BUILDING EXPLORATORY ROBOT POLICIES VIA ADVERSARIAL MID TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot policies today are predominantly trained on human-collected data, which naturally concentrates on competent, successful behavior. As a result, such data provides limited coverage of off-distribution states and diverse low-level behaviors needed to solve complex tasks over long horizons or improve through their own practice. In this paper, we study how to explicitly build this coverage into general-purpose robot policies. Specifically, we study a distinct collection protocol, *adversarial data collection*: while a human attempts a task, an adversary reversibly perturbs the environment or the task, forcing the operator to encounter unfamiliar states and adapt via low-level behaviors. We collect such rollouts, relabel segments with natural-language annotations identifying the local outcome pursued in each segment, and use them to midtrain a history-conditioned pretrained robot policy. The resulting policy retries after failures, makes sustained progress over longer horizons, and internalizes diverse ways of solving a given task. Across several long-horizon tasks on three real-robot platforms and in simulation, compared to training with standard demonstration data, our approach yields better scaling of zero-shot performance, greater robustness and generalization to novel configurations, gains from additional test-time compute (i.e., longer rollouts), and faster improvement during online finetuning. Project page and robot videos are available at https://explr-beta.pages.dev/.

1 INTRODUCTION

Generalist robot policies are trained on demonstrations spanning increasingly diverse tasks and environments, yet these demonstrations often consist of only a narrow range of competent behaviors to accomplish each task. Across various modalities, human-collected data is often restricted to narrow, near-optimal behavior; teleoperation data and UMI-style collection (Chi et al., 2024) recordsuccessful trajectories, and even human videos primarily expose models to behaviors people already execute well themselves. This contrasts with language models, where general-purpose capability has emerged from broad pretraining and midtraining distributions containing diverse behaviors and failures. In contrast, robot policies receive much of their action supervision from expert executions. We argue that this creates an issue of limited *behavioral coverage*: once a policy departs from states that are well represented in the training data distribution, it may simply not possess the low-level behaviors needed to continue making progress on a task. Broad low-level behavior coverage is also a key enabler for online improvement, since a policy must be able to self-generate diverse experience to improve via techniques like reinforcement learning.

For instance, consider packing unfamiliar objects of different shapes and sizes into a box. Even given a correct high-level plan specifying which object to manipulate next, successful execution can require very different low-level realizations depending on the evolving geometry of the task: approaching from another direction, changing grasp or orientation, repositioning an object before insertion, rearranging existing contents, or changing the sequence of motions used to realize the same planned step. Thus, the challenge is not primarily one of specifying *what* should happen next, but of providing sufficient coverage over *how* that outcome can be physically realized. These low-level behaviors are difficult to enumerate or specify precisely in language a priori, yet they determine whether a policy can continue making progress in unfamiliar configurations. Figure 1 illustrates distinct low-level behaviors for a task focused on disassembling structures made of bricks.

Obtaining such coverage with human data collectors is not straightforward. Prior intervention-based methods (Kelly et al., 2019; Liu et al., 2023) collect useful human corrections to steer a base policy

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.

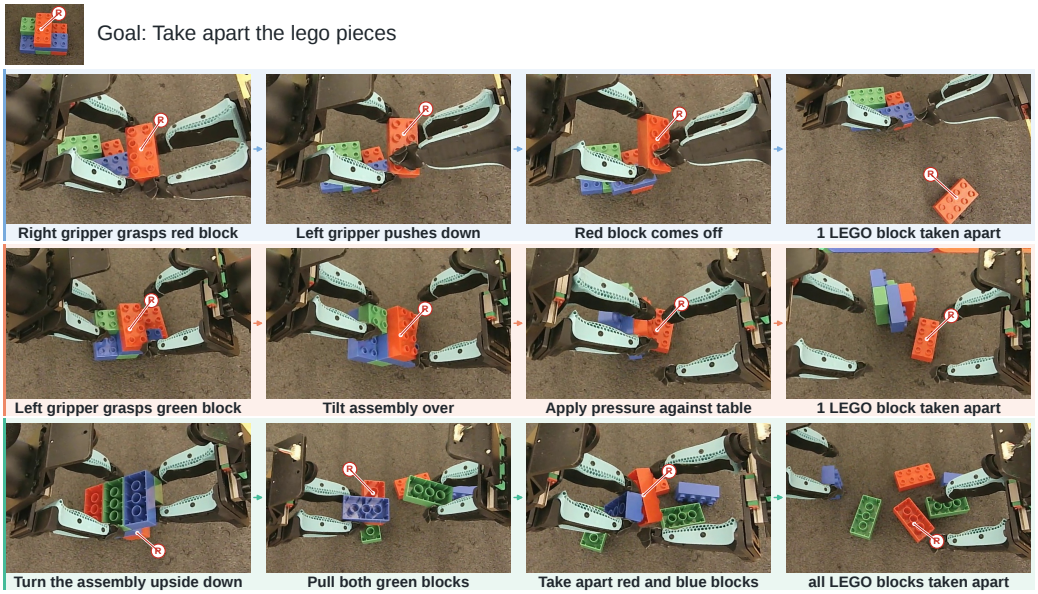

Figure 1: **Connected behavioral coverage induced by β-DATA policy in LEGO disassembly.** Starting from the same initial LEGO blocks assembly and task goal, slight deviations in grasps, orientations, and contact interactions lead to diverse sequences of outcome states. The top and middle rollouts each detach a different red brick while the bottom sequence separates all the bricks, via uniquely different strategies. Our exploratory policy is capable of generating meaningful alternative interactions and sustain progress toward task completion.

