A GPU-PARALLEL FRAMEWORK FOR HETEROGENEOUS MULTI-TASK REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

GPU-parallel simulation provides abundant robot interaction, but existing benchmarks rarely combine this scale with heterogeneous manipulation tasks and standardized multi-task RL evaluation. We introduce Hebero (Heterogeneous Benchmark for Robot Learning), a GPU-parallel Isaac Lab benchmark that enables efficient joint training and evaluation of a single policy across all 40 heterogeneous tasks. Scaling experiments show that increasing parallel replicas per task improves success under a fixed wall-clock budget. To support learning with sparse rewards and limited demonstrations, we propose Demonstration-Guided Policy Optimization (DGPO), which reuses demonstrations for dense tracking rewards and asymmetric value learning. Its shared stack supports controlled comparisons of learner-specific demonstration interfaces within PPO. Within DGPO framework, we introduce IW-ABC, which uses a lightweight per-task learning progress signal to coordinate adaptive behavior cloning (ABC), relaxing demonstration guidance with task progress, and importance weighting (IW), emphasizing lagging tasks in PPO updates. With 50 demonstrations per task, IW-ABC achieves 90.1% state-input mean success, outperforming the strongest baseline FAMO-ABC by 7.8 percentage points. Its visual counterpart reaches 93.5% mean success. Real-world experiments further demonstrate that a single multi-task policy trained in simulation can successfully perform four tasks on a physical Piper robot. A preview of the benchmark, dataset and training code is provided in the supplementary materials.

1 INTRODUCTION

GPU-parallel reinforcement learning has made robot simulation a high-throughput training substrate, but much of this throughput is still spent on specialist policies: one objective, one policy, and one training run at a time. The next scaling question is whether this infrastructure can support *capability breadth*, where a single policy acquires many structured manipulation skills in one training process. Multi-task reinforcement learning (MTRL) offers a natural path because shared task structure can improve representation learning, exploration, and data efficiency (D'Eramo et al., 2020; Xu et al., 2024). In massively parallel robot RL, however, sample collection is no longer the only bottleneck; value stability, task imbalance, and the use of prior data become central algorithmic challenges (Tao et al., 2025; Joshi et al., 2025; Janwani et al., 2026).

Existing benchmarks provide complementary parts of this capability. RLBench offers diverse tasks, demonstrations, and multi-task learning settings on CPU-based physics (James et al., 2020); LIBERO adds structured suites for lifelong learning (Liu et al., 2023b). Later, MTBench brings Meta-World tasks to GPU-parallel state-based MTRL (Yu et al., 2020b; Joshi et al., 2025), while ManiSkill3 supports heterogeneous GPU simulation and visual RL with per-task reference learners (Tao et al., 2025). **Hebero** (**He**terogeneous **Be**nchmark for **Ro**bot Learning) combines these capabilities in one Isaac Lab (Mittal et al., 2025) benchmark: multiple MTRL baselines jointly learn all 40 heterogeneous tasks through GPU-parallel interaction, with state and visual inputs, demonstration trajectories, and a standardized evaluation protocol (fig. 1). A preview of the benchmark, training code and evaluation videos is provided in the supplementary materials.

Benchmark throughput alone does not solve sparse exploration or multi-task optimization. Where Meta-World relies on manually engineered task rewards (Yu et al., 2020b), we derive a reusable tracking reward from demonstrations. For multi-task balancing, PCGrad manipulates per-task gra-

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.

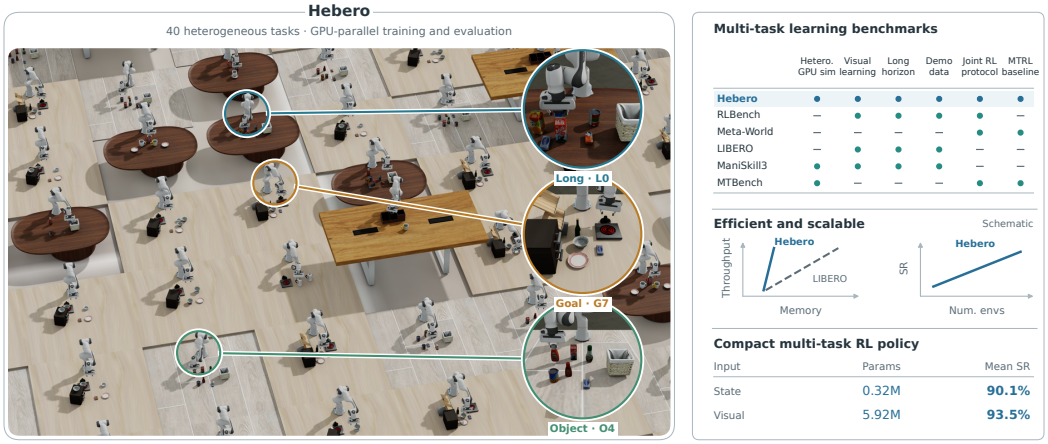

Figure 1: **Hebero: heterogeneous tasks, scalable reinforcement learning, and compact policies. Left:** Forty LIBERO tasks share one GPU-parallel simulation, with magnified views highlighting distinct manipulation scenes. **Right:** Benchmark capabilities, schematic efficiency and scaling trends, and compact multi-task RL policies trained with the DGPO framework.

dients (Yu et al., 2020a), while FAMO avoids this overhead through loss-based weighting (Liu et al., 2023a). We instead use a lightweight per-task learning progress signal, measured by per-task success-rate EMAs, to coordinate importance weighting (IW) toward lagging tasks and adaptive behavior cloning (ABC) that relaxes demonstration guidance with task progress and training time. The **Demonstration-Guided Policy Optimization (DGPO)** framework integrates these mechanisms with task-conditioned PPO (Schulman et al., 2017) and asymmetric value learning, using **IW-ABC** as its reference recipe. Its shared infrastructure enables fair comparisons of demonstration-guided RL algorithms under a common PPO backbone, reward, observation, and evaluation protocol. Reference learners vary how they use demonstrations, including BC initialization, DAPG-style regularization (Rajeswaran et al., 2018), and an RFCL-inspired reset curriculum (Tao et al., 2024).

Our experiments connect heterogeneous parallel simulation to joint learning with compact policies. The schematic curves in fig. 1 illustrate memory-efficient throughput scaling and improved success with more parallel environments. Across 40 tasks and 1,600 environments, Hebero delivers approximately 7.5k simulation steps per second on one L20 GPU, 3.4 times the throughput of our 96-core MuJoCo baseline. Increasing replicas per task improves success within a common wall-clock budget. Within the shared DGPO stack, IW-ABC achieves the highest state-input Mean SR of 90.1% with a 0.32M-parameter actor, outperforming the strongest baseline, FAMO-ABC (82.3%), by 7.8 percentage points. The visual policy reaches 93.5% with 5.92M total deployed parameters, including its frozen encoder. A separate jointly trained four-task policy transfers to a physical Piper robot with 82.5% success and no weight updates. Our contributions are threefold:

1. **Hebero: an efficient and scalable multi-task RL benchmark.** We compile all 40 LIBERO tasks into one heterogeneous GPU-parallel RL process with common state and visual interfaces, demonstrations, and evaluation protocols. GPU parallelism increases throughput, while scaling environments per task improves success under a fixed wall-clock budget. An anonymized preview of the benchmark code, together with data access and reproduction instructions, is provided in the supplementary materials.

2. **Demonstration-guided training framework and reference recipe.** DGPO reuses demonstrations for dense tracking rewards and asymmetric value learning, providing a shared training stack with IW-ABC as its reference recipe. Applied to four Piper tasks, this recipe trains a joint state-based policy that transfers to the physical robot achieving 82.5% success.

3. **Adaptive imitation guidance and task balancing.** Within DGPO, IW-ABC uses the same lightweight per-task learning progress signal to guide both imitation and online RL across heterogeneous tasks. Compared with FAMO-ABC, the strongest state-input baseline, it improves Mean SR from 82.3% to 90.1% and Long SR from 53.3% to 70.0%.

2 Related Work

2.1 GPU-Parallel Multi-Task Robot Learning

Early CPU-based benchmarks established diverse manipulation tasks and learning protocols. RLBench provided 100 tasks, including long-horizon compositions, with visual observations and motion-planned demonstrations for multi-task and few-shot learning (James et al., 2020). Contemporaneously, Meta-World defined joint multi-task and meta-RL evaluation over 50 tasks (Yu et al., 2020b), later standardized by Meta-World+ (McLean et al., 2025). LIBERO organized demonstration-rich tasks into object, spatial, goal, and long-horizon suites for lifelong learning (Liu et al., 2023b). Their original execution models expose useful task diversity but do not provide a single GPU-parallel process for thousands of heterogeneous environments. Subsequent GPU-accelerated systems scale simulation and learning: ManiSkill3 provides heterogeneous GPU physics and rendering (Tao et al., 2025), RoboVerse unifies simulation and data interfaces (Geng et al., 2025), and MTBench ports Meta-World to Isaac Gym for state-based MTRL (Joshi et al., 2025). Related directions include GPU-vectorized multi-objective RL in MO-Playground (Janwani et al., 2026) and massively multitask RL on MMBench with the Newt world model (Hansen et al., 2026). Hebero brings 40 LIBERO tasks to a single heterogeneous GPU-parallel process for demonstration-guided joint RL, preserving their assets and success predicates while supporting state and visual policies under common training and evaluation interfaces.

