real2Real setting. A key limitation is the restricted coverage of source-domain data. Both QAvatar-SAC and CAT-SAC use normalizing flows to constrain cross-domain mappings to feasible source state-action regions, but limited source data restricts the coverage learned by these models and, consequently, where target samples can be meaningfully mapped. Moreover, costly real-world interaction imposes a smaller target-domain budget than in sim2sim experiments, making sample efficiency especially important. Within this limited budget, none of the evaluated transfer methods gains a clear advantage over learning from scratch. Future work should therefore explore transfer mechanisms that rely less on additional source-domain data while remaining sample-efficient under limited target interaction.

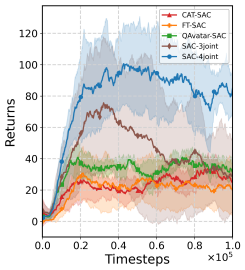



Figure 7: Training curves on the Real2Real experiment.

## 6 Conclusion

We introduce `CDRL-Bench`, a unified benchmark for cross-domain reinforcement learning with heterogeneous source and target domains. `CDRL-Bench` provides a standardized evaluation protocol spanning conventional morphology transfer, cross-task sub-skill transfer, generalist model transfer, and Real2Real transfer. Across these diverse settings, our results show that transfer effectiveness remains highly dependent on the source-target relationship. These findings also demonstrate the potential of reusing sub-skills across tasks, while highlighting the challenge of reliably identifying and integrating relevant source knowledge. Our Real2Real experiments further show that existing CDRL methods provide limited transfer benefits in the presence of real-world factors, underscoring the need for more robust transfer mechanisms. A limitation of `CDRL-Bench` is that it considers mainly source models of medium to high quality. Evaluating robustness to source models of varying quality remains an important direction for future work.

### AI USE STATEMENT

During the preparation of this work, we used generative AI tools to assist in literature discovery and linguistic editing of the manuscript. We independently verified all generated suggestions and confirmed the accuracy of all cited references. The AI tools were not involved in the study design, methodology, experimental execution, or interpretation of the results.

### ETHICS STATEMENT

All experiments in this study, both simulated and real-world, were conducted without human participants or sensitive data. Real-world robotic operations were executed autonomously via pre-programmed scripts, ensuring no human-robot interaction occurred during the trials. Consequently, this research fully complies with standard ethical guidelines.

### REPRODUCIBILITY STATEMENT

The code for our experiments is provided in the supplementary material, accompanied by README files detailing execution instructions. A complete list of dependencies is also included to ensure straightforward reproducibility in the simulated environment.

## REFERENCES

Haruki Abe, Takayuki Osa, Yusuke Mukuta, and Tatsuya Harada. Cross-embodiment offline reinforcement learning for heterogeneous robot datasets. In *International Conference on Learning Representations*, 2026.

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Manuel Wuthrich, Yoshua Bengio, Bernhard Schölkopf, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. In *International Conference on Learning Representations*, 2021.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 2013.

Carolin Benjamins, Theresa Eimer, Frederik Schubert, Aditya Mohan, Sebastian Döhler, André Biedenkapp, Bodo Rosenhahn, Frank Hutter, and Marius Lindauer. Contextualize me – the case for context in reinforcement learning. *Transactions on Machine Learning Research*, 2023.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Khalil Meftah, Maxime Ellerbach, Jess Moss, and Thomas Wolf. LeRobot: State-of-the-Art Machine Learning for Real-World Robotics in Pytorch. https://github.com/huggingface/lerobot, 2024.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *IEEE International Conference on Robotics and Automation*, 2019.

Ming-Hong Chen, Kuan-Chen Pan, You-De Huang, Xi Liu, and Ping-Chun Hsieh. Cross-domain policy optimization via bellman consistency and hybrid critics. In *The Fourteenth International Conference on Learning Representations*, 2026.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. In *IEEE International Conference on Robotics and Automation*, 2021.

Benjamin Eysenbach, Shreyas Chaudhari, Swapnil Asawa, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. In *International Conference on Learning Representations*, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Hiroki Furuta, Yusuke Iwasawa, Yutaka Matsuo, and Shixiang Shane Gu. A system for morphology-task generalization via unified representation and behavior distillation. In *International Conference on Learning Representations*, 2023.

Haiyuan Gui, Shanchen Pang, Shihang Yu, Sibo Qiao, Yufeng Qi, Xiao He, Min Wang, and Xue Zhai. Cross-domain policy adaptation with dynamics alignment. *Neural Networks*, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Josiah Hanna and Peter Stone. Grounded action transformation for robot learning in simulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

Nick Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

Donald Hejna, Lerrel Pinto, and Pieter Abbeel. Hierarchically decoupled imitation for morphological transfer. In *International Conference on Machine Learning*, 2020.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃĢo GM AraÃšjo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 2022.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. Safety gymnasium: A unified safe reinforcement learning benchmark. In *onference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Svetoslav Kolev and Emanuel Todorov. Physically consistent state estimation and system identification for contacts. In *International Conference on Humanoid Robots*, 2015.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Jiafei Lyu, Chenjia Bai, Jing-Wen Yang, Zongqing Lu, and Xiu Li. Cross-domain policy adaptation by capturing representation mismatch. In *International Conference on Machine Learning*, 2024a.

Jiafei Lyu, Kang Xu, Jiacheng Xu, Mengbei Yan, Jingwen Yang, Zongzhang Zhang, Chenjia Bai, Zongqing Lu, and Xiu Li. Odrl: A benchmark for off-dynamics reinforcement learning. In *Advances in Neural Information Processing Systems*, 2024b.

Reginald McLean, Evangelos Chatzaroulas, Luc McCutcheon, Frank Röder, Tianhe Yu, Zhanpeng He, KR Zentner, Ryan Julian, Jordan Terry, Isaac Woungang, et al. Meta-world+: An improved, standardized, rl benchmark. *Advances in Neural Information Processing Systems*, 2026.

Marius Memmel, Andrew Wagenmaker, Chuning Zhu, Dieter Fox, and Abhishek Gupta. Asid: Active exploration for system identification in robotic manipulation. In *International Conference on Learning Representations*, 2024.

Haoyi Niu, shubham sharma, Yiwen Qiu, Ming Li, Guyue Zhou, Jianming HU, and Xianyuan Zhan. When to trust your simulator: Dynamics-aware hybrid offline-and-online reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation*, 2018.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. In *International Conference on Learning Representations*, 2016.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International conference on intelligent robots and systems*, 2012.

Michael Wan, Tanmay Gangwani, and Jian Peng. Mutual information based knowledge transfer under state-action dimension mismatch. In *Conference on Uncertainty in Artificial Intelligence*, 2020.

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

Kang Xu, Chenjia Bai, Xiaoteng Ma, Dong Wang, Bin Zhao, Zhen Wang, Xuelong Li, and Wei Li. Cross-domain policy adaptation via value-guided data filtering. *Advances in Neural Information Processing Systems*, 2023.

Tianpei Yang, Heng You, Jianye Hao, Yan Zheng, and Matthew E Taylor. A transfer approach using graph neural networks in deep reinforcement learning. In *Association for the Advancement of Artificial Intelligence*, 2024.

Heng You, Tianpei Yang, Yan Zheng, Jianye Hao, and E. Taylor, Matthew. Cross-domain adaptive transfer reinforcement learning based on state-action correspondence. In *Conference on Uncertainty in Artificial Intelligence*, 2022a.

Heng You, Tianpei Yang, Yan Zheng, Jianye Hao, and E. Taylor, Matthew. Cross-domain adaptive transfer reinforcement learning based on state-action correspondence. In *Uncertainty in Artificial Intelligence*, 2022b.

Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. minari, 2024.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, 2020.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems*, 2017.

Yufeng Yuan and A. Rupam Mahmood. Asynchronous Reinforcement Learning for Real-Time Control of Physical Robots. In *IEEE International Conference on Robotics and Automation*, pp. 5546–5552, 2022.

Qiang Zhang, Tete Xiao, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Learning cross-domain correspondence for control with dynamics cycle-consistency. In *International Conference on Learning Representations*, 2021.

Ruiqi Zhu, Tianhong Dai, and Oya Celiktutan. Cross domain policy transfer with effect cycle-consistency. In *IEEE International Conference on Robotics and Automation*, 2024.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. Robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

# APPENDICES

## A SIMULATED BENCHMARK ENVIRONMENTS

CDRL-Bench comprises four transfer settings: morphology transfer, sub-skill transfer, generalist model transfer, and real-world transfer. Together, these settings cover transfer across embodiments and tasks, transfer from multiple source domains, transfer from pretrained multi-task models, and transfer between domains with heterogeneous state and action spaces.

### A.1 MORPHOLOGY TRANSFER

#### LOCOMOTION

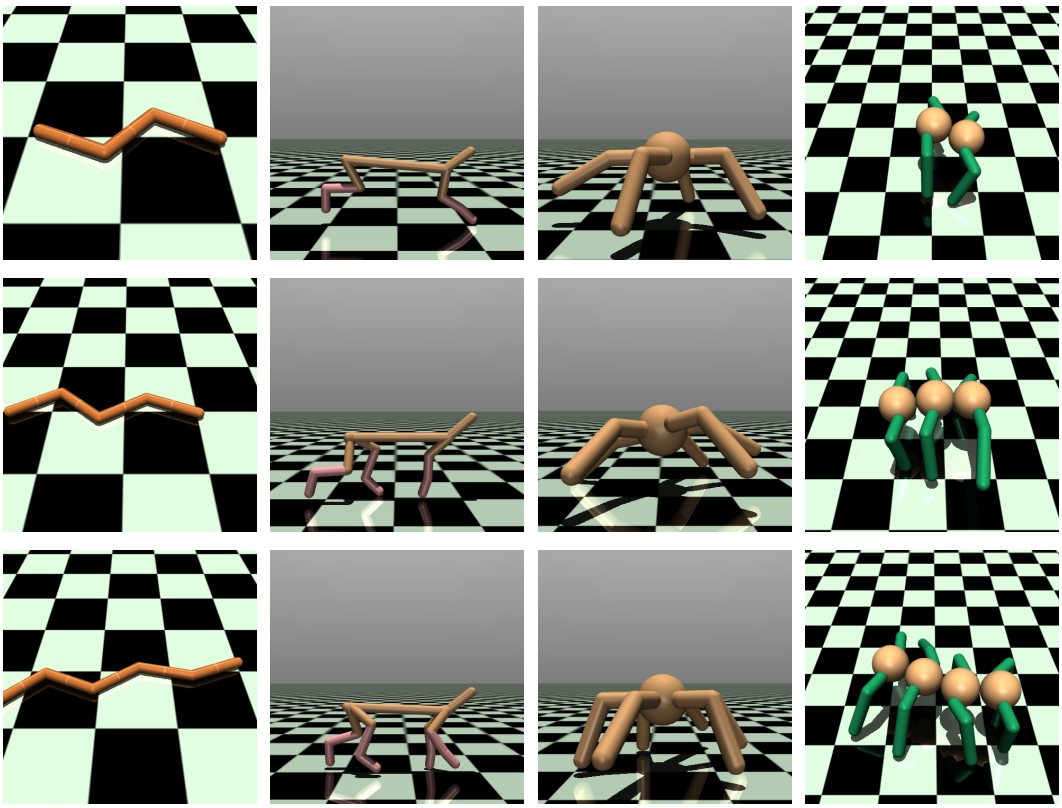


Figure 8: **The Locomotion Playground:** Renderings of the four morphological families evaluated. The agents are organized by column from left to right: ***Swimmer*** (3, 4, and 5 limbs), ***Cheetah*** (2, 3, and 4 legs), ***Ant*** (4, 5, and 6 legs), and ***Centipede*** (4, 6, and 8 legs). Within each column, the structural complexity of the agents increases from top to bottom.