on a given task, but only on states visited by a particular policy and for correcting mistakes that this particular policy makes on-policy. Noise injection into the policy (Fujimoto et al., 2018) or teleoperation interface (Laskey et al., 2017) broadens the data distribution, but it does so uniformly at all states, which becomes impractical for precise tasks and high dimensional action spaces. Finally, language-based metadata conditioning (Intelligence et al., 2026) can steer behavior along predefined axes but it does not generally capture the many low-level motions available at a given state. Thus, there remains no easy mechanism for deliberately improving low-level behavioral coverage.

How can we enhance low-level behavioral coverage? Perhaps the most direct approach is to collect data that exhibits such diversity by construction. However, simply increasing state or action coverage independently is insufficient. Independent demonstrations from hard initial states do not necessarily capture behaviors such as retries and recovery, while naïvely adding action stochasticity may produce incoherent behavior without improving low-level coverage at the states where it is most critical. What we need instead is coverage over useful low-level behaviors at *critical states*, together with transitions that *connect* critical states with other states from which subsequent task progress becomes easier. For example, when an object cannot be inserted into a crowded container, useful behaviors may include reorienting it or temporarily displacing another object to create a jointly feasible configuration (Figure 10), rather than simply adding arbitrary motion around the current state. We refer to this property as *connected behavioral coverage*.

To obtain this behavioral coverage, we need a scalable way to elicit diverse low-level behaviors from human operators, including retries and recovery. We build on *adversarial data collection* (Huang et al., 2025), which separates data collection into two roles: an *operator*, who continually attempts to complete the task, and an *adversary*, who modifies the ongoing task so the operator must find a different way forward. Rather than perturbing the operator's actions directly, the adversary changes the physical environment or task specification, controlling *where* alternative behavior is elicited while leaving *how* to respond to the operator. For example, during packing, the adversary may rotate an object, obstruct a placement, change container occupancy, or alter the task goal, forcing the operator to reorient objects, rearrange the scene, change its motion sequence, or redirect its behavior. Because the operator must continue toward successful completion after each perturbation, the resulting trajectories contain coherent yet diverse low-level behaviors precisely where they are needed. We call this protocol β-DATA (BETA-DATA): **B**uilding **E**xploratory policies via mid**TrA**ining.

We use the β-DATA to *midtrain* a pretrained robot policy, broadening its behavioral support before downstream task-specific post-training. Long adversarial rollouts create two challenges: a single

episode-level instruction, or even subtask instructions can inadequately describe local behavior, and the appropriate continuation can depend on previous attempts and perturbations in a non-Markovian way. We therefore relabel coherent trajectory segments with the local goals pursued and condition the policy on a short interaction history to better fit subsequent actions.

What can* β-DATA *enable? We evaluate the utility of midtraining (i.e., continued training after pretraining a generalist policy) with β-DATA through extensive data-scaling studies and downstream post-training experiments. On a long-horizon bimanual simulated packing environment with 20 evaluation tasks β-DATA improves the success of a $\pi_{0.5}$ (Intelligence et al., 2025) base policy 54.7% with standard human data to 69.7% with β-DATA. The largest gains emerge on hard, unseen scenes, where success improves from 39.2% to 57.6%. Behavioral analyses associate these gains with productive retries and sustained progress through the final stages of a task. We further find more favorable data-scaling curves with β-DATA, and observe similar improvements across three real-world long-horizon bimanual tasks involving item packing, disassembling complex shapes, and folding clothes. Finally, we also show that policies trained with β-DATA are better able to collect useful experience for subsequent post-training and improve online via practice.

2 PRELIMINARIES, NOTATION, AND PROBLEM STATEMENT

We consider language-conditioned robotic manipulation, with instruction space $\mathcal{L}$ and action space $\mathcal{A}$. A task is specified by a natural-language instruction $l \in \mathcal{L}$. At timestep t, the robot receives an observation s_t, consisting of image observations and proprioceptive states, and executes an action $a_t \in \mathcal{A}$. The policy predicts an action chunk $a_{t:t+H-1} = (a_t, \ldots, a_{t+H-1})$ of H actions rather than a single action. Because many behaviors of interest in this work depend on previous attempts and failures, we consider history-conditioned policies. We denote the action history at timestep t by $h_t = (a_{0:t-1})$, and write a policy with learned parameters θ as $a_{t:t+H-1} \sim \pi_\theta(\cdot \mid s_t, h_t, l)$. Our implementation uses a recent window $\bar{h}_t$ of this history, defined in Section 5. Executing a policy for T timesteps produces a trajectory $\tau = (s_0, a_0, \ldots, s_T)$. Each task instruction l induces a binary success function $r_l(\tau) \in \{0, 1\}$ indicating completion status at the end of the rollout. We also measure dense metrics quantifying the amount of task progress made by the rollout trajectory.