2.2 Demonstration-Guided and Multi-Task Reinforcement Learning

Prior data can influence policy initialization, online optimization, the training-state distribution, or the reward signal. DAPG uses demonstrations to initialize and regularize policy learning (Rajeswaran et al., 2018), while RLPD incorporates offline data into online off-policy RL (Ball et al., 2023) and Rainbow-DemoRL combines improvements to demonstration-augmented RL (Bhatt et al., 2026). Data-Efficient Multitask DAgger allocates additional expert data according to task progress (Fu et al., 2025). RFCL instead uses demonstration states to construct a reverse-to-forward reset curriculum (Tao et al., 2024). Reference tracking provides dense imitation-based rewards (Peng et al., 2018), while manipulation methods learn reusable dense rewards, combine demonstration-based reward and world-model learning, or estimate residual and interpretable rewards (Mu et al., 2024; Escoriza et al., 2025; Cao et al., 2025; Baimukashev et al., 2024). Object-centric and reward-observation world models provide further training signals (Ferraro et al., 2023; Tang et al., 2025); adversarial imitation and multi-task inverse RL offer complementary ways to infer task objectives (Kuang et al., 2025; Glazer et al., 2025). DGPO organizes these different uses of demonstrations through a shared trainability stack and additional learner-specific interfaces.

Multi-task optimization introduces a separate challenge: shared representations and task diversity can improve transfer and exploration (D'Eramo et al., 2020; Xu et al., 2024), yet tasks may progress unevenly and produce conflicting updates. Prior methods balance gradient magnitudes (Chen et al., 2018), modify gradient geometry (Yu et al., 2020a; Liu et al., 2021), or adapt task weights through loss dynamics (Liu et al., 2023a). PolicyGradEx estimates task affinities to group objectives for joint training (Zhang et al., 2026), while pessimistic offline RL addresses the risks of sharing data across task distributions (Bai et al., 2024). Our task importance weighting reweights PPO losses using lightweight online task-success statistics under fixed rollout allocation. Its signal is independent of reward scale and separate from demonstration interfaces, allowing composition with the supported PPO references. IW-ABC combines task prioritization with dynamic teacher guidance during online PPO training, adapting the strength of teacher supervision to each task's learning progress and overall training progress.

3 The Hebero Benchmark

We separate the benchmark definition from the demonstration-guided reference learning framework. Hebero specifies a multi-task RL problem, a GPU-parallel environment construction, and a standardized interaction and evaluation contract. DGPO, introduced in section 4, instantiates one demonstration-guided approach to this benchmark using the accompanying demonstration trajectories.

3.1 Problem Setting

We follow the standard multi-task RL formulation (Yu et al., 2020b), in which each task T_k is a finite-horizon MDP $(\mathcal{S}_k, \mathcal{A}, P_k, R_k, H, \gamma)$ drawn from a task distribution $p(T)$; notation is summarized in appendix A. The tasks may differ in scene geometry, transition dynamics, reward, and success predicate, but share an action space, control horizon, and robot embodiment. Multi-task RL learns one task-conditioned policy $\pi_\theta(a_t \mid o_t)$, where o_t includes the task encoding z_k, to maximize average discounted return $\max_\theta \mathbb{E}_{T_k \sim p(T)}[\mathbb{E}_{\pi_\theta}[\sum_{t=0}^{H-1} \gamma^t R_k(s_t, a_t)]]$. For Hebero, $p(T)$ is uniform over $K = 40$ tasks indexed by $k \in \{0, \ldots, K-1\}$. We evaluate the single policy on these same tasks, rather than on a held-out task set.

3.2 Benchmark Construction

Hebero separates task semantics from simulator execution. Offline task descriptors specify scene assets, reset rules, and success predicates. Shared asset prototypes are instantiated in the environments that use them, with one batched view per prototype. An environment-to-task map routes task-specific operations within one Isaac Lab simulator, renderer, rollout buffer, and policy update. This heterogeneous GPU-parallel design and implementation enables efficient joint RL training across all 40 tasks, with throughput and wall-clock learning gains quantified in section 5.1. Further construction details are provided in appendix A.1.

3.3 Standardized Learning Interface

Hebero exposes the same observation, action, reward, and success interfaces for every task, giving multi-task learners a common control contract across heterogeneous scenes.

Action space. At each step, the policy outputs a normalized action $a_t \in [-1, 1]^7$. The first six dimensions specify relative end-effector translation and rotation through an operational-space controller, and the last dimension controls the binary gripper. Exact action scaling and controller parameters are given in appendix A.2.

Actor observations. All actors receive proprioception o_t^{proprio}, the previous action a_{t-1}, and a task encoding z_k. State actors additionally receive a deployable object–target pose representation $o_t^{\text{object-target}}$, whereas visual actors replace this representation with pooled frozen-ViT features h_t^{visual} from third-person and wrist RGB observations:

$$\begin{aligned} o_t^{\text{state}} &= \left[a_{t-1}, z_k, o_t^{\text{proprio}}, o_t^{\text{object-target}}\right], \\ o_t^{\text{visual}} &= \left[a_{t-1}, z_k, o_t^{\text{proprio}}, h_t^{\text{visual}}\right]. \end{aligned} \tag{1}$$

Full observation dimensions, task encodings, and visual architecture are given in appendices A.3 and B.1.

Task reward. Every task exposes learner-independent task reward r_t^{task} composed of three terms: a goal reward r_t^{goal} where $g_k(s_t)$ indicates task success, smoothness and control regularizers r_t^{reg} where $\dot{q}_t^{\text{arm}}$ denotes arm-joint velocity, and joint-limit penalties r_t^{safe}:

$$\begin{aligned} r_t^{\text{task}} &= r_t^{\text{goal}} - r_t^{\text{reg}} - r_t^{\text{safe}}, \\ r_t^{\text{goal}} &= w_{\text{succ}} \mathbf{1}[g_k(s_t)], \\ r_t^{\text{reg}} &= w_{\Delta a}\|a_t - a_{t-1}\|^2 + w_a\|a_t\|^2 + w_{\dot{q}}\|\dot{q}_t^{\text{arm}}\|^2, \\ r_t^{\text{safe}} &= w_{\text{pos}}^{\text{lim}} \sum_j d_{j,t}^{\text{pos}} + w_{\text{vel}}^{\text{lim}} \sum_j d_{j,t}^{\text{vel}}. \end{aligned} \tag{2}$$

Detailed joint-limit penalties, and coefficient values are given in appendix A.4.

3.4 Evaluation Protocol

The benchmark evaluates placement robustness under a fixed, learner-independent reset distribution. For each task and movable object, we compute coordinate-wise bounds $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ from the 50 recorded demonstration starts. Each evaluation episode independently redraws the object's x and y coordinates uniformly within this box rather than replaying a recorded placement. Collision-based rejection sampling is applied to the sampled initial placements to ensure that objects do not collide with each other. Compared with LIBERO's fixed evaluation starts (Liu et al., 2023b), resampling better tests placement robustness rather than trajectory memorization. We report **mean success rate (Mean SR)** across all tasks and suite-level SR, using **Long SR** as the long-horizon measure. **Task coverage** is the number of tasks reaching at least 80% success and measures the breadth of reliably solved skills. **SR-AUC** is the normalized trapezoidal area under checkpoint-evaluated Mean SR as a function of PPO iteration over the shared training horizon, reported as a percentage of the maximum possible area. It reflects learning speed and overall success.

4 DGPO: A Demonstration-Guided Multi-Task Training Framework

GPU-parallel simulation accelerates interaction, but learning across Hebero's 40 heterogeneous tasks remains challenging under sparse goal rewards. We therefore introduce DGPO, a training framework that leverages the available demonstrations to make joint learning practical through shared reward, reset, and critic guidance coupled to task-conditioned PPO.

4.1 Shared Training Components

All online reference configurations use the same demonstration command stream, tracking reward, privileged critic inputs, task-conditioned PPO backbone and network capacity. Together these components form the shared DGPO trainability stack which supports controlled comparisons of learner-specific demonstration interfaces under matched training conditions.

Demonstration-trajectory command stream. The command stream centralizes demonstration bookkeeping across heterogeneous parallel environments. It maintains each environment's task identity, active demonstration, and timestep cursor, and caches the corresponding robot, object, gripper, and action references from a shared trajectory bank. These synchronized views let the tracking reward, critic, and learner interfaces reuse the same references without duplicating trajectory storage or cursor management.

Demonstration-tracking reward. Reference-based manipulation requires preserving robot–object and robot–environment interactions, as emphasized by OmniRetarget (Yang et al., 2026). We therefore track both robot motion and task-relevant world states, including rigid-object poses and articulation joint positions. Let $\mathcal{E}_k^{\text{robot}}$ contain task-k end-effector pose, arm-joint position and velocity, and gripper-state errors; let $\mathcal{E}_k^{\text{world}}$ contain rigid-object pose and articulation joint-position errors; and define $\mathcal{E}_k = \mathcal{E}_k^{\text{robot}} \cup \mathcal{E}_k^{\text{world}}$. For component error e_t^m, weight w_m, and component-specific bandwidth $\sigma_{r,m} > 0$, the demonstration-tracking reward is

$$r_t^{\text{demo}} = \sum_{m \in \mathcal{E}_k} w_m \exp\left(-\frac{\|e_t^m\|}{\sigma_{r,m}} \right), \tag{3}$$

and the reference online configuration adds this term to the task reward in eq. (2). The dense terms reward both how the robot moves and what its motion accomplishes in the world. Tracking weights w_m and component-specific kernel bandwidths are listed in table 4.