**Swimmer Family.** The Swimmer environments consist of {3, 4, 5}-limb configurations based on OpenAI Gymnasium's Swimmer-v5. The standard 3-limb model serves as the unmodified baseline. To construct the 4-limb and 5-limb variants, we attach one and two additional limbs to the articulated chain, respectively. All supplementary limbs retain the exact physical specifications—including mass, segment length, and fluid friction constraints—of the baseline limbs to ensure physical consistency across morphologies. Below is the main extension in the corresponding xml file:

- *4limb_swimmer.xml*: Extends the 3-link Swimmer with an additional posterior limb (`new`) actuated by a Z-axis hinge joint (`motor3_rot`).

```
<body name="new" pos="-1 0 0">
```

```
  <geom density="1000" fromto="0 0 0 -1 0 0" size="0.1" type="
      capsule"/>
  <joint axis="0 0 1" limited="true" name="motor3_rot" pos="0 0 0
      " range="-100 100" type="hinge"/>
</body>
```

- *5limb_swimmer.xml*: Appends a fifth limb (`new1`) and hinge joint (`motor4_rot`) to the 4-link model.

```
<body name="new1" pos="-1 0 0">
  <geom density="1000" fromto="0 0 0 -1 0 0" size="0.1" type="
      capsule"/>
  <joint axis="0 0 1" limited="true" name="motor4_rot" pos="0 0 0
      " range="-100 100" type="hinge"/>
</body>
```

**Cheetah Family.** The Cheetah environments consist of {2, 3, 4}-legged variants derived from OpenAI Gymnasium's HalfCheetah-v5. The standard 2-legged model is used without modification. We construct the 3-legged variant by adding a duplicate hind leg, and the 4-legged variant by additionally adding a duplicate front leg. All added limbs preserve the physical properties of their corresponding baseline limbs, including mass and friction coefficients, to maintain consistency across morphologies. Below is the main extension in the corresponding xml file:

- *3leg_cheetah.xml*: Extends the bipedal HalfCheetah with a laterally offset hind leg (`bthigh2`, `bshin2`, `bfoot2`).

```
<body name="bthigh2" pos="-.5 .06 0">
  <joint axis="0 1 0" damping="6" name="bthigh2" pos="0 0 0"
      range="-.52 1.05" stiffness="240" type="hinge"/>
  <geom axisangle="1 -1 0 3.8" name="bthigh2" pos=".1 0 -.13"
      size="0.046 .145" type="capsule"/>
  <body name="bshin2" pos=".16 .06 -.25">
    <joint axis="0 1 0" damping="4.5" name="bshin2" pos="0 0 0"
        range="-.785 .785" stiffness="180" type="hinge"/>
    <geom axisangle="0 1 0 -2.03" name="bshin2" pos="-.14 0 -.07"
        rgba="0.9 0.6 0.6 1" size="0.046 .15" type="capsule"/>
    <body name="bfoot2" pos="-.28 0 -.14">
      <joint axis="0 1 0" damping="3" name="bfoot2" pos="0 0 0"
          range="-.4 .785" stiffness="120" type="hinge"/>
      <geom axisangle="0 1 0 -.27" name="bfoot2" pos=".03 0 -.097"
          rgba="0.9 0.6 0.6 1" size="0.046 .094" type="capsule"/>
    </body>
  </body>
</body>
```

- *4leg_cheetah.xml*: Adds a front leg (`fthigh2`, `fshin2`, `ffoot2`) to the 3-legged model.

```
<body name="fthigh2" pos=".5 -.06 0">
  <joint axis="0 1 0" damping="4.5" name="fthigh2" pos="0 0 0"
      range="-1 .7" stiffness="180" type="hinge"/>
  <geom axisangle="1 -1 0 -.52" name="fthigh2" pos="-.07 0 -.12"
      size="0.046 .133" type="capsule"/>
  <body name="fshin2" pos="-.12 -.05 -.24">
    <joint axis="0 1 0" damping="3" name="fshin2" pos="0 0 0"
        range="-1.2 .87" stiffness="120" type="hinge"/>
    <geom axisangle="0 -1 0 .6" name="fshin2" pos=".065 0 -.09"
        rgba="0.9 0.6 0.6 1" size="0.046 .106" type="capsule"/>
    <body name="ffoot2" pos=".13 0 -.18">
      <joint axis="0 1 0" damping="1.5" name="ffoot2" pos="0 0 0"
          range="-.5 .5" stiffness="60" type="hinge"/>
      <geom axisangle="0 1 0 -.6" name="ffoot2" pos=".045 0 -.07"
          rgba="0.9 0.6 0.6 1" size="0.046 .07" type="capsule"/>
```

```
    </body>
  </body>
</body>
```

**Ant Family.** The Ant environments consist of {4, 5, 6}-legged configurations based on the OpenAI Gymnasium suite. The standard 4-legged Ant serves as the unmodified baseline. To construct the 5-legged and 6-legged variants, we attach supplementary limbs to the central torso. These legs are arranged symmetrically to uniformly divide the circumference of the central body. All supplementary limbs retain the exact physical specifications and simulation constraints of the baseline appendages to ensure physical consistency across morphologies. Below is the main extension in the corresponding xml file:

- *5leg_ant.xml*: Uniformly distributes 5 legs radially, maintaining identical physical specifications and simulation constraints.

```
<!-- leg_{x} : {x * 72} deg -->
<body name="leg_{x}" pos="0 0 0">
  <geom fromto="0 0 0 {R*cos(x*72)} {R*sin(x*72)} 0" name="aux_{x
      }_geom" size="0.08" type="capsule"/>
  <body name="aux_{x}" pos="{R*cos(x*72)} {R*sin(x*72)} 0">
    <joint name="hip_{x}" type="hinge" axis="0 0 1" pos="0 0 0"
        range="-30 30"/>
    <geom fromto="0 0 0 {R*cos(x*72)} {R*sin(x*72)} 0" name="leg{x
        }_geom" size="0.08" type="capsule"/>
    <body pos="{R*cos(x*72)} {R*sin(x*72)} 0">
      <joint name="ankle_{x}" type="hinge" axis="{-sin(x*72)} {cos
          (x*72)} 0" pos="0 0 0" range="30 70"/>
      <geom fromto="0 0 0 {2R*cos(x*72)} {2R*sin(x*72)} 0" name="
          leg{x}_ankle_geom" size="0.08" type="capsule" rgba="..."
          />
    </body>
  </body>
</body>
```

- *6leg_ant.xml*: Adds a sixth leg compared to the 5-leg model, keeping identical morphology and uniformly spacing them at 60-degree intervals.

```
  \begin{lstlisting}[language=customXML]
<!-- leg_{x} : {30 + x * 60} deg -->
<body name="leg_{x}" pos="0 0 0">
  <geom fromto="0 0 0 {R*cos(30+x*60)} {R*sin(30+x*60)} 0" size="
      0.08" type="capsule"/>
  <body pos="{R*cos(30+x*60)} {R*sin(30+x*60)} 0">
    <joint name="hip_{x}" type="hinge" axis="0 0 1" range="-30 30"
        />
    <geom fromto="0 0 0 {R*cos(30+x*60)} {R*sin(30+x*60)} 0" size=
        "0.08" type="capsule"/>
    <body pos="{R*cos(30+x*60)} {R*sin(30+x*60)} 0">
      <joint name="ankle_{x}" type="hinge"
          axis="{-sin(30+x*60)} {cos(30+x*60)} 0"
          range="30 70"/>
      <geom fromto="0 0 0 {2R*cos(30+x*60)} {2R*sin(30+x*60)} 0"
          size="0.08" type="capsule" rgba="..."/>
    </body>
  </body>
</body>
```

**Centipede Family.** The Centipede environments consist of 4-, 6-, and 8-legged configurations (CentipedeFour, CentipedeSix, and CentipedeEight) directly loaded using the models provided by NerveNet (Wang et al., 2018). Because these environments natively provide varying morphologies, they

serve as baselines without further structural modification. All physical attributes, joint constraints, and segment structures are preserved exactly as defined in their original implementations.

ROBOT ARM MANIPULATION.

NAVIGATION.

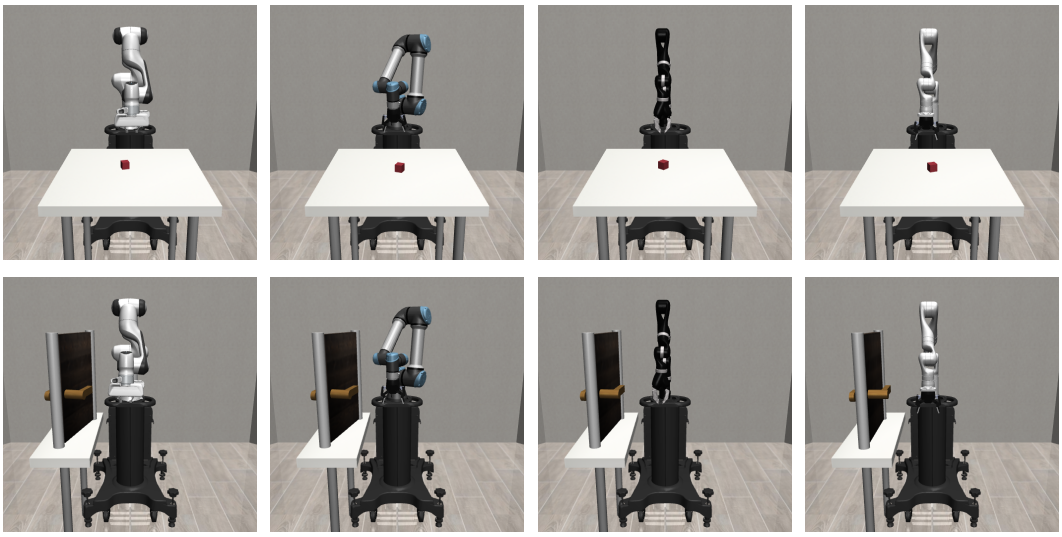


Figure 9: **Robosuite Environments and Arms:** The top row shows the **Lift** task, while the bottom row shows the **Door** task. The arms, from left to right, are the Panda, UR5e, Jaco, and Kinova3, respectively.