Imitation learning trains π_θ on a dataset $\mathcal{D}_{\text{demo}}$ of demonstrations. A demonstration can be written as $\tau_i = (l_i, s_0^i, a_0^i, \ldots, s_{T_i}^i)$, where the human controls the robot toward successful completion of instruction l_i. Such trajectories may be collected through teleoperation or other forms of human-generated behavior data, but in each case predominantly consist of goal-directed behavior produced by competent human operators. For a dataset $\mathcal{D}$ of such demonstrations, we write the log-loss form of behavioral cloning as $\mathcal{L}_{\text{BC}}(\theta; \mathcal{D}) = -\mathbb{E}_{(\tau,l)\sim\mathcal{D}} \left[\sum_t \log \pi_\theta(a_{t:t+H-1} \mid s_t, h_t, l)\right]$. Our focus is not on changing this imitation objective, but on the *distribution of behavior* represented in $\mathcal{D}$. In particular, we ask how data should be collected so that a generalist robot policy acquires broad coverage over both states and low-level behaviors, rather than primarily reproducing expert execution.

Online finetuning. Policies that exhibit genuinely broader behavioral coverage should provide a better initialization for autonomous online improvement on previously unseen tasks. Given a downstream task $l^\star$ and an initial policy π_{θ_0}, we perform filtered behavioral cloning (filtered BC) over successive rounds. At round k, we roll out the current policy π_{θ_k} on $l^\star$ until it produces n_s successful trajectories $\tau_k^{(1)}, \cdots, \tau_k^{(n_s)}$. We add these trajectories to an online success buffer, $\mathcal{D}_{\text{on}}^{(k)} = \mathcal{D}_{\text{on}}^{(k-1)} \cup \{\tau_k^{(1)}, \cdots, \tau_k^{(n_s)}\}$, and run behavioral cloning on them. Following standard practice, we retain a fraction of the original offline data during each update to regularize the policy and mitigate forgetting. We describe our procedure for online finetuning in Section 7.

3 ENVIRONMENTS, TASKS, AND ROBOT SETUPS

To assess the importance of low-level coverage and validate our approach, we study challenging long-horizon bimanual manipulation tasks that require dexterous skills across a sequence of dependent subtasks. A capable high-level planner should not resolve the challenges of low-level execution by itself. When the next subtask is specified in language, the robot must still determine how to grasp, reorient, and move objects under the constraints of the current scene. For example, placing an object in a crowded container may require changing the grasp or moving an obstruction before attempting the placement. These decisions also affect subsequent subtasks: an early placement can restrict later motions, and execution errors can accumulate over long horizon. Completing the task, therefore, requires the policy to adapt its low-level behavior reactively and sustain progress via exploration.

Simulation and real-world tasks. We instantiate these settings in one simulated and three real-world manipulation domains. In Mujoco (Todorov et al., 2012) simulator, we construct 64 distinct *grocery packing* tasks, in which bimanual xArm 7 robot arms arrange multiple grocery items of varying shapes and sizes between differently sized containers, posing non-trivial spatial constraints to the low level dexterous skills as shown in Figure 4. We describe their train-hold-out split, tasks goals, numbers of subtasks, and horizon length in Appendix A Table 3.

Our real-world experiments are conducted on three bimanual robot platforms: **(1) *grocery packing*** on xArm 7 robot, where grocery items are to be packed into two containers with spatial constraints; **(2) *LEGO bricks disassembly*** on bimanual YAM robot, where complex LEGO block structures are taken apart and sorted into colored trays, requiring precise contact forces and dexterous manipulation strategies; and **(3) *garment folding*** on Raiden (Iwase et al., 2026) bimanual YAM arms, where deformable garments are folded flat and transferred to a target location. Detailed task definitions and evaluation criteria are described in Appendix B.



Figure 2: **Robot setups & tasks overview.**

Full tasks and subtasks. A *full task* l specifies the desired final state, including object destinations and constraints (e.g., "pack all objects efficiently"). It may require a sequence of *subtasks* $\tilde{l}$, each specifying a local semantic outcome, such as "pack the sugar box into the medium tray." Neither level fully specifies the *low-level behavior*: the sequence of motor actions used to achieve that outcome. Depending on the scene, the same subtask may require different grasps, approach directions, reorientations, bimanual transfers, temporary restaging, or repeated attempts. Our goal is to broaden coverage over these low-level realizations, as opposed to only increasing the diversity of tasks.

Hierarchical and direct execution. In the *hierarchical* setting, a high-level planner provides the current subtask instruction $\tilde{l}_t$, and the policy produces continuous actions to realize it. In the *direct* setting, the policy receives the full task instruction l throughout the rollout, without an external subtask selector. The hierarchical setting tests whether broader behavioral coverage helps even when *what* to do next is given, leaving the policy to determine *how* to achieve it.

4 Related Work

Data scaling in robot learning. The success of foundation models (Kaplan et al., 2020; Hoffmann et al., 2022; Brown et al., 2020) has motivated efforts to scale data, model size, and compute in robot learning. Scaling real-robot demonstrations has enabled both dexterous task-specific policies (Zhao et al., 2024) and increasingly generalist policies spanning tasks, environments, and embodiments (O'Neill et al., 2024; Brohan et al., 2023). As robot datasets continue to grow, an important question is therefore not only how much data to collect, but what kind of experience is most productive to scale. Our work focuses on this question at the level of low-level behavior: standard expert demonstrations provide broad task and state coverage, but may still provide narrow coverage over the different ways a policy can adapt, recover, and continue making progress. In contrast, β-DATA-style datasets provide a far better alternative type of data for scaling.