Asymmetric critic observations. The state critic augments the state actor input as $o_t^V = [o_t^{\text{state}}, e_t^{\text{robot}}, e_t^{\text{world}}, \phi_t]$. Here e_t^{robot} contains privileged end-effector, arm-joint position, and gripper tracking errors; e_t^{world} contains object-pose and articulation-position tracking errors; and ϕ_t is the normalized current demonstration cursor (appendix eq. (11)). The visual critic input is $o_t^V = [a_{t-1}, z_k, o_t^{\text{proprio}}, h_t^{V,\text{visual}}, e_t^{\text{robot}}, e_t^{\text{world}}, \phi_t]$, where $h_t^{V,\text{visual}}$ uses a separate trainable attention pool over the same frozen image tokens (appendix B.1).

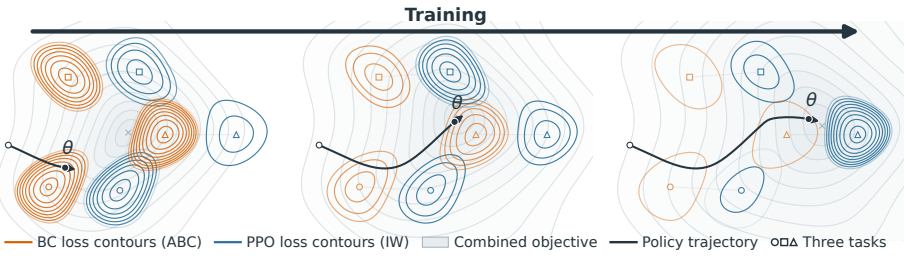

Figure 2: **Adaptive guidance and task balancing.** In a schematic parameter space, orange and blue contours show weighted per-task BC and PPO losses; matching marker shapes identify the same task. From early to late training, ABC guidance recedes at task-dependent rates while IW redistributes PPO emphasis, changing the combined objective (gray). The dark path illustrates how gradient updates move the shared policy θ as the combined objective evolves.

4.2 Reference Learners and Demonstration Interfaces

DGPO holds the shared components in section 4.1 fixed while varying the demonstration interface shown in the map; PPO adds no further demonstration mechanism. Our RFCL entry applies an RFCL-inspired reset curriculum to on-policy PPO. Reference-learner objectives, reset procedures, and settings are given in appendix B.5. We next specify IW-ABC for the state and visual configurations.

Learner family	Demonstration Entry
PPO	none
BC→PPO	offline actor initialization
DAPG	offline likelihood regularizer
ABC	live time-aligned action target
RFCL	demo-state reset curriculum

4.3 IW-ABC Reference Recipe

Demonstration-guided multi-task learning faces two coupled challenges: BC and PPO can favor competing policy updates, while tasks sharing one policy learn at different rates. IW-ABC uses a single lightweight learning progress signal to regulate both imitation pressure and task priority: ABC relaxes action guidance as task success and training time increase, while IW gives greater PPO weight to tasks lagging behind. Figure 2 illustrates how these changing weights reshape the combined objective for a shared policy, motivating the ABC and IW mechanisms below.

Task success as a progress signal. For each task k, the first batch of completed training episodes initializes $\tau_k \leftarrow \overline{\mathrm{SR}}_k$; subsequent batches update

$$\tau_k \leftarrow (1-\alpha_{\mathrm{ema}})\tau_k + \alpha_{\mathrm{ema}}\,\overline{\mathrm{SR}}_k, \tag{4}$$

Here $\tau_k \in [0,1]$ is task k's learning progress with update rate α_{ema}, and $\overline{\mathrm{SR}}_k$ is the mean binary success of the newly completed episodes. Computed directly from training rollouts, this scalar provides a lightweight proxy for per-task learning progress without additional evaluation (appendix B.3).

Adaptive action guidance. Let u count vectorized rollout steps, advancing once per control step across all parallel environments, and let U_{ann} be the annealing horizon. With annealing fraction $\rho_u = \min(u/U_{\mathrm{ann}}, 1)$, the coefficient is

$$\beta_{k,u} = \beta_{\min} + (\beta_{\max} - \beta_{\min})\big[(1-\tau_k)(1-\rho_u)\big]^{\kappa}. \tag{5}$$

An early-solved task can therefore relinquish imitation before the global schedule ends, while time annealing prevents a persistently difficult task from remaining indefinitely at maximum guidance. For rollout samples $(o_t, a_t^\star)$ with task index k, ABC adds

$$\mathcal{L}_{\mathrm{ABC}} = c_{\mathrm{BC}}\mathbb{E}_{t,k}\left[\beta_{k,u}\|\mu_\theta(o_t) - a_t^\star\|^2\right], \tag{6}$$

where μ_θ is the Gaussian policy mean and $a_t^\star$ is the demonstration action at the environment's current trajectory cursor. The exact target construction is given in appendix B.2.

Table 1: **Multi-task parallel simulation and RL training throughput (SPS).** Approximate MT-Bench results are from Joshi et al. (2025); hardware and peak memory were not reported.

Benchmark	Engine (hardware)	Tasks	Envs	Throughput	Memory
Simulation / collection					
Hebero All-40	Isaac Lab (1×L20)	40	1,600	7,500	10.6 GiB VRAM
LIBERO All-40	MuJoCo (96C/192T CPU)	40	1,600	2,200	1.5 TiB host PSS
End-to-end RL training					
Hebero All-40	Isaac Lab (8×L20)	40	25,600	78,500	144.9 GiB VRAM
MTBench MT50-rand	Isaac Gym (N/A)	50	24,576	69,000	N/A

Task balancing. Balanced collection does not ensure balanced learning: easy and persistently difficult tasks otherwise receive equal loss weight. For initialized tasks $\mathcal{K}_{\text{init}}$, we define the mean success $\bar{\tau}$, a sigmoid-transformed relative-success score p_k^{IW}, and the importance weight w_k:

$$\bar{\tau} = \frac{1}{|\mathcal{K}_{\text{init}}|} \sum_{j \in \mathcal{K}_{\text{init}}} \tau_j, \qquad p_k^{\text{IW}} = \sigma(s_{\text{IW}}(\tau_k - \bar{\tau})), \qquad w_k = w_{\max}(1 - p_k^{\text{IW}}) + w_{\min} p_k^{\text{IW}}. \quad (7)$$

where σ is the logistic sigmoid and $s_{\text{IW}} > 0$ controls the sensitivity to relative task success. Tasks below the multi-task mean receive larger weights. IW-ABC minimizes the combined objective using minibatch-normalized weights $\tilde{w}_k$ (appendix B.3):

$$\mathcal{L}^{\text{IW-ABC}} = \mathbb{E}_{t,k}\left[\tilde{w}_k \left(\ell_t^{\pi} + c_V \ell_t^{V} - c_H \mathcal{H}_t\right)\right] + \mathcal{L}_{\text{ABC}}, \quad (8)$$

where ℓ_t^{π} is the negative clipped PPO surrogate, ℓ_t^{V} is the clipped value-regression loss, and $\mathcal{H}_t$ is policy entropy. c_V and c_H are the value and entropy coefficients. Task IW scales all three PPO terms (ablation in appendix table 5). Detailed hyperparameters are given in table 4.

FAMO (Liu et al., 2023a) offers another lightweight approach to task balancing, adapting task weights from loss progress. Following FAMO's original actor–critic setting, we apply its task weighting only to critic losses and retain ABC, yielding FAMO-ABC (appendix B.4). We compare IW-ABC and FAMO-ABC under the same training budget in section 5.2.

5 Experiments

5.1 Scaling Efficiency of the Hebero RL Benchmark

High throughput with a practical memory footprint. Table 1 shows that Hebero supports higher simulation throughput than the evaluated MuJoCo configuration with a practical GPU memory footprint, and scales to multi-GPU end-to-end training across all 40 tasks.

Parallel scaling improves joint learning efficiency. Scaling the number of parallel environments translates simulation throughput into better joint learning: on a single 48 GB L20 GPU, more replicas per task yield higher success within the same wall-clock budget (fig. 3(a)) and reach matched reward levels sooner (fig. 3(b)). Thus, Hebero's parallel design enables faster learning across all 40 heterogeneous tasks.

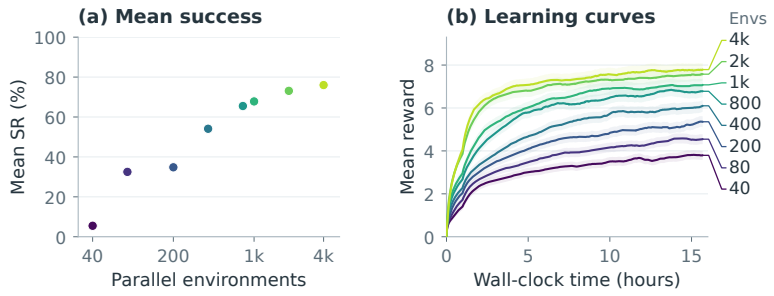

Figure 3: **Single-GPU parallel environment scaling under a fixed wall-clock budget.** (a) Mean SR across all 40 tasks, evaluated at the common training-budget endpoint as parallel environments increase. (b) Mean training reward versus wall-clock training time for each environment count.