For manipulation, we use Robosuite (Zhu et al., 2020) as the simulation platform, leveraging its support for multiple robot-arm embodiments under shared task and scene configurations. This allows us to isolate morphology-induced domain shifts while keeping the manipulation objective fixed. We consider two tasks, **Block Lifting** and **Door Opening**, with Panda, UR5e, Jaco, and Kinova3 embodiments. Visualizations of the manipulation environments are shown in Figure 9.

NAVIGATION.

The navigation setting evaluates transfer between agents with substantially different embodiments while preserving the maze and navigation objective. We use a Point agent as the source embodiment and an Ant agent as the target embodiment. Four maze layouts are considered: Open, UMaze, Medium, and Large. Each maze is instantiated with both dense and sparse reward variants, yielding eight source-target configurations. Visualizations of the navigation environments are shown in Figure 10.

### A.2 SUB-SKILL TRANSFER

Sub-skill transfer considers settings where the source task provides only part of the knowledge required to solve the target task. Unlike morphology transfer, where the source and target share the same objective, the source and target tasks may differ, requiring the target agent to determine which aspects of the source knowledge remain useful. `CDRL-Bench` instantiates this setting at increasing levels of complexity, ranging from single-source transfer to complementary multi-source transfer and multi-source transfer across heterogeneous embodiments.

**Single-Source Sub-Skill Transfer.** As shown in Figure 2a, in this scenario, we aim to solve four manipulation tasks within the Robosuite environment: *Stack*, *NutAssembly*, *PickPlaceCan*, and *PickPlace*. Since each of these tasks requires the agent to lift an object during the initial phase of the episode before proceeding to task-specific operations, we designate the *Lift* task as the source domain. By doing so, we hypothesize that the foundational lifting primitives learned in the source

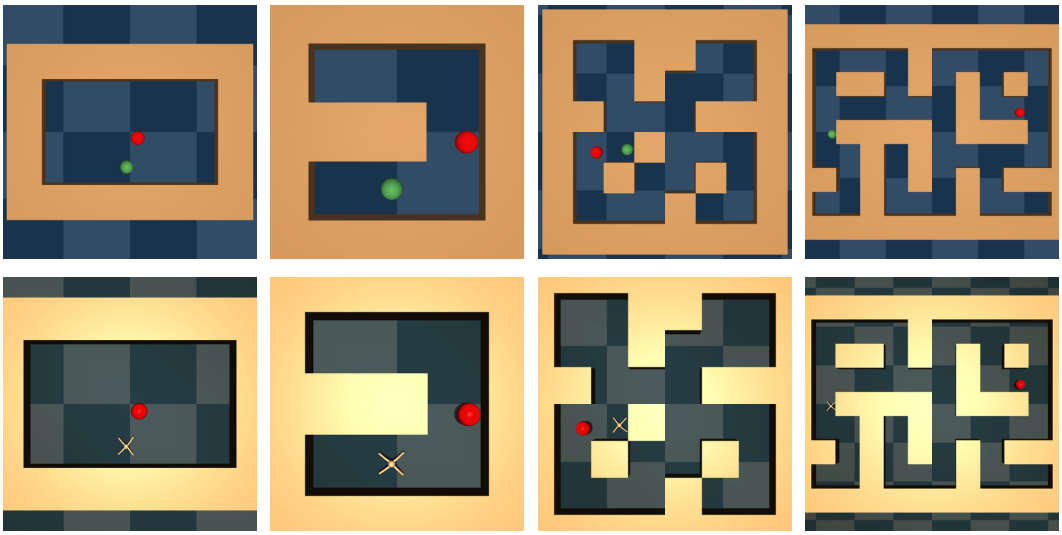


Figure 10: **Maze Navigation Environments:** Renderings of the ***PointMaze*** (top row) and ***AntMaze*** (bottom row) domains. The environments are organized by structural complexity, increasing from left to right: Open, U-Maze, Medium, and Large configurations.

domain can be effectively transferred to the more complex target domains. To maintain morphological consistency, the Franka Emika Panda robotic arm is utilized across all source and target tasks.

**Complementary Multi-Source Sub-Skill Transfer.** As shown in Figure 2b, in this scenario, we aim to solve the AntMaze suite of tasks. Within these environments, the agent must simultaneously master complex low-level locomotion and high-level spatial navigation to reach a designated goal position. To address this dual challenge, we designate PointMaze and Ant-v5 as the source domains. We hypothesize that PointMaze transfers structural maze representations and navigational heuristics, while Ant-v5 transfers foundational quadrupedal motor control primitives.

**Heterogeneous-Embodiment Multi-Source Sub-Skill Transfer.** As shown in Figure 2c, in this scenario, we address the *TwoArmLift* task within the Robosuite framework. This environment requires a dual-arm agent to learn coordinated bimanual manipulation for lifting a shared object. We evaluate four distinct robotic morphologies: Panda, UR5e, Jaco, and Kinova3. Each arm is initially trained in the single-arm *Lift* task, which serves as our source domain. We subsequently perform multi-source transfer to the *TwoArmLift* target domain to investigate whether individual, morphology-specific lifting primitives can effectively facilitate the acquisition of cooperative manipulation skills.

### A.3 Generalist-model transfer

In this benchmark, we use the publicly released 317M-parameter TD-MPC2 model (Hansen et al., 2024), which was jointly pretrained on 30 tasks from the DeepMind Control Suite (DMC) (Tassa et al., 2018), as the generalist source model. We treat each of the 30 DMC tasks as an independent source task, thereby evaluating knowledge transfer from individual source-task behaviors to the target domains.

A key requirement of this benchmark is that the target policy be substantially more compact than the source model. Under this setting, QAvatar is the only applicable method among the considered approaches, as it transfers knowledge without requiring the target policy to retain or directly reuse the source model's feature representations. In contrast, FT and CAT rely on source-model features for transfer and therefore assume architectural compatibility with, or direct access to, the source representation, making them unsuitable when the target policy adopts a substantially smaller architecture.

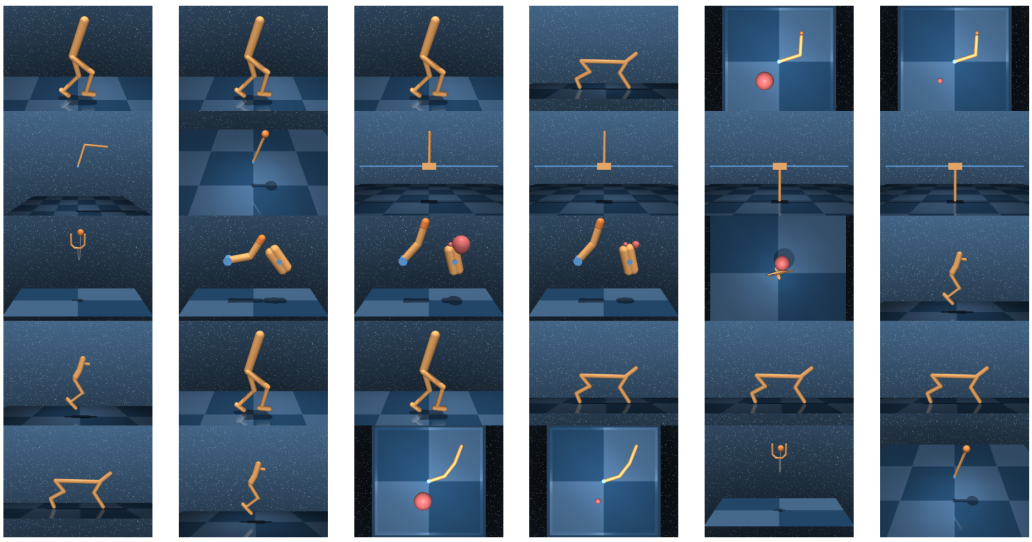
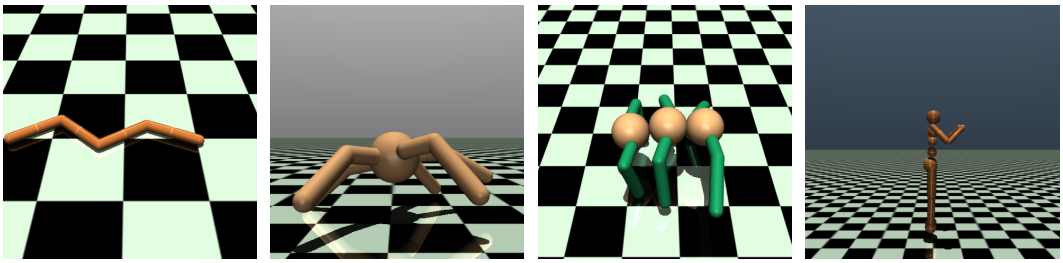


Figure 11: Overview of the 30 source tasks from the DeepMind Control Suite used to jointly pretrain the 317M-parameter TD-MPC2 generalist model. Each environment is treated as an independent source task in our transfer experiments.

Figure 11 provides an overview of all 30 source tasks used to pretrain the 317M-parameter TD-MPC2 model, while the target domains considered in our transfer experiments are shown in Figure 12.



(a) 4limb_swimmer (b) 5leg_ant (c) CentipedeSix (d) Humanoid-v5

Figure 12: Target domains used to evaluate generalist model transfer

### A.4 State and Action Space Summary

Table 2 summarizes the state and action dimensions of each benchmark environment, together with the corresponding `env_name` used by the environment construction interface.