Data collection for scaling imitation learning. Because real-world robot data is expensive, several works study how to use a fixed collection budget more effectively. Prior approaches broaden coverage across objects, environments, initial poses, or combinations of these factors (Lin et al., 2024; Gao et al., 2024; Zha et al., 2025; Xiao et al., 2026). However, the resulting trajectories still largely consist of successful expert execution and do not necessarily improve low-level behavioral coverage. Play data (Lynch et al., 2019) provides broader experience through task-agnostic interaction, but does not deliberately interrupt task progress to elicit alternative low-level behaviors. A complementary line of work collects corrections or recoveries during policy execution (Wu et al., 2025; Hu et al., 2026). While this exposes diverse behaviors from challenging states, the resulting data distribution remains tied to the failures of the base policy, making it harder to collect a reusable training corpus independently of the downstream learner. Adversarial data collection (ADC) (Huang et al., 2025) instead perturbs the environment or task during human teleoperation while the operator continues toward completion, naturally eliciting retries, strategy changes, and recovery without learner rollouts. We use a variant of ADC in β-DATA and use these trajectories as midtraining data



Figure 3: **Perturbation and recovery during data collection.** The teleoperator packs an item, the adversary returns it to the table, and the teleoperator repacks it. We use this protocol to collect β-DATA.

for pretrained policies. Rather than proposing a new protocol, our focus is on studying how the resulting behavioral coverage affects generalization, scaling, and online improvement.

Exploratory behaviors in robot policies. Broad behavioral support can be useful in two related ways. First, it can allow a policy to continue searching for a successful realization of a task during a single rollout. Analogously, language and vision-language models can revise intermediate solutions through reasoning and interaction (Wei et al., 2022; Yao et al., 2022; Zawalski et al., 2024; Guo et al., 2025), while recurrent policies can adapt their behavior based on previous observations and actions (Duan et al., 2016). In robotics, this motivates policies that can retry, recover, or change low-level strategy rather than committing to a single nominal execution. Second, a policy's behavioral support determines the experience it can generate for subsequent learning. Reinforcement-learning methods encourage exploration through action noise (Fujimoto et al., 2018), entropy maximization (Haarnoja et al., 2018), or intrinsic rewards (Pathak et al., 2017; Burda et al., 2018), while offline approaches learn behavioral priors, skills, or exploratory policies that improve subsequent data collection (Singh et al., 2020; Pertsch et al., 2021; Wagenmaker et al., 2025b). Recent work also broadens pretrained robot-policy behavior by modeling uncertainty in demonstrator actions(Wagenmaker et al., 2025a) or perturbing policy inputs(Hong et al., 2026). In contrast, β-DATA aims to build exploratory behavior directly into a pretrained VLA through its midtraining distribution. The resulting policy can use alternative low-level behaviors to sustain progress on unfamiliar configurations and, importantly, generate more useful successful trajectories for subsequent online finetuning.

5 β-DATA: EXPLORATORY POLICIES VIA ADVERSARIAL MIDTRAINING

In this section, we introduce β-DATA, our approach for midtraining pretrained robot policies to broaden their connected low-level behavioral coverage. As discussed above, the same semantic subtask can admit many physical realizations depending on the state and interaction history, including different grasps, reorientations, rearrangements, and retries. Our goal is to expose the policy to these alternative physical realizations, together with the transitions that connect them to continued task progress, rather than simply increasing coverage over tasks or language instructions.

Why are state and action diversity alone insufficient? Diversifying initial states can expose operators to challenging configurations, but does not necessarily capture the behaviors through which execution moves from difficulty to adaptation and resumed progress. Conversely, directly diversifying actions, either by instructing the data collector or injecting noise into the teleoperation interface, provides little control over *where* useful diversity appears. Cloning such behavior may also produce incoherent behavior. What we seek instead is *connected behavioral coverage*: challenging states should arise within ongoing execution, together with several low-level actions at these states that allow a competent operator to reach other states from where they can continue making progress.

Protocol: *operator-adversary data collection*. To obtain this type of connected coverage, we build on the adversarial data collection protocol of Huang et al. (2025). In this protocol, two humans jointly collect each rollout: an *operator* continually attempts to complete the task, while an *adversary* intervenes during the episode to change the environment or task instructions and force the operator to adapt. This separation allows the adversary to control *where* alternative behavior is elicited while leaving *how* to respond up to the operator. The operator's objective remains task completion, and they are never instructed to behave diversely or produce a particular recovery strategy, avoiding arbitrary variation in low-level actions at states where expert behavior is already sufficient. Crucially, after each intervention, the operator continues from the perturbed state without restarting. As a result, the collected rollouts are long horizon and contain connected sequences from disruption, adaptation, to resumed task progress, providing alternative low-level continuations precisely at critical states. We refer to the collected data as β-DATA (BETA-DATA).