5.2 Controlled Learner Comparison within DGPO

All learners in Table 2 jointly train on 40 tasks for 30,000 online PPO iterations with the same 50 demonstrations per task and undergo unified evaluation protocol (section 3.4). State-input learners share a compact task-conditioned $[512, 256, 128]$ actor–critic. We train each configuration with three training seeds; each run uses eight NVIDIA L20 GPUs with 2,000 parallel environments per GPU and takes approximately two days. Detailed reproducibility settings are provided in table 4 in the appendix.

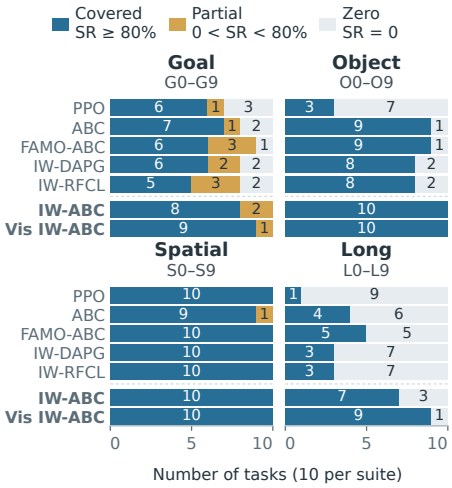

Figure 4: **Task coverage by suite.** Each bar partitions ten tasks by final SR.

Reference learner comparison. ABC improves all metrics over BC→PPO, highlighting the value of online adaptive demonstration guidance beyond offline BC initialization. With task IW fixed, IW-ABC outperforms IW-DAPG and IW-RFCL across all four metrics, supporting adaptive matched-action guidance over offline demonstration regularization and reverse-reset curricula for demonstration-guided joint learning. Adding IW to PPO trades lower Mean SR for higher Long SR, whereas pairing it with ABC improves both. With comparable computational complexity and training throughput, IW-ABC outperforms FAMO-ABC under the same training budget, showing the benefit of success-based IW as a simple, computationally efficient task-balancing mechanism when paired with ABC. Figure 4 shows how these gains extend across task suites rather than only raising average success. ABC, IW-ABC, and Vis IW-ABC cover 4, 7, and 9 of the ten Long tasks, demonstrating broader coverage of long-horizon skills.

Table 2: **PPO-based reference learner results.** All rows share DGPO framework; demonstration interfaces and task weighting vary.

Learner	IW	Mean SR (%)	Long SR (%)	Task coverage	SR-AUC (%)
PPO	×	50.8±3.1	10.0±0.0	20/40	45.2
BC→PPO	×	47.5±2.6	20.0±0.0	18/40	40.8
ABC	×	75.5±3.6	40.0±0.0	29/40	71.9
FAMO-ABC	×	82.3±2.9	53.3±5.8	30/40	75.6
IW-PPO	✓	44.9±2.5	20.0±0.0	18/40	41.3
IW-DAPG	✓	68.5±3.0	30.0±0.0	27/40	62.4
IW-RFCL	✓	66.4±2.8	30.0±0.0	26/40	60.1
IW-ABC	✓	**90.1±3.8**	**70.0±10.0**	**35/40**	**82.1**
Vis IW-ABC	✓	**93.5±2.6**	**81.9±11.5**	**38/40**	**83.3**

5.3 Learning Dynamics of IW-ABC

We examine the training dynamics of Vis IW-ABC to assess whether demonstration guidance recedes with task progress while PPO updates remain focused on lagging tasks; fig. 5 traces task success alongside IW and BC coefficients.

Prioritizing lagging tasks. IW reweights PPO losses under fixed rollout counts and optimizer steps. Most Object and Spatial tasks succeed early, whereas the harder Goal and Long tasks learn more slowly and require greater optimization emphasis. IW assigns these lagging tasks larger weights, amplifying their contribution to the shared PPO gradient until their success approaches the multi-task mean.

Annealing BC while retaining task priority. Once $u \geq U_{\text{ann}}$, all BC coefficients reach the residual floor $\beta_{\min}$, relaxing constraints on deviation from demonstration actions while IW continues

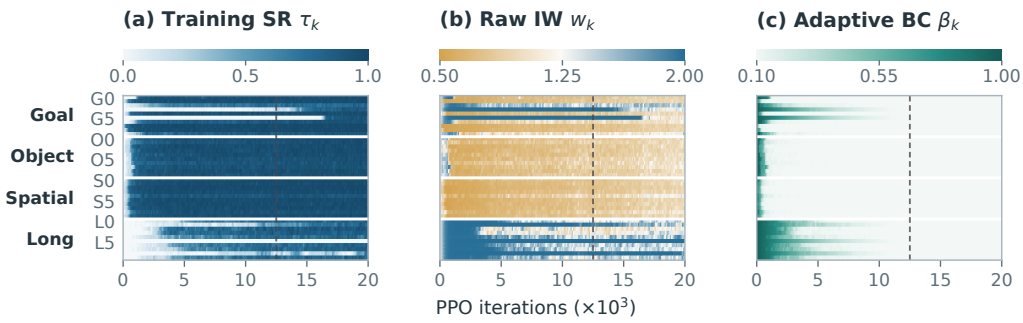

Figure 5: **Task-wise weighting dynamics of Vis IW-ABC.** Training SR EMAs (a) and reconstructed raw IW (b) and BC (c) coefficients from a single training run, averaged in 100-iteration bins. Rows retain task IDs 0–9 within each suite. Dashed lines mark the end of ABC annealing.

prioritizing lagging tasks (Fig. 5; see also Appendix Fig. 9). Together, ABC regulates how strongly each task follows demonstrations, while IW directs PPO updates toward tasks that still need improvement, coupling a gradual release from imitation with continued support for harder tasks.

5.4 Multi-Task Deployment on a Physical Robot

To evaluate IW-ABC's real-world deployability within DGPO, we jointly train one policy on four representative RoboTwin Piper tasks (Mu et al., 2025), selected for diverse interactions beyond repeated pick-and-place variants (fig. 6). Training uses parallel Isaac Lab replicas and 50 demonstration trajectories per task. Since our focus is not visual-policy sim-to-real transfer, we deploy the state-input policy on the physical Piper robot using an external model-based perception pipeline (appendix E), with policy weights fixed throughout deployment. Across 20 physical trials per task, the policy achieves an overall success rate of 82.5%. Figure 6 shows representative execution sequences and per-task success counts. Joint simulation training thus yields a deployable multi-task policy.



Figure 6: **Real-world deployment of one jointly trained four-task policy.** The simulation-trained state-based policy is deployed without weight updates on Piper robot. Rows show the four tasks with success counts over 20 trials each; columns show episode progress. Overall success is 82.5%.

6 Conclusion and Limitations

Hebero supports joint GPU-parallel RL on all 40 LIBERO tasks; increasing replicas per task improves learning within a fixed wall-clock budget. DGPO provides a shared demonstration-guided training stack for comparing PPO-based learners. IW-ABC adapts teacher guidance and task weights using a cheap learning progress signal, improving mean and long-horizon success. One jointly trained state-input policy achieves 82.5% success on four physical Piper tasks without weight updates. However, IW-ABC's time-indexed action targets can misalign under trajectory drift despite the tracking reward, and budget constraints limit training and evaluation to LIBERO and RoboTwin. Future work will extend the benchmark construction methodology and DGPO training framework to more challenging tasks and larger task sets.

AI USE STATEMENT

Generative AI tools assisted with experimental design, software implementation and debugging, result analysis and visualization, literature search, and manuscript editing. The authors made all research decisions, ran the experiments, and verified AI-assisted artifacts against source code, experiment logs, and cited literature. AI generated no training or evaluation data and no experimental measurements. The authors take full responsibility for the final work.

REPRODUCIBILITY STATEMENT

Sections 3.4 and 5.2 specify the training and evaluation protocols. Implementation details and hyperparameters are provided in appendices A and B and table 4. The supplementary materials include an anonymized preview of the Hebero training and evaluation code, pinned dependencies, and instructions for accessing the demonstration data and running experiments. We will open-source the complete codebase and release the datasets and reproduction documentation after the review process.

REFERENCES

Chenjia Bai, Lingxiao Wang, Jianye Hao, Zhuoran Yang, Bin Zhao, Zhen Wang, and Xuelong Li. Pessimistic value iteration for multi-task data sharing in offline reinforcement learning. *Artificial Intelligence*, 326:104048, January 2024. ISSN 0004-3702. doi: 10.1016/j.artint.2023.104048. URL http://dx.doi.org/10.1016/j.artint.2023.104048.

Daulet Baimukashev, Gokhan Alcan, Kevin Sebastian Luck, and Ville Kyrki. Learning transparent reward models via unsupervised feature selection. In *8th Annual Conference on Robot Learning*, 2024.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1577–1594. PMLR, 23–29 Jul 2023.

Dwait Bhatt, Shih-Chieh Chou, and Nikolay Atanasov. Rainbow-DemoRL: Combining improvements in demonstration-augmented reinforcement learning. *arXiv preprint arXiv:2603.27400*, 2026. URL https://arxiv.org/abs/2603.27400.