Table 2: Summary of benchmark environments and their state and action spaces. For navigation, each maze layout is available with both dense- and sparse-reward variants.

| Domain | Task / Family | Environment | `env_name` | State Dim. | Action Dim. |
|---|---|---|---|---|---|
| Locomotion | Cheetah | 2-leg Cheetah | `2leg_cheetah` | 17 | 6 |
| | | 3-leg Cheetah | `3leg_cheetah` | 23 | 9 |
| | | 4-leg Cheetah | `4leg_cheetah` | 29 | 12 |
| | Swimmer | 3-limb Swimmer | `3limb_swimmer` | 8 | 2 |
| | | 4-limb Swimmer | `4limb_swimmer` | 10 | 3 |
| | | 5-limb Swimmer | `5limb_swimmer` | 12 | 4 |

Continued on next page

| Domain | Task / Family | Environment | env_name | State Dim. | Action Dim. |
|---|---|---|---|---|---|
| Manipulation | Ant | 4-leg Ant | 4leg_ant | 105 | 8 |
| | | 5-leg Ant | 5leg_ant | 127 | 10 |
| | | 6-leg Ant | 6leg_ant | 149 | 12 |
| | Centipede | CentipedeFour | CentipedeFour | 97 | 10 |
| | | CentipedeSix | CentipedeSix | 139 | 16 |
| | | CentipedeEight | CentipedeEight | 181 | 22 |
| | Block Lifting | Panda | Lift_Panda | 60 | 7 |
| | | UR5e | Lift_UR5e | 63 | 7 |
| | | Jaco | Lift_Jaco | 68 | 7 |
| | | Kinova3 | Lift_Kinova3 | 68 | 7 |
| | Door Opening | Panda | Door_Panda | 64 | 7 |
| | | UR5e | Door_UR5e | 67 | 7 |
| | | Jaco | Door_Jaco | 72 | 7 |
| | | Kinova3 | Door_Kinova3 | 72 | 7 |
| | NutAssembly | Panda | NutAssembly_Panda | 78 | 7 |
| | Stack | Panda | Stack_Panda | 73 | 7 |
| | PickPlaceCan | Panda | PickPlaceCan_Panda | 64 | 7 |
| | PickPlace | Panda | PickPlace_Panda | 106 | 7 |
| | TwoArmLift | Jaco–UR5e | TwoArmLift_JacoUR5e | 130 | 14 |
| | | Kinova3–Jaco | TwoArmLift_Kinova3Jaco | 135 | 14 |
| | | Kinova3–UR5e | TwoArmLift_Kinova3UR5e | 130 | 14 |
| | | Panda–Jaco | TwoArmLift_PandaJaco | 127 | 14 |
| | | Panda–Kinova3 | TwoArmLift_PandaKinova3 | 127 | 14 |
| | | Panda–UR5e | TwoArmLift_PandaUR5e | 122 | 14 |
| Navigation | PointMaze | Open | PointMaze_Open-v3 | 8 | 2 |
| | | UMaze | PointMaze_UMaze-v3 | 8 | 2 |
| | | Medium | PointMaze_Medium-v3 | 8 | 2 |
| | | Large | PointMaze_Large-v3 | 8 | 2 |
| | AntMaze | Open | AntMaze_Open-v3 | 109 | 8 |
| | | UMaze | AntMaze_UMaze-v3 | 109 | 8 |
| | | Medium | AntMaze_Medium-v3 | 109 | 8 |
| | | Large | AntMaze_Large-v3 | 109 | 8 |
| Locomotion | Humanoid | Humanoid-v5 | Humanoid-v5 | 348 | 17 |

## B Benchmark Baselines Implementation Details

To ensure a controlled and fair comparison, all methods built on the same underlying RL algorithm share an identical base configuration within each task family. We use one base configuration for Robosuite-based manipulation tasks and another for all non-Robosuite environments. As summarized in Tables 3, respectively.

**QAvatar-SAC.** QAvatar is originally formulated as an SAC-based cross-domain transfer algorithm. We follow the official implementation provided by the authors.[1] QAvatar transfers knowledge by combining the learned target-domain critic with pretrained source-domain critics, while learning a separate cross-domain mapping for each source through Bellman consistency.

Suppose there are $K$ source domains. For source $i$, let $Q_{\mathrm{src},i}$ denote its pretrained and frozen Q-function, and let $\phi_i$ and $\psi_i$ denote the corresponding target-to-source state and action mappings. The mapped source critic is

$$\widetilde{Q}_i^{(t)}(s,a) = Q_{\mathrm{src},i}\left(F_{s,i}\left(\phi_i^{(t)}(s)\right), F_{a,i}\left(\psi_i^{(t)}(a)\right)\right),$$

where $F_{s,i}$ and $F_{a,i}$ are the pretrained and frozen RealNVP flow decoders of source domain $i$.

[1] https://github.com/rl-bandits-lab/Cross-Domain-RL

The target policy is updated using the multi-source hybrid critic

$$Q_{\mathrm{hyb}}^{(t)}(s,a) = w_{\mathrm{tar}}^{(t)} Q_{\mathrm{tar}}^{(t)}(s,a) + \sum_{i=1}^{K} \alpha_i^{(t)} \widetilde{Q}_i^{(t)}(s,a),$$

where $w_{\mathrm{tar}}^{(t)} + \sum_{i=1}^{K} \alpha_i^{(t)} = 1$.

For each source domain, the mappings are optimized independently using the cross-domain Bellman loss

$$\mathcal{L}_{\mathrm{CD},i}(\phi_i, \psi_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\mathrm{tar}}}\left[\left(r + \gamma \mathbb{E}_{a'\sim\pi_{\mathrm{tar}}(\cdot|s')}[\widetilde{Q}_i(s',a')] - \widetilde{Q}_i(s,a)\right)^2\right].$$

Every $N_\alpha$ environment steps, QAvatar evaluates the target TD error $e_{\mathrm{tar}}^{(t)} = \|\epsilon_{\mathrm{td}}^{(t)}\|_{\mathcal{D}_{\mathrm{tar}}}$ and the cross-domain Bellman error $e_i^{(t)} = \|\epsilon_{\mathrm{cd},i}^{(t)}\|_{\mathcal{D}_{\mathrm{tar}}}$ for each source. The critic weights are then updated by inverse-error normalization:

$$Z^{(t)} = \frac{1}{e_{\mathrm{tar}}^{(t)}} + \sum_{j=1}^{K} \frac{1}{e_j^{(t)}}, \qquad w_{\mathrm{tar}}^{(t)} = \frac{1/e_{\mathrm{tar}}^{(t)}}{Z^{(t)}}, \qquad \alpha_i^{(t)} = \frac{1/e_i^{(t)}}{Z^{(t)}}.$$

The weights remain fixed between consecutive $N_\alpha$ updates. The QAvatar-SAC-specific hyperparameters used in our experiments are summarized in Table 4.

**CAT-PPO.** CAT was originally proposed as a PPO-based cross-domain transfer method. It performs feature-level transfer by combining intermediate representations from pretrained source networks with those of the target networks.

Suppose there are $K$ source domains. CAT evaluates each mapped source policy in the target environment and converts its episodic return $u_i$ into an adaptive source weight

$$w_i = \frac{\exp(u_i)}{\sum_{k=1}^{K} \exp(u_k)}.$$

Let $\mathbf{z}_\pi^{(j)}$ and $\mathbf{z}_V^{(j)}$ denote the pre-activation representations at the $j$-th hidden layer of the target policy and value networks, and let $\mathbf{z}_{\pi_i}^{(j)}$ and $\mathbf{z}_{V_i}^{(j)}$ denote the corresponding source-domain representations. CAT constructs the transferred features as

$$\tilde{\mathbf{z}}_\pi^{(j)} = p_\pi \mathbf{z}_\pi^{(j)} + (1-p_\pi)\sum_{i=1}^{K} w_i \mathbf{z}_{\pi_i}^{(j)}, \qquad \tilde{\mathbf{z}}_V^{(j)} = p_V \mathbf{z}_V^{(j)} + (1-p_V)\sum_{i=1}^{K} w_i \mathbf{z}_{V_i}^{(j)}.$$

Here, $p_\pi, p_V \in (0,1)$ control the relative contributions of the target and source features for the policy and value networks, respectively. In the official implementation, these student-preference parameters are optimized through backpropagation using

$$\mathcal{L}_{p_\pi} = -\mathbb{E}\left[\log p_\pi\right], \qquad \mathcal{L}_{p_V} = -\mathbb{E}\left[\log p_V\right].$$

The cross-domain mapping functions are optimized using mutual-information and cycle-consistency objectives. The mutual-information loss is

$$\mathcal{L}_{\mathrm{MI}}(\phi) = -\mathbb{E}_{s_t\sim\rho_{\mathrm{tar}}}\left[\log q_\omega\left(s_t \mid \phi(s_t)\right)\right],$$

where $q_\omega(s_t \mid \phi(s_t))$ is a variational decoder that reconstructs the target-domain state from its mapped representation. The cycle-consistency loss is

$$\begin{aligned}\mathcal{L}_{\mathrm{cycle}}(\phi,\phi^-) = \mathbb{E}_{s_t\sim\rho_{\mathrm{tar}}}&\left[\left\|\phi^-(\phi(s_t)) - s_t\right\|_2^2\right] \\ &+ \mathbb{E}_{s_s\sim\rho_{\mathrm{src}}}\left[\left\|\phi\big(\phi^-(s_s)\big) - s_s\right\|_2^2\right],\end{aligned}$$

where $\phi$ and $\phi^-$ denote the forward and reverse mapping functions, respectively.

During PPO optimization, the overall objective is

$$\mathcal{L}_{\mathrm{CAT}} = \mathcal{L}_{\mathrm{PPO}} + \lambda_\pi \mathcal{L}_{p_\pi} + \lambda_V \mathcal{L}_{p_V} + \lambda_{\mathrm{MI}} \mathcal{L}_{\mathrm{MI}}.$$

**CAT-SAC.** CAT-SAC extends CAT-PPO to the SAC backbone while retaining the same transfer architecture and auxiliary losses. The main implementation difference is that, inspired by QAvatar-SAC, we employ flow-based cross-domain mapping functions and periodically reevaluate the adaptive source weights with frequency $N_w = 1000$.

The CAT-specific hyperparameters used for both CAT-PPO and CAT-SAC are summarized in Table 5.

**FT-SAC.** FT-SAC initializes the target agent from pretrained SAC models of one or more source domains. When multiple source models are available, we compute an unweighted parameter-wise average of their network weights to obtain a unified initialization. To accommodate dimensionality differences between the source and target domains, layers whose input or output dimensions depend on the environment are replaced and randomly initialized. Specifically, we replace the first layer of the actor representation network and the first layer of both Q-functions to match the target observation and state–action dimensions, respectively. The actor output heads for the policy mean and log standard deviation are also reinitialized to match the target action dimension. All remaining hidden layers inherit the pretrained (or averaged) source weights. During target-domain fine-tuning, no parameters are frozen; the entire actor and critic networks are jointly optimized using target-domain experience.

**FT-PPO.** Similar to FT-SAC, FT-PPO initializes the target policy and value networks from one or multiple pretrained PPO source models. When multiple source models are available, only shape-compatible parameters are averaged, while dimension-dependent layers are reinitialized to match the target observation and action spaces. All remaining compatible layers inherit the transferred source weights. During fine-tuning, no parameters are frozen; the entire policy and value networks, including the state-independent log standard deviation, are optimized end-to-end using target-domain experience.