Midtraining on the collected data. The protocol above produces diverse distinctive behaviors as the operator adapts to changing states and goals. Conditioning every action in such a trajectory only

on the original task instruction can therefore be misspecified: the operator may locally remove an obstruction, transfer an object, retry a failed maneuver, or temporarily undo progress before continuing toward the final goal. We introduce two design choices. First, **D1: segment-level outcome relabeling**, where we segment each rollout according to the local outcome pursued and annotate each segment with this outcome in language, $\tilde{l}_t$. The annotation specifies *what* the operator is trying to accomplish locally, while the demonstrated actions determine *how* that outcome is realized. We apply the same relabeling procedure to trajectories from expert demonstration data. Second, **D2: action-history conditioning**, where we condition the policy on a short 2 seconds window of recent history actions. At critical states, the same observation and subtask instruction can require different continuations depending on whether the robot has just approached an object, failed a grasp, withdrawn, or begun a retry. Recent actions therefore provide behavioral context for selecting among these continuations, rather than merely compensating for partial observability. We then train the pretrained robot policy to predict future action chunks conditioned on the current observation, segment-level language annotation, and recent action history. At evaluation, a high-level planner supplies a corresponding subtask instruction. We ablate these design choices next.

Implementation details. We instantiate β-DATA by midtraining the open-source VLA $\pi_{0.5}$ pretrained robot policy. To implement action-history conditioning (**D2**), we encode the previous 120 steps of action into four tokens that are provided as prefix to the action expert. During midtraining, we fully finetune the image encoder and action expert and train rank-16 LoRA adapters on the language backbone, using the standard flow-matching objective for 50K updates. We provide the remaining implementation details, policy architecture, and evaluation details in Appendix E.

6 β-DATA SCALES ZERO-SHOT PERFORMANCE DATA EFFICIENTLY

We study the zero-shot benefits of β-DATA through two questions. **RQ1:** Does β-DATA broaden the learned behavioral repertoire and improve zero-shot generalization? We evaluate task progress and success in simulation and the real world, examine low-level policy behavior, and test harder configurations. **RQ2:** How do these benefits scale with training data (i.e., training compute) and rollout horizon (i.e., test-time compute)? We also ablate the two main design choices from Section 5: **Ablation 1:** action-history conditioning (**D2**), which provides behavioral context; and **Ablation 2:** segment-level outcome annotations (**D1**), which condition control on the local subtask.

Result 1: Zero-shot performance of policies trained on β-DATA vs. expert data. We first study whether training on β-DATA improves zero-shot success rates on our tasks. Table 1 summarizes results for policies trained on expert demonstrations and β-DATA on simulated and real-world tasks. The twenty simulation packing tasks consist of in-distribution (ID) tasks represented during training and out-of-distribution (OOD) tasks with held-out goals and configurations and are grouped based on the expected task horizon. We provide a 6 minute wall-clock budget for any evaluation rollout in both simulation and the real world. We report detailed per-task results in Appendix D. β-DATA substantially improves zero-shot task completion compared to training on expert data, both on the success rate and partial progress.

	Success rate		Progress (%)	
Task	**Expert**	**β-DATA**	**Expert**	**β-DATA**
Simulation				
Short, OOD	136/150	**148/150**	95.4	**99.4**
Medium, ID	121/150	**137/150**	88.4	**94.3**
Medium, OOD	142/200	**183/200**	84.9	**94.7**
Long, ID	50/250	**85/250**	41.4	**58.9**
Long, OOD	98/250	**144/250**	64.0	**77.6**
Real robot (OOD)				
YAM, interlock	5/20	**19/20**	42.5	**96.7**
YAM, double	2/20	**6/20**	35.8	**59.4**
xArm, packing	4/20	**7/20**	66.1	**91.7**
YAM, folding	2/20	**5/20**	38.3	**45.8**

Table 1: ***Zero-shot evaluation performance on simulation and real robot tasks.*** Midtraining robot policies on β-DATA yields higher success than expert data across all reported task groups, including challenging held-out configurations on real robot platforms. The largest gains in task progress occur on long-horizon tasks, supporting our hypothesis that broader behavioral coverage helps policies sustain progress over extended horizons.

Qualitative behavior. In Figure 4 and Figure 12, we illustrate how broader low-level behavioral coverage can matter even when the desired subtask is already specified. Starting from the same state on OOD Task 9, both policies successfully pack the first three objects and are instructed to place

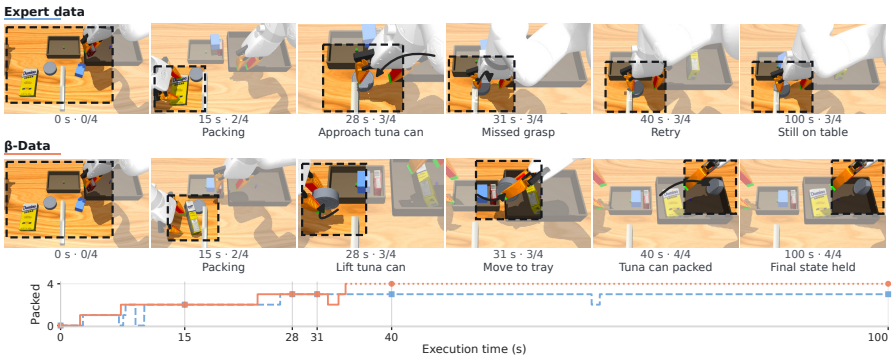

Figure 4: **Comparisons of simulated packing task rollouts from the same initial state.** The policy trained on expert data (top) packs three objects but repeatedly fails to grasp and lift the final tuna can, which remains on the table after 100 s. In contrast, β-DATA trained policy (bottom) successfully lifts and packs the tuna can, completing the four-object task in 34.43 s. The bottom curve shows task progress over the course of rollouts.

the final can into the tray. Expert data trained policy lifts the can but fails to complete the grasp and continues attempting without making further progress. In contrast, β-DATA successfully lifts the can, transfers it to the tray, and completes the task. This example shows that the remaining challenge is not deciding *what* to do, but executing an appropriate low-level realization of the specified subtask.