Chenyang Cao, Miguel Rogel-García, Mohamed Nabail, Xueqian Wang, and Nicholas Rhinehart. Residual reward models for preference-based reinforcement learning. *CoRR*, abs/2507.00611, July 2025. URL https://arxiv.org/abs/2507.00611v1.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 794–803. PMLR, 2018. URL https://proceedings.mlr.press/v80/chen18a.html.

Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Adrià López Escoriza, Nicklas Hansen, Stone Tao, Tongzhou Mu, and Hao Su. Multi-stage manipulation with demonstration-augmented reward, policy, and world model learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 15542–15563. PMLR, 13–19 Jul 2025.

Stefano Ferraro, Pietro Mazzaglia, Tim Verbelen, and Bart Dhoedt. FOCUS: Object-centric world models for robotic manipulation. In *Intrinsically-Motivated and Open-Ended Learning Workshop @NeurIPS2023*, 2023.

Haotian Fu, Ran Gong, Xiaohan Zhang, Maria Vittoria Minniti, Jigarkumar Patel, and Karl Schmeckpeper. Data-efficient multitask DAgger. arXiv preprint arXiv:2509.25466, 2025. URL https://arxiv.org/abs/2509.25466.

Haoran Geng, Feishi Wang, Songlin Wei, Yuyang Li, Bangjun Wang, Boshi An, Haozhe Lou, Charlie Tianyue Cheng, Peihao Li, Haozhe Chen, Yutong Liang, Yuxi Qian, Jiageng Mao, Weikang Wan, Yiran Geng, Mingtong Zhang, Jiangran Lyu, Siheng Zhao, Jiazhao Zhang, Chaoyi Xu, Jialiang Zhang, Chengyang Zhao, Haoran Lu, Yufei Ding, Ran Gong, Yuran Wang, Yuxuan Kuang, Ruihai Wu, Baoxiong Jia, Hao Dong, Siyuan Huang, Yue Wang, Jitendra Malik, and Pieter Abbeel. RoboVerse: A Unified Platform, Benchmark and Dataset for Scalable and Generalizable Robot Learning. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.022.

Neta Glazer, Aviv Navon, Aviv Shamsian, and Ethan Fetaya. Multi task inverse reinforcement learning for common sense reward. arXiv preprint arXiv:2402.11367, 2025. URL https://arxiv.org/abs/2402.11367.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control. In *The Fourteenth International Conference on Learning Representations*, 2026.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020. doi: 10.1109/LRA.2020.2974707. URL https://doi.org/10.1109/LRA.2020.2974707.

Neil Janwani, Ellen Novoseller, Vernon J. Lawhern, and Maegan Tucker. MO-Playground: Massively parallelized multi-objective reinforcement learning for robotics. arXiv preprint arXiv:2603.09237, 2026. URL https://arxiv.org/abs/2603.09237.

Viraj Joshi, Zifan Xu, Bo Liu, Peter Stone, and Amy Zhang. Benchmarking massively parallelized multi-task reinforcement learning for robotics tasks. arXiv preprint arXiv:2507.23172, 2025. URL https://arxiv.org/abs/2507.23172.

Yingyi Kuang, Luis J. Manso, and George Vogiatzis. Goal-based self-adaptive generative adversarial imitation learning (Goal-SAGAIL) for multi-goal robotic manipulation tasks. arXiv preprint arXiv:2506.12676, 2025. URL https://arxiv.org/abs/2506.12676.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18878–18890, 2021.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. FAMO: Fast adaptive multitask optimization. arXiv preprint arXiv:2306.03792, 2023a. URL https://arxiv.org/abs/2306.03792.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023b.

Reginald McLean, Evangelos Chatzaroulas, Luc McCutcheon, Frank Röder, Tianhe Yu, Zhanpeng He, K.R. Zentner, Ryan Julian, J K Terry, Isaac Woungang, Nariman Farsad, and Pablo Samuel Castro. Meta-World+: An improved, standardized, RL benchmark. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning. arXiv preprint arXiv:2511.04831, 2025. URL https://arxiv.org/abs/2511.04831.

Tongzhou Mu, Minghua Liu, and Hao Su. DrS: Learning reusable dense rewards for multi-stage tasks. In *The Twelfth International Conference on Learning Representations*, 2024.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, Lunkai Lin, Zhiqiang Xie, Mingyu Ding, and Ping Luo. RoboTwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27649–27660, June 2025.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37 (4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL http://doi.acm.org/10.1145/3197517.3201311.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.049.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017. URL https://arxiv.org/abs/1707.06347.

Jinghuan Shang, Karl Schmeckpeper, Brandon B. May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. In *8th Annual Conference on Robot Learning*, 2024.

Yinzhou Tang, Yu Shang, Yinuo Chen, Bingwen Wei, Xin Zhang, Shu'ang Yu, Liangzhi Shi, Chao Yu, Chen Gao, Wei Wu, and Yong Li. RoboScape-R: Unified reward-observation world models for generalizable robotics training via RL. arXiv preprint arXiv:2512.03556, 2025. URL https://arxiv.org/abs/2512.03556.

Stone Tao, Arth Shukla, Tse-kai Chan, and Hao Su. Reverse forward curriculum learning for extreme sample and demo efficiency. In *The Twelfth International Conference on Learning Representations*, 2024.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-Kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Demonstrating GPU Parallelized Robot Simulation and Rendering for Generalizable Embodied AI with ManiSkill3. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.021.

Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. FoundationPose: Unified 6D pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17868–17879, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Wen_FoundationPose_Unified_6D_Pose_Estimation_and_Tracking_of_Novel_Objects_CVPR_2024_paper.html.

Ziping Xu, Zifan Xu, Runxuan Jiang, Peter Stone, and Ambuj Tewari. Sample efficient myopic exploration through multitask reinforcement learning with diverse tasks. In *The Twelfth International Conference on Learning Representations*, 2024.

Lujie Yang, Xiaoyu Huang, Zhen Wu, Angjoo Kanazawa, Pieter Abbeel, Carmelo Sferrazza, C. Karen Liu, Rocky Duan, and Guanya Shi. OmniRetarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026. URL https://arxiv.org/abs/2509.26633.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020a.

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 Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the*

Conference on Robot Learning, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 30 Oct–01 Nov 2020b.

Zhenshuo Zhang, Minxuan Duan, Youran Ye, and Hongyang R. Zhang. Scalable multi-objective and meta reinforcement learning via gradient estimation. arXiv preprint arXiv:2511.12779, 2026. URL https://arxiv.org/abs/2511.12779.

A HEBERO BENCHMARK SPECIFICATION AND IMPLEMENTATION

This section specifies the benchmark artifact independently of any particular learner: task compilation, heterogeneous simulation, controller, observation, reward, success, and randomized evaluation interfaces. The accompanying DGPO framework and reference learners are documented separately in appendix B.

Problem notation. In section 3.1, task T_k has state space $\mathcal{S}_k$, shared action space $\mathcal{A}$, transition kernel P_k, reward function R_k, episode horizon H in control steps, and discount factor γ. The distribution $p(T)$ samples tasks, indexed by k, from the K-task set. The policy parameters are θ; s_t, o_t, and a_t denote simulator state, actor observation, and action at control step t, while z_k is the task encoding. The inner expectation in the return objective averages over policy-induced trajectories for the sampled task.

A.1 TASK COMPILATION AND HETEROGENEOUS SIMULATION

Offline task descriptors. Before training, each LIBERO BDDL task is compiled into a cached JSON descriptor containing its instruction, scene, assets, spawn regions, goal predicates, and thresholds. The descriptor governs task encoding, observations, resets, success checks, demonstration lookup, and logging; the original BDDL path is retained for provenance.

Goal predicates. Relational predicates (e.g., `in`, `on`), articulation predicates (e.g., `open`, `close`), and activation predicates (e.g., `turn_on`) are evaluated from live simulator state. Task success is their conjunction and supplies the common sparse-reward event and evaluation outcome.

Asset pipeline. MuJoCo MJCF assets are converted offline to cached USD files while preserving geometry, appearance, dynamics, joints, and articulation limits. Runtime descriptors resolve objects and fixtures to shared assets while retaining task-specific placement and randomization.

Heterogeneous vectorized simulation. Each environment instantiates the assets specified by its assigned task, while all environments share one simulator and renderer and expose a rectangular interaction batch to a single rollout and update pipeline. For K tasks and g environments per task, the total environment count is $N = Kg$; the current implementation assigns environment $i \in \{0, \dots, N-1\}$ to task $i \bmod K$. This environment-to-task map routes reset rules, demonstration trajectories, goal predicates, and statistics for both state and visual policies.

Shared asset views. Each distinct asset prototype has one batched simulator view spanning exactly the environments in which it is instantiated. Prototypes are shared across tasks and suites by asset model, scale, and physical configuration, with separate instance slots when a scene contains multiple copies of the same model. Static fixtures additionally distinguish their fixed poses. A view can cover a noncontiguous subset of the full environment batch; cached mappings between global environment IDs and local view indices support state gathering and reset writes. Task-specific bindings retain each object's initial pose and semantic role, while observations are assembled into the common environment batch.