**SAC.** We implement the Soft Actor-Critic (SAC) algorithm following its standard formulation, utilizing a squashed Gaussian policy for continuous action spaces and a clipped double-Q critic architecture to mitigate overestimation bias.

The twin critic networks, parameterized by $\theta$, are updated by minimizing the soft Bellman residual:

$$L_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V_{\bar{\theta}}(s_{t+1})\right]\right)\right)^2\right]$$

where the soft state-value function $V_{\bar{\theta}}(s_{t+1})$ is implicitly defined using the target Q-networks and the entropy-augmented policy:

$$V_{\bar{\theta}}(s_{t+1}) = \mathbb{E}_{a_{t+1}\sim\pi_\phi}\left[\min_{j=1,2} Q_{\bar{\theta}_j}(s_{t+1},a_{t+1}) - \alpha\log\pi_\phi(a_{t+1}|s_{t+1})\right]$$

The actor network, parameterized by $\phi$, is optimized by minimizing the Kullback-Leibler (KL) divergence between the policy and the exponentiated target Q-distribution. By applying the reparameterization trick via a squashed Gaussian distribution ($a_t = f_\phi(\epsilon_t; s_t)$), the differentiable actor objective is formulated as:

$$L_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}\left[\alpha\log\pi_\phi(f_\phi(\epsilon_t;s_t)|s_t) - \min_{j=1,2} Q_{\theta_j}(s_t, f_\phi(\epsilon_t;s_t))\right]$$

Finally, to automate the entropy-reward trade-off, the temperature parameter $\alpha$ is dynamically autotuned. We optimize $\alpha$ by minimizing the following objective with respect to a target entropy $\bar{\mathcal{H}}$, which is heuristically set to the negative dimension of the action space:

$$L(\alpha) = \mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\phi}\left[-\alpha\log\pi_\phi(a_t|s_t) - \alpha\bar{\mathcal{H}}\right]$$

**PPO.** We implement the Proximal Policy Optimization (PPO) algorithm utilizing a decoupled actor-critic architecture. To stabilize training, observation inputs are standardized using a running mean and variance prior to passing through the networks.

The baseline value function, parameterized by $\psi$, is utilized to compute the Generalized Advantage Estimation (GAE) $\hat{A}_t$ to balance the bias-variance tradeoff in return estimation. The critic objective

minimizes the squared error against the GAE returns $\hat{R}_t$. We employ a clipped value function objective:

$$L_V(\psi) = \mathbb{E}_t \left[ \max \left( \left( V_\psi(s_t) - \hat{R}_t \right)^2, \left( V_{\psi_{\text{old}}}(s_t) + \text{clip}\left( V_\psi(s_t) - V_{\psi_{\text{old}}}(s_t), -\epsilon_v, \epsilon_v \right) - \hat{R}_t \right)^2 \right) \right]$$

The actor network, parameterized by $\phi$, models a Gaussian policy where the network outputs the action mean, while the log standard deviation is maintained as a state-independent, learnable parameter. The policy is optimized by minimizing a clipped surrogate objective that penalizes overly large policy updates relative to the old policy $r_t(\phi) = \frac{\pi_\phi(a_t|s_t)}{\pi_{\phi_{\text{old}}}(a_t|s_t)}$. The total loss incorporates an entropy bonus $\mathcal{S}[\pi_\phi]$ to encourage exploration:

$$L_\pi(\phi) = \mathbb{E}_t \left[ \max \left( -r_t(\phi)\hat{A}_t, -\text{clip}\left( r_t(\phi), 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) - c_1 \mathcal{S}[\pi_\phi](s_t) \right]$$

To further guard against destructive policy collapse, we integrate an early stopping mechanism at the mini-batch level: if the approximate Kullback-Leibler (KL) divergence between the updated and previous policies exceeds a predefined target threshold ($D_{\text{KL}} > \text{KL}_{\text{target}}$), the optimization loop for the current epoch is immediately terminated. Finally, the optimization process applies global gradient clipping and linear learning rate annealing.

Table 3: Base SAC and PPO hyperparameters.

| Algorithm | Hyperparameter | Non-Robosuite | Robosuite |
|---|---|---|---|
| SAC | Actor learning rate | $3 \times 10^{-4}$ | $1 \times 10^{-4}$ |
| | Critic learning rate | $3 \times 10^{-4}$ | $1 \times 10^{-4}$ |
| | Entropy temperature $\alpha$ | Auto-tuned | Fixed at 0.02 |
| | Batch size | 256 | 1024 |
| | Replay buffer size | $1 \times 10^{6}$ | $1 \times 10^{6}$ |
| | Discount factor $\gamma$ | 0.99 | 0.99 |
| | Target smoothing coefficient $\tau$ | 0.005 | 0.005 |
| | Learning starts | 100 | 5000 |
| | Policy update frequency | 1 | 1 |
| | Target network update frequency | 1 | 1 |
| | Number of hidden layers | 2 | 2 |
| | Hidden units per layer | 256 | 256 |
| PPO | Learning rate | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ |
| | Rollout length | 2048 | 2048 |
| | Minibatch size | 64 | 64 |
| | Number of update epochs | 10 | 10 |
| | Discount factor $\gamma$ | 0.99 | 0.99 |
| | GAE coefficient $\lambda$ | 0.95 | 0.95 |
| | PPO clipping coefficient $\epsilon$ | 0.2 | 0.2 |
| | Entropy coefficient | 0.0 | 0.0 |
| | Value-loss coefficient | 0.5 | 0.5 |
| | Maximum gradient norm | 0.5 | 0.5 |
| | Advantage normalization | True | True |
| | Number of hidden layers | 2 | 2 |
| | Hidden units per layer | 64 | 64 |

Table 4: QAvatar-SAC-specific hyperparameters.

| Hyperparameter | Value |
|---|---|
| Mapping function learning rate | $1 \times 10^{-4}$ |
| Mapping function optimizer | AdamW |
| Mapping hidden dimension | 256 |
| $\alpha$ update frequency $N_\alpha$ | 1000 |

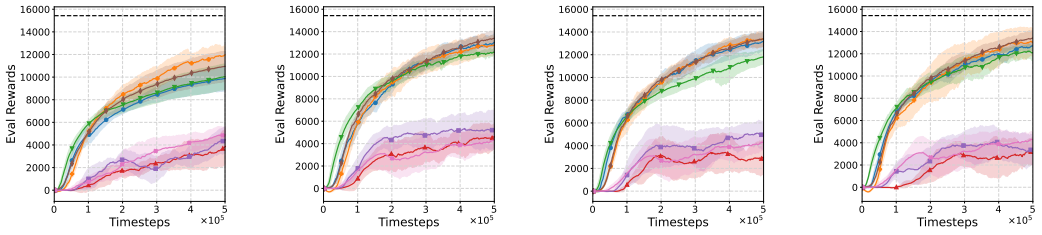
Table 5: CAT-specific hyperparameters used in our experiments. Here, $t \in [0, 1]$ denotes the normalized training progress.

| Hyperparameter | Value |
|---|---|
| Mapping learning rate | $3 \times 10^{-4}$ |
| Policy student-preference coefficient $\lambda_\pi$ | $3 \times 10^{-4}$ |
| Value student-preference coefficient $\lambda_V$ and cycle-consistency coefficient $\lambda_{\text{cycle}}$ | $\begin{cases} 0, & t < 0.45, \\ 0.05\dfrac{t-0.45}{0.45}, & 0.45 \le t < 0.9, \\ \text{None}, & t \ge 0.9 \end{cases}$ |
| Mutual-information coefficient $\lambda_{\text{MI}}$ | 0.5 |

# C Additional Results

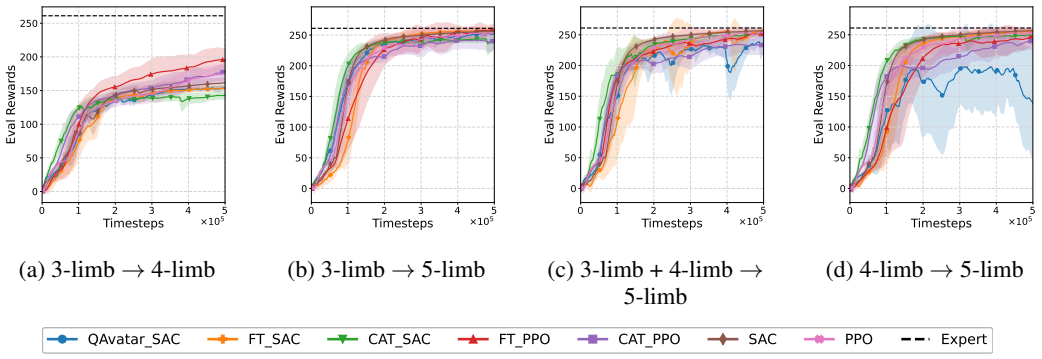
## C.1 Learning Curves for Morphology Transfer

We report the complete learning curves for all morphology-transfer configurations considered in CDRL-Bench. Results include locomotion transfer across Cheetah, Swimmer, Ant, and Centipede morphologies, robot-arm transfer on Lift and Door, and Point-to-Ant navigation.



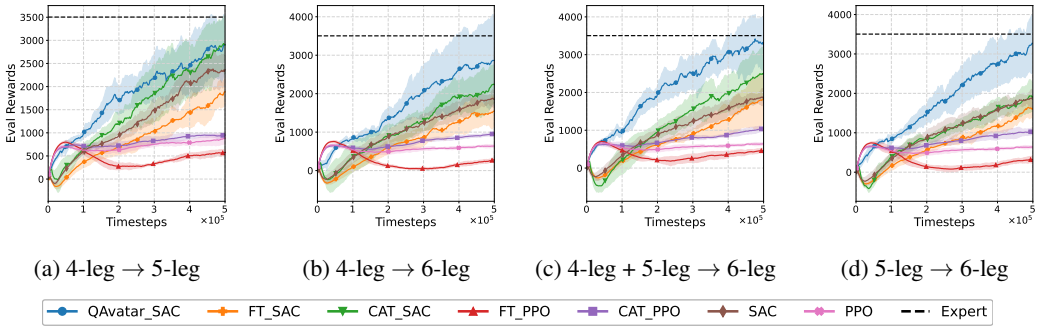

(a) 2-leg → 3-leg (b) 2-leg → 4-leg (c) 2-leg + 3-leg → 4-leg (d) 3-leg → 4-leg

Figure 13: Morphology transfer performance on Cheetah tasks against an expert baseline.