Result 2: β-DATA scales more favorably with both training data (i.e., training compute) and rollout horizon (i.e., test-time compute). Next, we study if the benefits of β-DATA and its relative gains over standard expert data persist as we scale up the amount data. We train policies on varying amounts of β-DATA or expert data and evaluate zero-shot success on the twelve OOD simulation tasks. Figure 5 shows that while standard expert data performs better at the smallest budget (16.5% vs. 6.0%), β-DATA overtakes it thereafter. At 1.3, 2.6, and 5.2 hours, β-DATA improves over expert data by 16.3, 13.8, and 10.7 percentage points, respectively, reaching 69.3% vs. 58.7% at the largest budget. Notably, β-DATA trained on 2.6 hours reaches 60.3% success, comparable to the 58.7% achieved by expert data trained on *twice* as much data. Here, β-DATA achieves comparable performance with about ***2*** $\times$ ***less data***, while continuing to improve as the training-data budget increases.

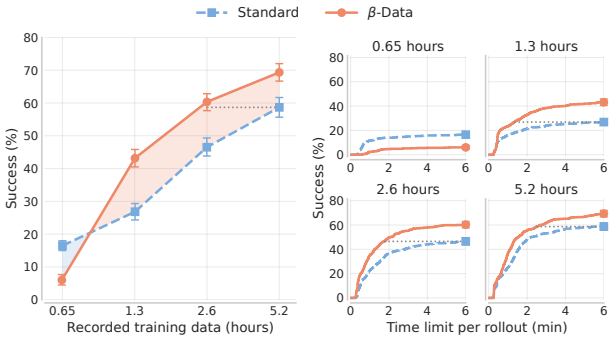

Figure 5: **Training data and test-time scaling of β-DATA in simulation.** ***Left:*** zero-shot success as a function of the amount of midtraining data across four matched data budgets. β-DATA overtakes expert-data midtraining beyond the smallest budget and achieves comparable performance with roughly half as much data at the larger budgets. ***Right:*** success as a function of rollout horizon for policies trained at each data budget. From 1.3 hours onward, β-DATA outperforms expert data across rollout horizons and continues to gain from additional interaction time. Each point is evaluated over 600 trials (12 tasks $\times$ 50); error bars show pointwise 95% bootstrap intervals over evaluation scenes and trials.

The very same policies also benefit more effectively from additional rollout time (i.e., test-time compute). At the 5.2-hour training budget, increasing the rollout limit from two to six minutes raises β-DATA success rate from 55.5% to 69.3%, compared with 47.8% to 58.7% for standard expert data. From 1.3 hours of training data onward, β-DATA outperforms expert data at every rollout limit from one to six minutes. Moreover, β-DATA reaches the six-minute success of the corresponding expert-data policy within only 1.3, 1.7, and 2.5 minutes at the 1.3, 2.6, and 5.2 hour budgets, respectively. The advantage is not only due to faster early completion: at 5.2 hours, 12.0% of evaluation rollouts for β-DATA still unfinished at four minutes succeed by six minutes, compared with 4.6% for expert data. Together, these results indicate that broader behavioral coverage allows the policy to continue making productive progress as additional test-time interaction is available.

Ablation 1: Action history (D2) helps sustain behavior over long horizons. We next ablate the various decisions involved in midtraining on β-DATA. First, we study whether recent action history helps the policy select how to continue a low-level behavior, beyond what is specified by the

current observation and subtask instruction. We compare β-DATA and standard expert data policies trained either with a 120-step action history or without history, using matched training and evaluation settings otherwise. Figure 6(a) shows that the benefit is largest for β-DATA on long-horizon tasks. In simulation, on OOD long-horizon tasks, history improves β-DATA success from 41.6% to 57.2%, compared with a 4.0 point gain for standard imitation, and increases packing ratio from 62.8% to 77.4%. All five OOD long tasks improve, uniformly. On ID tasks, β-DATA success increases from 28.8% to 34.0%, with packing ratio increasing from 47.0% to 59.0%. In contrast, history has little effect on easier OOD Short and Medium tasks, where β-DATA is already near saturation without it.

These results support the role of action history as *behavioral context*. Long adversarial trajectories in β-DATA expose the policy to behaviors such as approaching a constraint, backing off, retrying, and continuing from a new configuration; recent actions help distinguish which stage of such a maneuver the policy is currently executing. This is particularly important on long-horizon tasks, where repeated manipulation and rearrangement continually change the constraints faced by the policy. Thus, broad behavioral coverage alone is not sufficient: the policy must also condition on its recent behavior to select an appropriate continuation from that coverage.