Task-set configuration. The canonical benchmark contains ten tasks from each of Goal, Object, Spatial, and Long. The figure labels Gi, Oi, Si, and Li preserve the logged within-suite index $i \in \{0, \dots, 9\}$ for `libero_goal`, `libero_object`, `libero_spatial`, and `libero_10`, respectively. The active task set can be restricted for focused experiments without changing the canonical task registry. Focused encoding restricts the task encoding and object–target pose buffer to the active subset, reducing the input dimension for single-suite and ablation runs. Canonical encoding preserves the full benchmark dimensionality, allowing checkpoints to be loaded across task subsets without an input-dimension mismatch.

A.2 Action and Controller Interface

The normalized action $a_t \in [-1, 1]^7$ commands relative end-effector motion through an operational-space controller:

$$\Delta p_t = s_p a_{t,1:3}, \qquad \Delta R_t = \mathrm{Exp}\left(s_R a_{t,4:6}\right), \tag{9}$$

where s_p and s_R scale translation and axis-angle rotation, respectively. The seventh component opens the gripper when positive and closes it otherwise. Action scales and gripper targets are listed in table 4.

A.3 Actor Observations and Task Encoding

Table 3 summarizes the actor inputs in eq. (1), where square brackets concatenate features. Figure 7 illustrates the task encoding and object–target pose buffer.

Table 3: **Benchmark observation groups and input dimensions.** Dimensions use the full 40-task configuration with the default multi-hot encoding. State and visual actors share 66 features of task context and proprioception; their task-state interfaces contribute 234 and 384 features, respectively. Symbols follow eq. (1).

Group	Symbol	Contents	Actor	Dimension
Task context	$a_{t-1},\ z_k$	Previous action and task encoding	both	$7 + 36 = 43$
Proprioception	o_t^{proprio}	End-effector pose (7), gripper joints (2), arm joint positions (7) and velocities (7)	both	23
State task input	$o_t^{\text{object-target}}$	Object–target pose buffer: 26 entity slots, 9 features per slot	state	$26 \times 9 = 234$
Visual task input	h_t^{visual}	Pooled frozen-ViT features from third-person and wrist images	visual	$2 \times 192 = 384$
State actor total: $43 + 23 + 234$				**300**
Visual actor total: $43 + 23 + 384$				**450**

Task encoding. The default multi-hot vector $z_k \in \{0, 1\}^{36}$ combines six shared subtask indicators with 30 dedicated task-ID bits (fig. 7a). Ten tasks activate their applicable shared indicators; each remaining task activates one dedicated bit, ordered lexicographically by its suite-qualified task key. The task ID k remains fixed throughout the rollout; a canonical one-hot task-ID encoding is also available.

Object–target pose buffer. At each step, $o_t^{\text{object-target}}$ concatenates live object and target states in the robot base frame using the fixed entity slots in fig. 7b. Each slot allocates three coordinates each to position, roll–pitch–yaw orientation, and articulation, with unused joint coordinates and task-inactive slots set to zero. Focused encoding retains only entities in the active task subset.

Actor input dimensions. Both actors share the previous action, task encoding, and 23-dimensional proprioception (end-effector position and quaternion in the robot base frame, gripper finger positions, and arm-joint positions and velocities). Adding the pose buffer gives 300 state-input features; using the pooled visual feature instead gives 450 (table 3). Each 224×224 camera image yields 196 patch tokens of width 192; concatenating the two views gives a 196×384 array, which attention pooling reduces to $h_t^{\text{visual}} \in \mathbb{R}^{384}$ (appendix B.1). The DGPO reference stack's privileged critic inputs are specified separately in section 4.1.

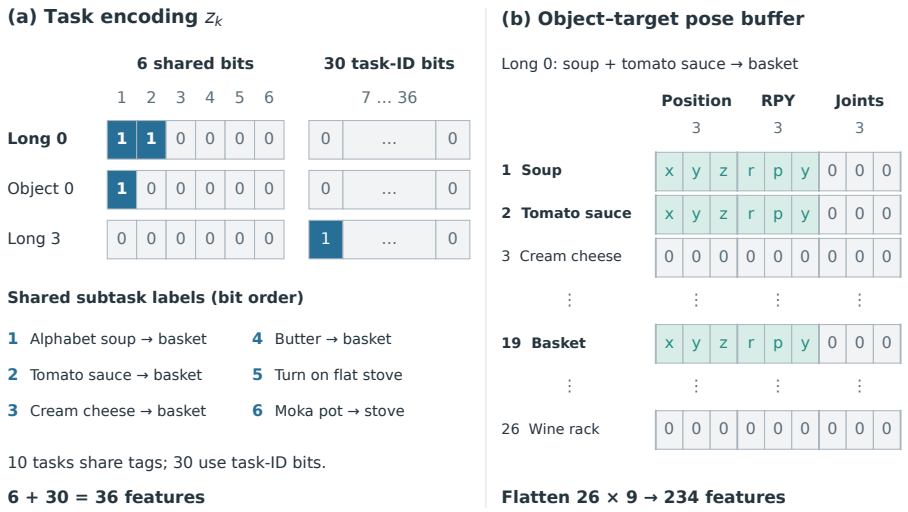

Figure 7: **Task encoding and object–target pose layout.** (a) Long 0 places alphabet soup and tomato sauce in the basket, activating shared bits 1 and 2. Object 0 activates only bit 1; Long 3 uses a dedicated task-ID bit. (b) The same Long 0 task fills the soup, tomato sauce, and basket slots. Symbols denote live pose coordinates; gray cells are zero, including the joint coordinates of these rigid entities. Row numbers indicate canonical entity slots; ellipses omit intermediate entries.

A.4 Joint-Limit Penalties

In eq. (2), the joint-limit penalties use position bounds $[q_j^{\min}, q_j^{\max}]$ and velocity limit $\dot{q}_j^{\max}$:

$$\begin{aligned} d_{j,t}^{\text{pos}} &= [q_j^{\min} - q_{j,t}]_+ + [q_{j,t} - q_j^{\max}]_+, \\ d_{j,t}^{\text{vel}} &= \text{clip}\big(|\dot{q}_{j,t}| - 0.95\dot{q}_j^{\max}, 0, 1\big)\,. \end{aligned} \tag{10}$$

Here $[x]_+ = \max(x, 0)$, and the safety sums span all robot joints. All reward coefficients are listed in table 4.

B DGPO Framework and Reference Learners

This section details the asymmetric critic inputs and visual features, IW-ABC action targets and weighting schedules, and reference learner implementations.

B.1 Asymmetric Critic Inputs and Visual Features

Actor and critic inputs follow eq. (1) and section 4.1; network dimensions are listed in table 4. For environment e paired with demonstration $\xi(e)$ of length $T_{\xi(e)}$, let r_e be its zero-based reset cursor and t the number of control steps since reset. The current demonstration cursor and its normalized critic input are

$$m_e(t) = \min\big(r_e + t,\, T_{\xi(e)} - 1\big)\,, \qquad \phi_{e,t} = \frac{m_e(t)}{\max\big(T_{\xi(e)} - 1, 1\big)} \in [0, 1], \tag{11}$$

where r_e and $T_{\xi(e)}$ are measured in stored control steps; $\phi_{e,t}$ is written as ϕ_t when the environment index is omitted.

Visual feature aggregation. A frozen Theia-style Tiny ViT (Shang et al., 2024) extracts P patch tokens of dimension D_v from each third-person and wrist image. The two views are concatenated along the feature dimension into $Z_t \in \mathbb{R}^{P \times 2D_v}$ and reduced by the actor's trainable single-query attention pool:

$$h_t^{\text{visual}} = \sum_{p=1}^{P} \text{softmax}_p \left(\frac{q_\pi^\top Z_{t,p}}{\sqrt{2D_v}} \right) Z_{t,p}, \qquad q_\pi \in \mathbb{R}^{2D_v}. \tag{12}$$

Here $Z_{t,p}$ is the concatenated token at patch p, and softmax normalizes attention over patches. The pooled vector $h_t^{\text{visual}} \in \mathbb{R}^{2D_v}$ supplies the actor input in eq. (1). The critic applies the same pooling operation with its own query q_V, producing $h_t^{V,\text{visual}}$ from the same frozen tokens. The resulting visual critic input is given in section 4.1. The Tiny ViT image encoder remains frozen; PPO trains the separate actor and critic attention pools and MLPs.

B.2 Time-Indexed Action Targets

ABC uses direct time indexing to construct lightweight action targets for massively parallel rollouts, avoiding online state matching, phase estimation, or optimal-transport alignment. Ordinary training episodes restore the stored initial simulator state $s^D_{\xi(e),0}$ with $r_e = 0$; RFCL's reset curriculum is specified in appendix B.5. Using the cursor $m_e(t)$ from eq. (11), the ABC action target is

$$a^\star_{e,t} = a^{\text{demo}}_{\xi(e),m_e(t)}. \tag{13}$$

Here $a^{\text{demo}}_{\xi,m}$ is the stored action at index m of trajectory ξ; $a^\star_{e,t}$ is the environment-specific form of $a^\star_t$ in eq. (6). The cursor advances by one index per control step until reaching $T_{\xi(e)} - 1$, triggering demonstration-horizon truncation and an environment reset. This lightweight action target may become misaligned when the policy drifts from the paired demonstration. Demonstration-initial-state resets establish initial alignment, while the shared tracking reward encourages the rollout to remain close to the reference, supporting the target's use as a soft imitation guide. The coefficient in eq. (5) further reduces imitation pressure as task success increases and anneals it over training, allowing the policy to increasingly depart from the demonstrated actions.