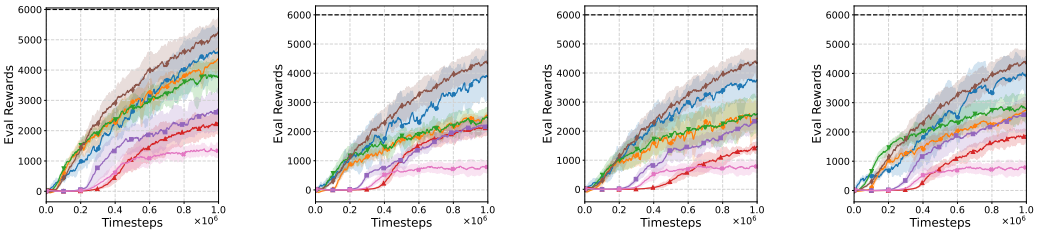

(a) 3-limb → 4-limb (b) 3-limb → 5-limb (c) 3-limb + 4-limb → 5-limb (d) 4-limb → 5-limb

Figure 14: Morphology transfer performance on Swimmer tasks against an expert baseline.



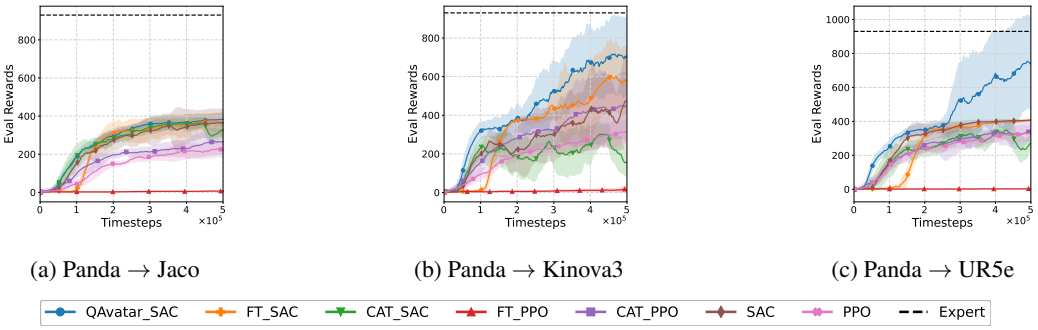

(a) 4-leg → 5-leg (b) 4-leg → 6-leg (c) 4-leg + 5-leg → 6-leg (d) 5-leg → 6-leg

Figure 15: Morphology transfer performance on Ant tasks against an expert baseline.



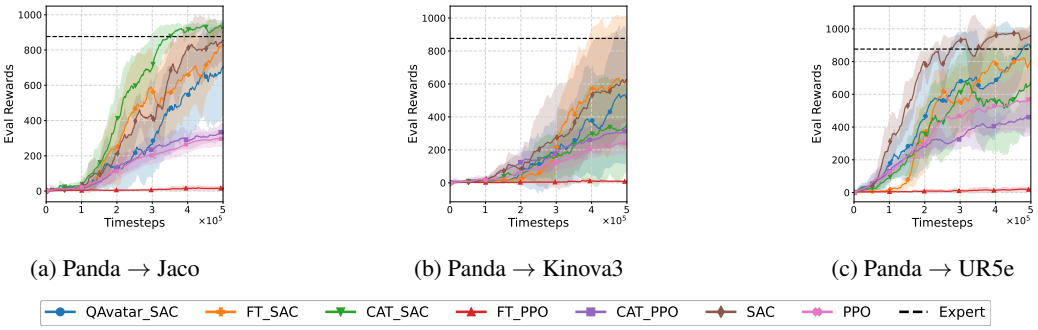

(a) Four → Six (b) Four → Eight (c) Six → Eight (d) Four+Six → Eight

Figure 16: Morphology transfer performance on Centipede tasks against an expert baseline.



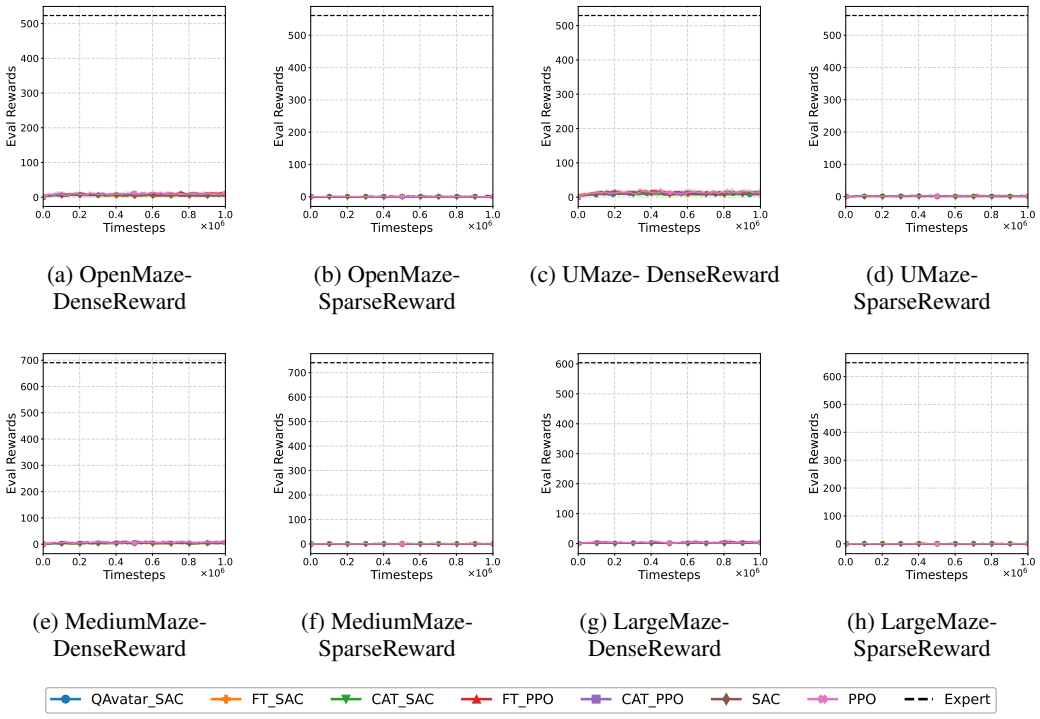

(a) Panda → Jaco (b) Panda → Kinova3 (c) Panda → UR5e

Figure 17: Morphology transfer performance on Lift tasks against an expert baseline.




(a) Panda → Jaco (b) Panda → Kinova3 (c) Panda → UR5e

Figure 18: Morphology transfer performance on Door tasks against an expert baseline.



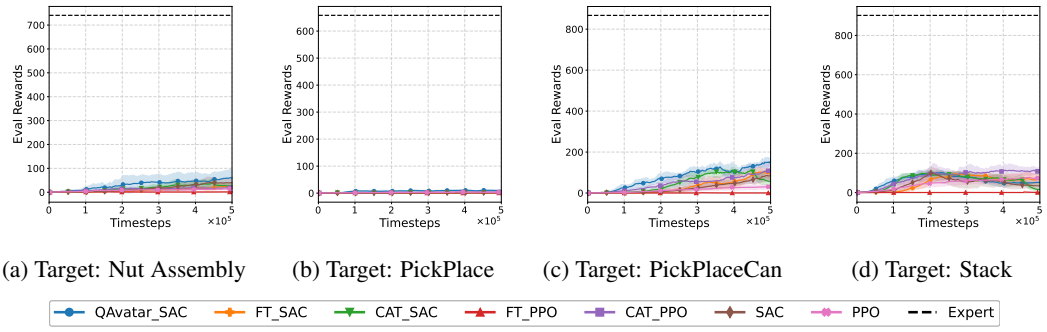

(a) OpenMaze-DenseReward (b) OpenMaze-SparseReward (c) UMaze- DenseReward (d) UMaze-SparseReward

(e) MediumMaze-DenseReward (f) MediumMaze-SparseReward (g) LargeMaze-DenseReward (h) LargeMaze-SparseReward

Figure 19: Single-source morphology transfer from PointMaze to AntMaze across different maze layouts and reward settings.

### C.2 Learning Curves for Sub-Skill(s) Transfer

We provide the complete learning curves for the three sub-skill transfer settings described in Section 4.2: Single-Source Manipulation Sub-Skills, Complementary Multi-Source Sub-Skill Transfer, Heterogeneous-Embodiment Multi-Source Sub-Skill Transfer.




(a) Target: Nut Assembly (b) Target: PickPlace (c) Target: PickPlaceCan (d) Target: Stack

Figure 20: Performance on the Single-Source Manipulation Sub-Skills Transfer experiments. Source domains: Lift Panda. Target robot arm: Panda.

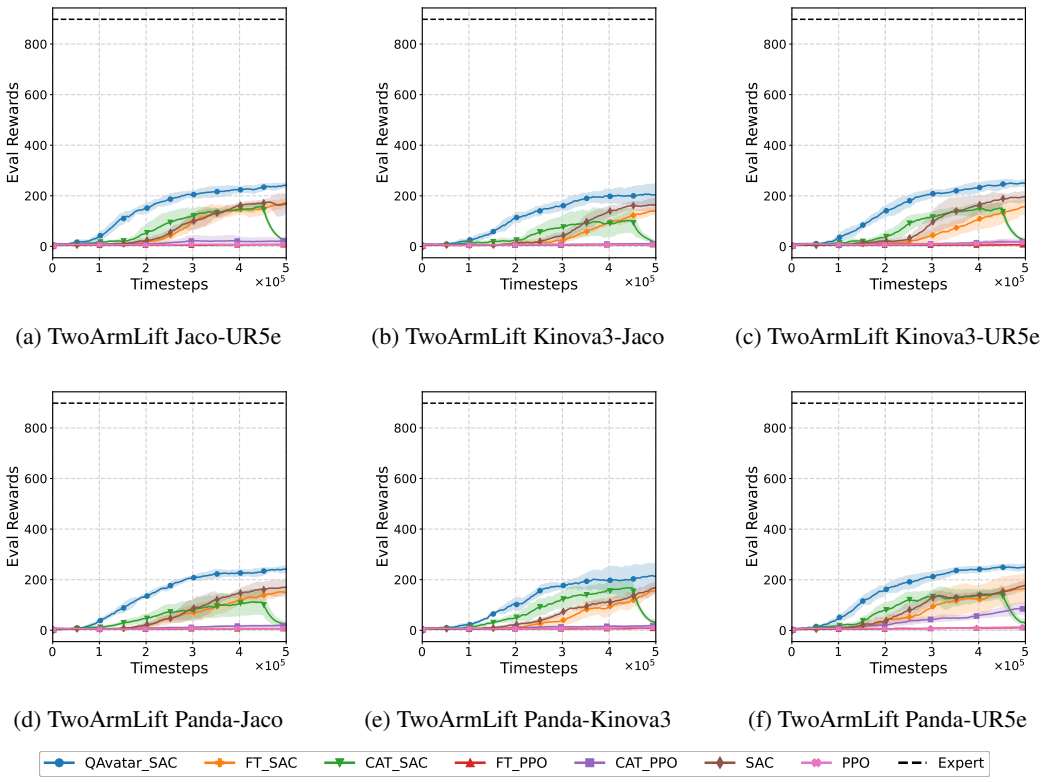



Figure 21: Performance on the TwoArmLift target domains. Source domains: Lift Panda, Lift UR5e, Lift Kinova3, and Lift Jaco.