Ablation 2: Subtask annotations (D1) improve generalization to held-out tasks. We next study the role of the segment-level task annotations used during midtraining. We compare direct execution from a single full-task instruction with hierarchical execution, where a high-level planner provides the policy with the current subtask. The intuition is that a subtask instruction makes the immediate outcome explicit, while action history specifies how the current attempt has unfolded. Figure 6(b) shows that providing subtask instructions substantially improves performance on OOD tasks. For β-DATA, success on OOD Long tasks increases from 20.0% to 57.6%, with packing ratio increasing from 52.6% to 77.6%; on OOD Medium tasks, success increases from 62.5% to 91.5%. Policies trained on standard expert data also benefit, improving from 4.8% to 39.2% on OOD Long tasks and from 2.5% to 71.0% on OOD Medium tasks. The gains are much smaller and more mixed on ID tasks, suggesting that explicitly specifying the local outcome is most useful when executing tasks and configurations not represented during training.

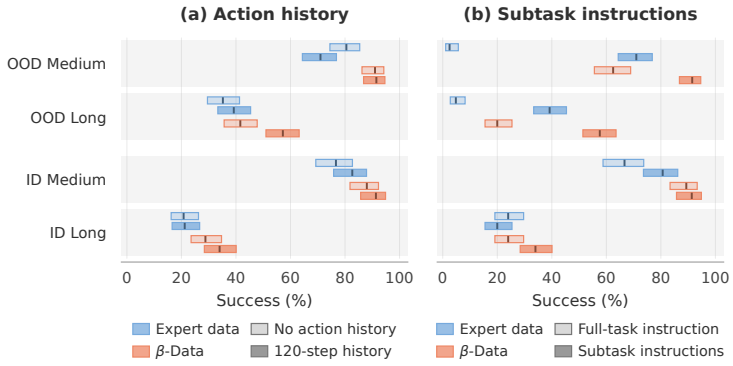

Figure 6: **Ablations of action-history conditioning and language instruction granularity.** Each ablation reports full-task success averaged equally across tasks in each category. **(a)** β-DATA benefits substantially from action-history conditioning on long OOD tasks, whereas the corresponding gain is much smaller for policies trained on expert demonstrations. **(b)** Policies trained on β-DATA are also less sensitive to instruction granularity: β-DATA outperforms expert-data training under both full-task and subtask-level instructions. This suggests that its gains are not explained simply by improved high-level task specification or planning, but instead arise from complementary improvements in low-level behavioral coverage.

Importantly, ***subtask decomposition does not remove the need for broad behavioral coverage***. Even when both policies receive subtask instructions, β-DATA exceeds the expert-data policy by 20.5 and 18.4 percentage points on OOD Medium and Long tasks, respectively. Conversely, both policies remain at or below 16% success on the hardest OOD rearrangement tasks, showing that specifying *what* outcome to pursue does not by itself determine *how* to realize it. Together with Result 2, these results suggest complementary roles for the two design choices: subtask annotations specify the outcome we desire, while action history provides context for selecting an appropriate continuation from the policy's learned behavioral repertoire.

7 TRAINING VIA β-DATA ENABLES EFFICIENT ONLINE FINETUNING

We next ask whether the broader behavioral coverage induced by β-DATA also improves a policy's ability to learn from its own experience. There are two ways this can happen. First, a policy with

broader low-level coverage may be able to generate successful experience more easily. Second, even given the same number of successful rollouts, the content of those rollouts may differ: a policy with broader behavioral coverage can succeed from a wider range of states and through more diverse low-level behaviors, in turn providing more useful supervision for subsequent updates. We observe both effects when applying filtered behavioral cloning (BC) to policies trained with β-DATA, where the current policy collects successful rollouts and is repeatedly updated on a mixture of this online experience and existing offline midtraining data.

Result 1: β-DATA makes useful online experience substantially easier to collect on challenging tasks. The largest difference appears on long-horizon tasks, where a successful trajectory may require recovering from local failures while still completing the full task. As shown in Figure 7 and Figure 8, collecting successful experience on a long OOD task requires only 160 rollouts on average from the β-DATA initialization, compared with 3810 from the standard expert-data initialization. Despite this substantially smaller interaction budget, β-DATA improves from 57.6% to 73.2% success, while the expert-data policy improves from 39.2% to 52.0%. As such, on this task the slope of performance improvement is higher for β-DATA compared to standard expert data. Thus, broader behavioral coverage substantially lowers the interaction cost of obtaining the successful experience needed for online improvement.

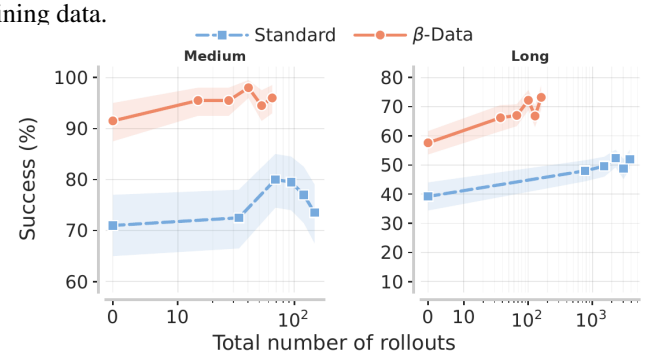

Figure 7: **β-DATA policies enable more efficient online finetuning.** On held-out simulated ***grocery packing*** tasks, filtered BC on top of β-DATA policies achieves higher success using fewer online interactions than initialized with expert-data initialization.