B.3 IW-ABC Minibatch Weighting and Schedules

For minibatch $\mathcal{B}$, sample i belongs to task $k(i)$ and uses $\tilde{w}_k = w_k / (|\mathcal{B}|^{-1} \sum_{i \in \mathcal{B}} w_{k(i)})$ in eq. (8). This normalization gives unit mean weight across samples, preserving the average loss scale while changing relative task contributions. The weighted losses are reduced in one shared backward pass, without computing separate per-task gradients. Warm-up settings are listed in table 4; fig. 8 visualizes the schedules defined in eqs. (5) and (7).

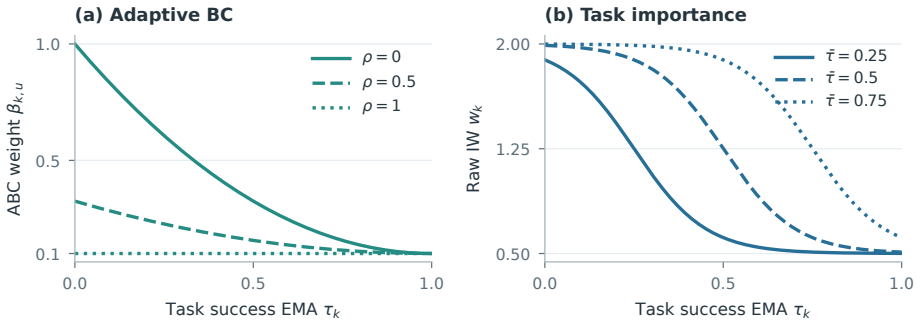

Figure 8: **Online success controls the two adaptive framework weights.** ABC pressure (left; $\beta_{k,u} \in [0.1, 1]$, κ=2) decreases with absolute task success and time, with $\rho_u = 1$ after 200,000 vectorized control steps. Raw task importance (right; $w_k \in [0.5, 2]$, s_{IW}=10) compares each task with the mean over $\mathcal{K}_{\text{init}}$ and is minibatch-normalized. Both use the shared default in table 4.

B.4 FAMO-ABC Implementation

FAMO-ABC adapts the critic-balancing rule from FAMO's original actor–critic implementation (Liu et al., 2023a) to the shared PPO backbone. It retains the ABC targets and coefficient

schedule from appendices B.2 and B.3, while the PPO policy surrogate and entropy use uniform sample weights. FAMO supplies the critic objective in place of the IW-weighted value loss.

For each PPO minibatch, let L_k be the mean clipped value-regression error for task k and $D_k = L_k + \epsilon$, with $\epsilon = 10^{-8}$. Task loss sums and sample counts are aggregated across GPUs; the active task set $\mathcal{K}_\mathcal{B}$ contains tasks represented in the global minibatch. With trainable logits $\mathbf{q}$ and $\omega_k^{\mathrm{FAMO}} = \mathrm{softmax}(\mathbf{q}_{\mathcal{K}_\mathcal{B}})_k$, the value objective is

$$c = \mathrm{sg}\left(\sum_{k\in\mathcal{K}_\mathcal{B}} \frac{\omega_k^{\mathrm{FAMO}}}{D_k}\right), \qquad \mathcal{L}_V^{\mathrm{FAMO}} = c \sum_{k\in\mathcal{K}_\mathcal{B}} \mathrm{sg}(\omega_k^{\mathrm{FAMO}}) \log D_k, \tag{14}$$

where sg stops gradients. This multiplicative scaling follows the FAMO RL implementation and gives effective critic coefficients $c\omega_k^{\mathrm{FAMO}}/D_k$.

After the network update, an additional critic forward pass recomputes D_k on the same observations with fixed returns and old value predictions. The logits receive the progress-based gradient

$$\delta_k = \log D_k^{\mathrm{before}} - \log D_k^{\mathrm{after}}, \qquad g_k = \omega_k^{\mathrm{FAMO}} \left(\delta_k - \sum_{j\in\mathcal{K}_\mathcal{B}} \omega_j^{\mathrm{FAMO}} \delta_j\right). \tag{15}$$

We initialize $\mathbf{q} = \mathbf{0}$ and apply Adam to $\mathbf{q}$ using $\mathbf{g}$, learning rate 0.025, and weight decay 0.01 after every PPO minibatch. Tasks absent from the global minibatch are excluded from the softmax and progress calculation. The shared policy and critic retain one network backward pass per minibatch; FAMO adds the critic forward pass and the task-logit update.

B.5 Reference Learner Specifications

The shared setup and demonstration-interface taxonomy are defined in sections 4.1 and 4.2. We give only the additional objectives, initialization rules, and reset curriculum below; hyperparameters are collected in table 4.

Offline demonstration dataset preparation. Before training, demonstrations are preprocessed into a shared dataset $\mathcal{D} = \{(s_t^D, o_t^D, a_t^D)\}$, where s_t^D is the stored simulator state and o_t^D is the actor observation including task encoding. Trajectory and timestep indices are retained: BC pretraining and DAPG use the observation–action pairs, while RFCL uses the stored states $s_{\xi,j}^D$ for curriculum resets.

Offline BC and transfer to PPO. Offline BC trains an architecture-matched actor on the full dataset $\mathcal{D}$ using mean-squared action error:

$$\mathcal{L}_{\mathrm{BC}}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(s_t^D, o_t^D, a_t^D)\in\mathcal{D}} \left\|\mu_\theta(o_t^D) - a_t^D\right\|_2^2. \tag{16}$$

Here (o_t^D, a_t^D) is a recorded pair from the dataset. BC→PPO transfers only the pretrained actor weights, initializes a fresh critic and optimizer, and removes the BC loss when online training begins. Its online budget matches the other learners; offline pretraining is additional.

DAPG regularization. DAPG (Rajeswaran et al., 2018) augments policy gradient with a demonstration log-likelihood term evaluated on pairs from the same preprocessed dataset $\mathcal{D}$:

$$\mathcal{L}_i^{\mathrm{DAPG}} = -\lambda_0 \lambda_1^i \mathbb{E}_{(s^D, o^D, a^D)\sim\mathcal{D}}\left[\log \pi_\theta(a^D \mid o^D)\right], \tag{17}$$

Here i is the PPO-update index, λ_0 is the initial coefficient, and λ_1 is its per-update decay factor; task encoding is included in o^D.

PPO-based RFCL curriculum. Our adaptation of RFCL (Tao et al., 2024) maintains an independent reverse-reset curriculum per task, with on-policy PPO updates and no offline demonstration replay or auxiliary imitation loss. Unlike the original forward-expansion phase, its second phase samples uniformly from all demonstration initial states of the task. For task k, let $\xi_1, \ldots, \xi_{N_k}$ index its demonstration trajectories. During the reverse phase, trajectory ξ_i of length T_{ξ_i} initializes its

frontier cursor to $c_i = \lfloor p_{\text{init}} T_{\xi_i} \rfloor$. An environment e paired with $\xi(e) = \xi_i$ samples its reset cursor and stored state as

$$\begin{aligned} r_e &= \min(c_i + \Delta, T_{\xi_i} - 1), \qquad s_0 = s^D_{\xi_i, r_e}, \\ \Delta &= \min(X, 4), \quad X \sim \text{Geom}(p_\Delta), \\ c_i &\leftarrow \max(c_i - \delta, 0) \quad \text{after frontier mastery.} \end{aligned} \tag{18}$$

Here $s^D_{\xi_i,j}$ is the stored state at step j of trajectory ξ_i, r_e is the reset cursor in eq. (11), and X is a nonnegative geometric offset. Frontier mastery requires m consecutive successful frontier rollouts, after which the cursor moves backward by δ. Once all active cursors for task k reach zero, resets switch to its full set of N_k demonstration starts:

$$i \sim \text{Uniform}\{1, \ldots, N_k\}, \qquad \xi(e) = \xi_i, \quad r_e = 0, \quad s_0 = s^D_{\xi_i, 0}, \tag{19}$$

Initialization and mastery settings are listed in table 4.

C Reproducibility Configuration

Table 4 groups the benchmark settings, reference-policy architectures, optimization parameters, and learner-specific schedules by component. The tracking weights $w_{\text{ee,p}}, w_{\text{ee,r}}, \ldots$ instantiate the component weights w_m in eq. (3).

Table 4: **Reproducibility configuration by component.** Settings for the benchmark and reported reference learners, including schedule units.