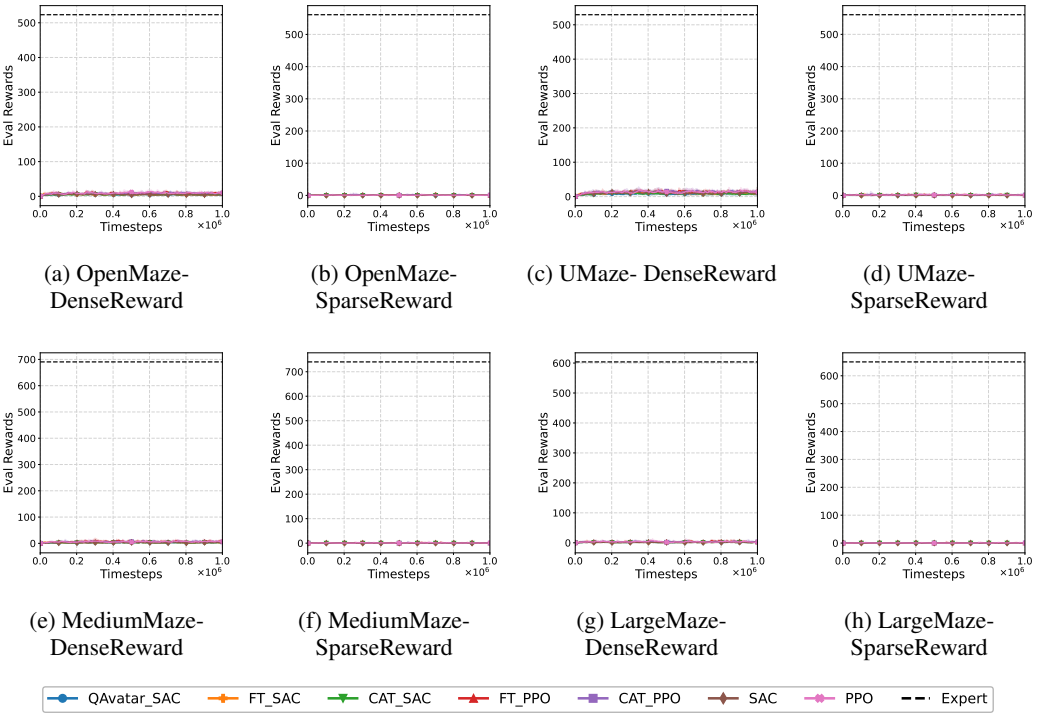



Figure 22: Learning Curves for Transfer from PointNavigate and Ant Locomotion to AntMaze.

### C.3 Full Regret table

Table 6 reports Average Normalized Regret for every source–target configuration in the benchmark. Regret is computed from the expert-normalized learning curves according to Section 4.5; lower values indicate more efficient learning over the target-domain interaction budget.

| Task/Alg | QAvatar-SAC | CAT-SAC | FT-SAC | SAC | CAT-PPO | FT-PPO | PPO |
|---|---|---|---|---|---|---|---|
| *Morphology:Cheetah* | | | | | | | |
| 2leg → 3leg | $0.51 \pm 0.03$ | $0.48 \pm 0.05$ | $0.44 \pm 0.04$ | $0.46 \pm 0.05$ | $0.84 \pm 0.04$ | $0.86 \pm 0.02$ | $0.81 \pm 0.04$ |
| 2leg → 4leg | $0.36 \pm 0.04$ | $0.34 \pm 0.03$ | $0.37 \pm 0.05$ | $0.34 \pm 0.02$ | $0.70 \pm 0.05$ | $0.77 \pm 0.06$ | $0.77 \pm 0.06$ |
| 2leg+3leg → 4leg | $0.33 \pm 0.03$ | $0.38 \pm 0.04$ | $0.35 \pm 0.02$ | $0.34 \pm 0.02$ | $0.74 \pm 0.07$ | $0.81 \pm 0.05$ | $0.77 \pm 0.06$ |
| 3leg → 4leg | $0.36 \pm 0.02$ | $0.35 \pm 0.04$ | $0.36 \pm 0.08$ | $0.34 \pm 0.02$ | $0.78 \pm 0.05$ | $0.84 \pm 0.04$ | $0.77 \pm 0.06$ |
| *Morphology:Swimmer* | | | | | | | |
| 3limb → 4limb | $0.56 \pm 0.02$ | $0.54 \pm 0.01$ | $0.57 \pm 0.02$ | $0.54 \pm 0.02$ | $0.51 \pm 0.03$ | $0.49 \pm 0.06$ | $0.54 \pm 0.03$ |
| 3limb → 5limb | $0.25 \pm 0.02$ | $0.23 \pm 0.02$ | $0.29 \pm 0.02$ | $0.25 \pm 0.01$ | $0.28 \pm 0.04$ | $0.30 \pm 0.06$ | $0.27 \pm 0.04$ |
| 3limb_4limb → 5limb | $0.32 \pm 0.04$ | $0.22 \pm 0.02$ | $0.31 \pm 0.08$ | $0.25 \pm 0.01$ | $0.30 \pm 0.03$ | $0.27 \pm 0.02$ | $0.27 \pm 0.04$ |
| 4limb → 5limb | $0.43 \pm 0.18$ | $0.21 \pm 0.01$ | $0.30 \pm 0.04$ | $0.25 \pm 0.01$ | $0.31 \pm 0.04$ | $0.35 \pm 0.08$ | $0.27 \pm 0.04$ |
| *Morphology: Ant* | | | | | | | |
| 4leg → 5leg | $0.46 \pm 0.07$ | $0.55 \pm 0.08$ | $0.71 \pm 0.05$ | $0.62 \pm 0.03$ | $0.70 \pm 0.01$ | $0.78 \pm 0.01$ | $0.72 \pm 0.01$ |
| 4leg → 6leg | $0.49 \pm 0.10$ | $0.64 \pm 0.08$ | $0.73 \pm 0.04$ | $0.66 \pm 0.04$ | $0.70 \pm 0.01$ | $0.81 \pm 0.01$ | $0.74 \pm 0.01$ |
| 4leg_5leg → 6leg | $0.38 \pm 0.06$ | $0.76 \pm 0.13$ | $0.72 \pm 0.04$ | $0.66 \pm 0.04$ | $0.70 \pm 0.01$ | $0.79 \pm 0.01$ | $0.74 \pm 0.01$ |
| 5leg → 6leg | $0.46 \pm 0.07$ | $0.67 \pm 0.03$ | $0.74 \pm 0.02$ | $0.66 \pm 0.04$ | $0.69 \pm 0.01$ | $0.80 \pm 0.01$ | $0.74 \pm 0.01$ |
| *Morphology:Centipede* | | | | | | | |
| Four → Eight | $0.66 \pm 0.09$ | $0.74 \pm 0.03$ | $0.76 \pm 0.01$ | $0.60 \pm 0.05$ | $0.84 \pm 0.01$ | $0.85 \pm 0.02$ | $0.92 \pm 0.01$ |
| Four → Six | $0.61 \pm 0.06$ | $0.61 \pm 0.04$ | $0.60 \pm 0.04$ | $0.52 \pm 0.06$ | $0.79 \pm 0.06$ | $0.85 \pm 0.04$ | $0.88 \pm 0.02$ |
| Four+Six → Eight | $0.66 \pm 0.09$ | $0.68 \pm 0.04$ | $0.75 \pm 0.02$ | $0.60 \pm 0.05$ | $0.79 \pm 0.01$ | $0.88 \pm 0.02$ | $0.92 \pm 0.01$ |
| Six → Eight | $0.64 \pm 0.07$ | $0.75 \pm 0.06$ | $0.75 \pm 0.05$ | $0.60 \pm 0.05$ | $0.84 \pm 0.03$ | $0.92 \pm 0.01$ | $0.92 \pm 0.01$ |
| *Morphology: Door Opening* | | | | | | | |
| Panda → Jaco | $0.74 \pm 0.12$ | $0.46 \pm 0.08$ | $0.63 \pm 0.14$ | $0.62 \pm 0.08$ | $0.83 \pm 0.04$ | $1.00 \pm 0.00$ | $0.84 \pm 0.04$ |
| Panda → Kinova3 | $0.85 \pm 0.14$ | $0.86 \pm 0.10$ | $0.80 \pm 0.13$ | $0.77 \pm 0.16$ | $0.85 \pm 0.06$ | $1.00 \pm 0.00$ | $0.90 \pm 0.04$ |
| Panda → UR5e | $0.50 \pm 0.23$ | $0.60 \pm 0.14$ | $0.56 \pm 0.11$ | $0.27 \pm 0.02$ | $0.70 \pm 0.06$ | $0.99 \pm 0.01$ | $0.63 \pm 0.09$ |
| *Morphology: Block Lifting* | | | | | | | |
| Panda → Jaco | $0.73 \pm 0.02$ | $0.73 \pm 0.03$ | $0.76 \pm 0.04$ | $0.75 \pm 0.05$ | $0.84 \pm 0.01$ | $1.01 \pm 0.00$ | $0.87 \pm 0.02$ |
| Panda → Kinova3 | $0.57 \pm 0.10$ | $0.81 \pm 0.07$ | $0.69 \pm 0.07$ | $0.74 \pm 0.11$ | $0.72 \pm 0.09$ | $1.01 \pm 0.00$ | $0.82 \pm 0.03$ |
| Panda → UR5e | $0.59 \pm 0.11$ | $0.77 \pm 0.03$ | $0.76 \pm 0.01$ | $0.71 \pm 0.03$ | $0.77 \pm 0.04$ | $1.02 \pm 0.00$ | $0.79 \pm 0.02$ |
| *Morphology: Maze* | | | | | | | |
| LargeDense | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ |
| LargeSparse | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| MediumDense | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ |
| MediumSparse | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| OpenDense | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $0.98 \pm 0.00$ | $0.98 \pm 0.00$ | $0.98 \pm 0.00$ |
| OpenSparse | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| UMazeDense | $0.98 \pm 0.00$ | $0.98 \pm 0.00$ | $0.98 \pm 0.00$ | $0.98 \pm 0.00$ | $0.98 \pm 0.00$ | $0.97 \pm 0.00$ | $0.98 \pm 0.00$ |
| UMazeSparse | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| *SubSkills-Categories1* | | | | | | | |
| Lift → NutAssembly | $0.96 \pm 0.03$ | $0.98 \pm 0.01$ | $0.99 \pm 0.00$ | $0.99 \pm 0.01$ | $0.99 \pm 0.00$ | $1.00 \pm 0.00$ | $0.99 \pm 0.00$ |
| Lift → PickPlace | $0.99 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| Lift → PickPlaceCan | $0.92 \pm 0.01$ | $0.95 \pm 0.01$ | $0.98 \pm 0.01$ | $0.98 \pm 0.01$ | $0.96 \pm 0.02$ | $1.00 \pm 0.00$ | $0.99 \pm 0.01$ |
| Lift → Stack | $0.94 \pm 0.00$ | $0.93 \pm 0.01$ | $0.95 \pm 0.01$ | $0.95 \pm 0.01$ | $0.92 \pm 0.02$ | $1.00 \pm 0.00$ | $0.96 \pm 0.02$ |
| *SubSkills-Categories2* | | | | | | | |
| LargeDense | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ |
| LargeSparse | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| MediumDense | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ |
| MediumSparse | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| OpenDense | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $0.99 \pm 0.00$ | $0.98 \pm 0.00$ | $0.98 \pm 0.00$ | $0.98 \pm 0.00$ |
| OpenSparse | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| UMazeDense | $0.99 \pm 0.00$ | $0.98 \pm 0.00$ | $0.98 \pm 0.00$ | $0.98 \pm 0.00$ | $0.97 \pm 0.00$ | $0.97 \pm 0.00$ | $0.98 \pm 0.00$ |
| UMazeSparse | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| *SubSkills-Categories3* | | | | | | | |
| TwoArmJacoUR5e | $0.85 \pm 0.01$ | $0.93 \pm 0.02$ | $0.94 \pm 0.01$ | $0.94 \pm 0.01$ | $0.99 \pm 0.01$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| TwoArmKinova3Jaco | $0.89 \pm 0.01$ | $0.96 \pm 0.02$ | $0.97 \pm 0.00$ | $0.96 \pm 0.01$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| TwoArmKinova3UR5e | $0.86 \pm 0.01$ | $0.93 \pm 0.01$ | $0.97 \pm 0.01$ | $0.94 \pm 0.02$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| TwoArmPandaJaco | $0.86 \pm 0.01$ | $0.95 \pm 0.03$ | $0.96 \pm 0.01$ | $0.94 \pm 0.01$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| TwoArmPandaKinova3 | $0.88 \pm 0.02$ | $0.92 \pm 0.02$ | $0.97 \pm 0.01$ | $0.96 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| TwoArmPandaUR5e | $0.84 \pm 0.01$ | $0.92 \pm 0.02$ | $0.94 \pm 0.03$ | $0.93 \pm 0.01$ | $0.97 \pm 0.01$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| *Real2Real* | | | | | | | |
| 4joint → 3joint | $0.78 \pm 0.01$ | $0.85 \pm 0.03$ | $0.86 \pm 0.08$ | $0.69 \pm 0.21$ | – | – | – |