Symbol / setting	Description	Value
Task set and demonstrations		
K	Jointly trained tasks	40 (ten per suite)
$p(T)$	Task allocation	Uniform
N_k	Demonstration trajectories per task	50
Benchmark simulation		
Δt_{sim}	Physics time step	$1/60$ s
d	Action decimation	3
Δt_{ctrl}	Control time step	0.05 s (20 Hz)
H	Maximum episode horizon in control steps	520
$T_{\max} = H \Delta t_{\text{ctrl}}$	Maximum episode duration	26 s
η_v	Joint velocity limit multiplier	1.5
Action and controller		
$\dim(a_t)$	Policy action dimension	7
s_p	Relative translation action scale	$1/20$
s_R	Relative rotation action scale	$1/2$
$\theta^{\text{grip}}_{\text{open}}$	Open gripper target	0.04 m
$\theta^{\text{grip}}_{\text{close}}$	Closed gripper target	0.0 m
Benchmark task reward and safety		
w_{succ}	Sparse success bonus	1.0
$w_{\Delta a}$	Action smoothness penalty	$5{\times}10^{-4}$
w_a	Action norm penalty	$5{\times}10^{-4}$
$w_{\dot{q}}$	Joint velocity penalty	10^{-3}
$w^{\text{lim}}_{\text{pos}}$	Joint position soft-limit penalty	1.0
$w^{\text{lim}}_{\text{vel}}$	Joint velocity soft-limit penalty	0.5

Table 4 (continued)

Symbol / setting	Description	Value
DGPO demonstration-tracking reward		
$\sigma_{r,m}$	Component-specific tracking kernel bandwidth	0.1
$w_{\mathrm{ee,p}}$	Dense EE position tracking weight	0.5
$w_{\mathrm{ee,r}}$	Dense EE rotation tracking weight	1.0
w_{grip}	Dense gripper tracking weight	0.4
$w_{\mathrm{joint,p}}$	Dense arm-joint position tracking weight	0.5
$w_{\mathrm{joint,v}}$	Dense arm-joint velocity tracking weight	0.1
$w_{\mathrm{obj,p}}$	Dense object position tracking weight	0.10
w_{art}	Dense articulation tracking weight	0.05
$w_{\mathrm{obj,r}}$	Dense object rotation tracking weight	0.05
Shared state/visual IW-ABC default schedule		
u	Schedule counter accumulated across rollouts	Vectorized control steps
s_{IW}	Task-importance sigmoid slope	10
$w_{\max}$	Maximum importance weight	2
$w_{\min}$	Minimum importance weight	0.5
α_{ema}	Per-task success EMA rate	0.05
N_{reset}	Completed episodes per task required for inclusion in $\mathcal{K}_{\mathrm{init}}$	20
U_{IW}	Task-importance warmup (vectorized control steps)	1,000
$\beta_{\max}$	Maximum ABC weight	1
$\beta_{\min}$	Minimum ABC weight	0.1
κ	Adaptive-BC beta-kernel concentration	2
U_{ann}	ABC horizon (vectorized control steps)	200,000
$U_{\mathrm{ann}}/T_{\mathrm{rollout}}$	Equivalent ABC horizon in PPO iterations	12,500
c_{BC}	Base ABC coefficient	1
Shared reference-policy architecture		
$\dim(o_t^{\mathrm{state}})$	State actor input dimension	300
h_π, h_V	Actor and critic MLP hidden widths	$[512, 256, 128]$
act	MLP activation	ELU
History	Temporal context	None (feed-forward)
σ_0	Initial policy standard deviation (log-parameterized Gaussian)	0.8
norm	Observation normalization	running mean/std
Shared PPO reference stack		
Training horizon	Online PPO iterations	30,000
opt	Optimizer	Adam, β_1=0.9, β_2=0.999
γ	Discount factor	0.99
ε	PPO policy and value clip range	0.15
c_V	Value-loss coefficient	1.0
c_H	Entropy bonus coefficient	0.005
λ_{GAE}	GAE trace decay	0.95
T_{rollout}	Vectorized control steps per PPO iteration	16
N_{ep}	PPO epochs per rollout	5
N_{mb}	Minibatches per epoch	4
η_θ	Policy and value learning rate	2×10^{-4}
$\mathrm{KL}^\star$	Adaptive learning rate KL target	0.005
$g_{\max}$	Global gradient norm clip	1.0

Table 4 (continued)

Symbol / setting	Description	Value
Visual reference configuration		
Backbone	Tiny ViT training and execution mode	Frozen; inference mode
Cameras	RGB observation views	Third-person and wrist
p_{vit}	ViT patch size	16
D_v	ViT hidden dimension	192
L_v	ViT transformer depth	12
$H_{\text{img}}, W_{\text{img}}$	Camera image resolution	$224, 224$
P	Patch tokens per camera	196
Offline BC and BC→PPO pretraining		
$\lvert\mathcal{D}\rvert$	Demonstration state–observation–action records	336,575
Training horizon	Offline optimizer steps	30,000
Batch size	Transitions per optimizer step	4,096
Learning rate	Offline actor learning rate	10^{-4}
Normalization	Actor observations	Running mean/std
DAPG offline regularization		
λ_0	DAPG initial coefficient	0.1
λ_1	DAPG iteration decay	0.995
RFCL reset curriculum		
m	RFCL frontier success window	3 rollouts
δ	RFCL cursor decrement	8 steps
p_Δ	RFCL geometric offset parameter	0.5
p_{init}	RFCL Phase 1 cursor init fraction	0.85

D Supplementary IW-ABC Analyses

We examine where to apply task importance weighting in the actor and critic, and how IW and ABC weights evolve across tasks during training.

D.1 Task Weighting in the Actor and Critic

The common task allocation, policy class, demonstration set, online PPO horizon, and training seeds are specified in section 5.2; numerical optimization and rollout settings are consolidated in table 4. The ablation below tests where task IW is applied within the selected IW-ABC reference configuration while retaining the same learner family.

Where task IW is applied. Within IW-ABC, table 5 compares policy-side weighting with the default that also weights value regression. Using the minibatch notation from appendix B.3 and coefficients from eq. (8), let ℓ_i^π and ℓ_i^V denote the per-sample negative clipped PPO surrogate loss and clipped value-regression loss at minibatch index i, respectively; $\mathcal{H}_i$ is the policy entropy. The ABC demonstration term in eq. (6) is unchanged and omitted below; the two PPO variants optimize

$$\mathcal{L}_{\text{actor+critic}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \tilde{w}_{k(i)} \left(\ell_i^\pi + c_V \ell_i^V - c_H \mathcal{H}_i \right), \tag{20}$$

$$\mathcal{L}_{\text{actor only}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left[\tilde{w}_{k(i)} (\ell_i^\pi - c_H \mathcal{H}_i) + c_V \ell_i^V \right]. \tag{21}$$

Thus entropy follows the actor weight in both variants; all other settings are identical. Weighting both actor and critic improves Mean SR by 11.9 percentage points and Long SR by 20.0 points over actor-only weighting, while increasing task coverage from 31 to 35 out of 40 (table 5). These gains

support applying task IW to value regression as well as policy optimization, which we adopt as the default.

Table 5: **Task-IW placement.** The reported ablation uses IW-ABC with a PPO backbone. "Actor" includes policy and entropy; "critic" denotes value regression.

IW placement	Actor	Critic	Mean SR (%)	Long SR (%)	Task coverage
Actor only	✓	×	78.2±3.1	50.0±0.0	31/40
Actor + critic (default)	✓	✓	**90.1±3.8**	**70.0±10.0**	**35/40**

D.2 Task-Level IW and ABC Dynamics

Figure 9 complements fig. 5 with four tasks selected to illustrate different learning speeds in the same Vis IW-ABC run. G7 learns early, reducing both its BC coefficient and its relative PPO weight, while L2 improves during annealing and progressively relinquishes demonstration guidance. G5 retains high IW after BC reaches its residual floor; success rises around 17k iterations, followed by lower IW as the task approaches the multi-task mean. L5 remains unsolved within the observed training budget and retains high IW, maintaining its PPO priority. Together, these examples show how ABC relaxes imitation with task progress and training time, while IW continues emphasizing tasks that lag behind.

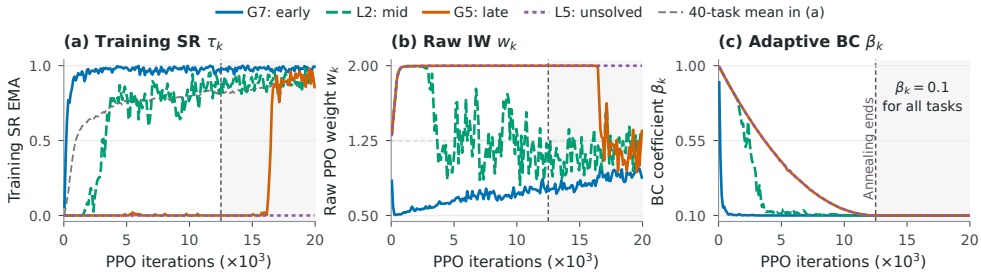

Figure 9: **Illustrative task trajectories for Vis IW-ABC.** Training success EMA τ_k (a), raw IW w_k (b), and BC coefficient $\beta_{k,u}$ (c) for G7, L2, G5, and L5 from the same single training run as fig. 5. Gray in (a) denotes the 40-task mean. Vertical dashed lines mark the end of ABC annealing at 200,000 vectorized control steps, after which every task has $\beta_{k,u} = 0.1$.

E Real-World Deployment Details

Deployment pipeline. FoundationPose (Wen et al., 2024) estimates object poses from RGB-D observations using object meshes corresponding to the simulation USD assets. Camera calibration and object-frame alignment express these estimates in the robot base frame. The planar target pad uses color segmentation and depth back-projection instead of FoundationPose. The frozen actor produces joint-position targets and gripper commands in a closed feedback loop.

Piper observation layout. The Piper policy uses a 42-dimensional object–target buffer with six slots, each storing a 3D position and a four-component quaternion; task-inactive slots are zero. Together with task identity (4), previous action (7), joint positions (8) and joint velocities (8), this yields a 69-dimensional actor input matching the Piper training interface.