Table 6: Full Regret computed over expert-normalized learning curves. Lower is better.

# D Details on the Real Robot Experimental Setup

This appendix specifies the real-robot domain of Section 4.4 and the settings behind Figure 7; Table 7 lists the hyperparameters.

**Hardware and task.** The robot is an SO-ARM101 follower arm: a 3D-printed arm with six Feetech STS3215 position-controlled servos (five joints and a gripper) on a single half-duplex serial bus, driven through LeRobot (Cadene et al., 2024). A camera rigidly mounted on the wrist streams $640 \times 480$ frames at 30 fps, and a $1680 \times 1050$ monitor in front of the arm shows the stimulus. At every reset a green disc of 5 cm diameter is drawn on a grey background at a position sampled uniformly at random, subject to the whole disc staying on screen. Green on grey was chosen after measuring the room: every bright saturated pixel in the workspace falls in the red–orange hue band, and a white background saturated the camera and washed out the dot. Environment, replay buffer and gradient updates all run on one laptop; nothing is simulated.

**Observations and actions.** The policy observes a $48 \times 48$ RGB image with a 3-frame stack, plus a proprioceptive vector of the six joint angles, their values at the previous decision, the six servo velocities, and the previous action, all scaled to $[-1, 1]$. The action is a position delta in $[-1, 1]$ per controlled joint; one unit moves a joint by 2.2 normalised units (of a $[-100, 100]$ calibrated range) per control tick, a limit that covers 99% of the steps in 30 episodes of human teleoperation, so the policy moves at roughly demonstration speed. The source domain controls `shoulder_pan`, `shoulder_lift`, `elbow_flex` and `wrist_flex` (a 22-dimensional proprioceptive vector, 4 actions); the target domain holds `wrist_flex` at its home value (21 and 3). `wrist_roll` and the gripper are held fixed in both. An episode is 200 control ticks at 20 Hz (10 s); it ends by truncation only.

**Reward.** A pixel with hue $h \in [0, 179]$, saturation $s$ and value $v \in [0, 255]$ counts as the dot when

$$M(x, y) = \big[40 \le h \le 85\big] \wedge \big[s \ge 80\big] \wedge \big[v \ge 120\big]; \tag{1}$$

the value threshold separates the backlit dot from the gripper fingers, which sit in every wrist frame. Detected pixels are weighted by centrality with the separable triangular kernel of UR5-VisualReacher (Yuan & Mahmood, 2022), where $W \times H$ is the frame size,

$$r = \min\left( \frac{c}{WH} \sum_{x,y} M(x, y)\, w_x(x)\, w_y(y),\ r_{\max} \right), \qquad w_x(x) = \frac{1}{2} - \frac{|x - W/2|}{W}, \tag{2}$$

with $w_y$ defined likewise over $H$, $c = 150$ and $r_{\max} = 4$. The reward is computed on the full $640 \times 480$ frame, not on the policy's $48 \times 48$ observation, and the clip stops the arm from being rewarded for pressing the lens into the screen.

**Autonomy and safety.** Runs are unattended, so the environment resets and protects itself. A reset interpolates the arm in joint space to a recorded home pose over 2 s and draws a new dot. A stall guard ends the episode when a servo reports load $\ge 400$ (in 0.1% of rated torque) for 10 consecutive ticks; contact-free motion stays below 290. An overheat guard ends it when a servo reads above 55°C on 5 consecutive one-second reads, with readings above 50°C discarded as corrupted bus packets rather than heat. Guard aborts are stored with `done` $= 0$, so the value bootstrap treats them as truncations. LeRobot's per-command clamp is set to 5.5 units, above the policy's 2.2, so that the policy and not the clamp sets the speed, and the arm is parked under torque on every exit.

**Learning algorithms.** The from-scratch baseline is SAC (Haarnoja et al., 2018) with a convolutional encoder trained by the critic loss only. The encoder is four $3 \times 3$ convolutions with 32 channels and strides $2, 2, 2, 1$, followed by a spatial softmax (Levine et al., 2016) that returns the expected pixel coordinate of every channel, giving 64 features; actor and critic are two-layer MLPs of width 512 on the features concatenated with the proprioceptive vector. Images are augmented with a random shift of 2 pixels (DrQ's 4 pixels at $84 \times 84$, scaled). A control tick is one 20 Hz command; with an action repeat of 2, one policy decision is held for two ticks and stored as one transition, with the reward summed over both ticks. The x-axis of Figure 7 counts ticks. FT-SAC loads the encoder, actor and critic of the source policy trained with the same seed, replaces the layers whose shape

changes (the actor's first and last, the critic's first) with freshly initialised ones, and trains as SAC. QAvatar-SAC initialises all target networks randomly, keeps the source critic frozen, and blends it with the target critic through the adaptive weight of Chen et al. (2026); its state and action decoders are constrained by RealNVP flows fitted on 50k transitions collected by replaying the source policy deterministically. CAT-SAC feeds the frozen source actor and critic laterally into the hidden layers of the target networks, gated per layer by a learned student–teacher share, and maps target states and actions into the source domain through the same flows, trained with a cycle-consistency and a mutual-information loss; we lower the weight of the mutual-information loss from the authors' 0.5 to 0.1, since at 0.5 the student switched its teachers off within 50k steps and then lost return on both seeds tried. All transfer methods use a source policy selected by its best 100-episode training mean; all other settings are those of SAC.

**Practical constraints.** There is one arm, so seeds run sequentially, with the arm resting for 10 minutes with torque off between runs; the 16 runs in Figure 7 are about 30 hours of robot time. A SAC gradient update costs about 9 ms and the environment about 9 ms of each 50 ms tick, so the update ratio is bounded by the control period; a method whose update is markedly more expensive lowers the control rate rather than raising an error. QAvatar-SAC and CAT-SAC, whose updates cost roughly three times SAC's on our laptop, ran at about 18 and 17 Hz instead of 20.

Table 7: Hyperparameters of the real-robot experiments. Settings marked † were chosen from measurements on this arm.

| | hyperparameter | value |
|---|---|---|
| environment | control rate | 20 Hz |
| | episode length | 200 ticks (10 s), truncation only |
| | training budget | 100k ticks per run |
| | image observation | $48 \times 48$ RGB, 3-frame stack |
| | proprioception | 22-d (source), 21-d (target) |
| | action scale† | 2.2 normalised units per tick |
| | reward | centre-weighted, $c = 150$, $r_{\max} = 4$ |
| | stall guard† | load $\geq 400$ for 10 ticks |
| | overheat guard† | $> 55°$C on 5 reads; $> 50°$C discarded |
| SAC | action repeat† | 2 ticks per decision |
| | updates per decision† | 2 |
| | random-action warmup | 2,000 ticks |
| | replay capacity | 100,000 transitions |
| | batch size | 128 |
| | learning rate (all networks) | $3 \times 10^{-4}$ |
| | discount | 0.99 |
| | target Polyak $\tau$ | 0.005 |
| | initial temperature | 0.1, auto-tuned |
| | actor and target update period | every 2 updates |
| | encoder | 4 convs $\times$ 32 channels, spatial softmax, 64 features |
| | MLP width | 512 |
| | random shift | 2 px |
| FT-SAC | source policy | same-seed 4-joint SAC, best 100-episode mean |
| | transferred | encoder, actor, critic; reshaped layers reset |
| QAvatar-SAC | source policy | 4-joint SAC, best 100-episode mean |
| | flow data | 50k transitions from the source policy |
| | decoder learning rate | $1 \times 10^{-4}$ |
| | weight update period | every 1,000 updates |
| CAT-SAC | source policy and flow data | as QAvatar-SAC |
| | mapping learning rate | $3 \times 10^{-4}$ |
| | mutual-information weight† | 0.1 (authors' default 0.5) |
| | student-preference anneal | 0 to 0.05 over 45–90% of training |