Result 2: β-DATA dominates even when training on a matched number of online successful rollouts. We next test whether differences remain even when the amount of accepted successful experience collected during online finetuning is matched to what the standard expert data approach collects. On the real-world dual-xArm packing task, both policies receive exactly twenty accepted successful trajectories and 200 gradient updates. Nevertheless, the policy initialized with β-DATA reaches 70% full-task success rate after finetuning, compared with 15% for the standard expert-data initialization, while packing ratio (i.e., progress) reaches 90.6% versus 53.3% (Table 2). Thus, providing the same number of successful trajectories does not erase the gap between β-DATA and standard expert data initialization: the usefulness of online data also depends on the states and behaviors represented within those trajectories.

Training data	Success (%)	Progress (%)
Expert data	$20 \rightarrow 15$ [5.2, 36.0]	$66 \rightarrow 53.3$ [41.7, 65.6]
β-DATA	**$35 \rightarrow 70$** [48.1, 85.5]	**$91.7 \rightarrow 90.6$** [82.2, 97.2]

Table 2: **Real-world dual xArm *grocery packing* eval performance after online finetuning.** Both policies receive the same number of online successful trajectories and 200 gradient updates. Despite matched post-training budgets, the policy initialized with β-DATA reaches substantially higher full-task success and packing ratio (i.e., progress).

Qualitative behavior: why can equally many successes provide different learning signal? The accepted trajectories collected by the two policies differ substantially in their behavioral coverage. For the expert-data policy, 95% of accepted successes come from a single arrangement in which objects are grouped by destination box, whereas β-DATA collects successes under re-randomized object configurations. The former can therefore collect a rollout that depicts a relatively scene-specific execution strategy, while the latter exposes the learner to a broader range of object arrangements and low-level continuations. Consistent with this difference, after fine-tuning the β-DATA policy shows stronger instruction grounding and more productive recovery: it acts on the instructed object more reliably and eventually completes 91% of failed-grasp sequences, compared with 62% for the expert-data policy. These observations suggest that online improvement depends not only on the quantity of successful experience, but also on its behavioral coverage.

Limitations. Our online finetuning uses filtered behavioral cloning and does not explicitly perform credit assignment; incorporating a learned critic or Q-function could provide more targeted feedback. In addition, we condition on only a short fixed action history, while richer long-context policies may better capture longer-term behavioral dependencies and exploit the coverage in β-DATA.

AI USE STATEMENT

In this work, we used AI coding agents to assist with research code implementation. Additionally, we used generative AI tools to polish the grammar, wording, and readability of the manuscript. We reviewed all AI-assisted work and take responsibility for the final content of this work, including all texts, claims, codes, and artifacts produced with the aid of generative AI.

REFERENCES

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. Pmlr, 2018.

Jensen Gao, Annie Xie, Ted Xiao, Chelsea Finn, and Dorsa Sadigh. Efficient data collection for robotic manipulation via compositional generalization. *arXiv preprint arXiv:2403.05110*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Matthew M. Hong, Jesse Zhang, Anusha Nagabandi, and Abhishek Gupta. Tmrl: Diffusion timestep-modulated pretraining enables exploration for efficient policy finetuning. In *Robotics: Science and Systems (RSS)*, 2026.

Zheyuan Hu, Robyn Wu, Naveen Enock, Jasmine Li, Riya Kadakia, Zackory Erickson, and Aviral Kumar. Rac: Robot learning for long-horizon tasks by scaling recovery and correction. *IEEE Transactions on Robotics*, 2026.

Siyuan Huang, Yue Liao, Siyuan Feng, Shu Jiang, Si Liu, Hongsheng Li, Maoqing Yao, and Guanghui Ren. Adversarial data collection: Human-collaborative perturbations for efficient and robust robotic imitation learning. *arXiv preprint arXiv:2503.11646*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Shun Iwase, Patrick Miller, Jonathan Yao, Krishna Murthy Jatavallabhula, and Sergey Zakharov. RAIDEN: A toolkit for policy learning with YAM bimanual robot arms, 2026.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083, 2019. doi: 10.1109/ICRA.2019.8793698.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pp. 143–156. PMLR, 2017.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024.

Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Robotics: Science and Systems (RSS)*, 2023.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1903.01973.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Andrew Wagenmaker, Perry Dong, Raymond Tsao, Chelsea Finn, and Sergey Levine. Posterior behavioral cloning: Pretraining bc policies for efficient rl finetuning, 2025a. URL https://arxiv.org/abs/2512.16911.

Andrew Wagenmaker, Zhiyuan Zhou, and Sergey Levine. Behavioral exploration: Learning to explore via in-context adaptation. 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. 61885–61912. PMLR, 13–19 Jul 2025b. URL https://proceedings.mlr.press/v267/wagenmaker25a.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Philipp Wu, Yide Shentu, Qiayuan Liao, Ding Jin, Menglong Guo, Koushil Sreenath, Xingyu Lin, and Pieter Abbeel. Robocopilot: Human-in-the-loop interactive imitation learning for robot manipulation. *arXiv preprint arXiv:2503.07771*, 2025.

Yuyang Xiao, Yifei Zhou, Haoran Wang, Wenxuan Ou, and Yuxiao Liu. Towards generalizable robotic data flywheel: High-dimensional factorization and composition. *arXiv preprint arXiv:2603.25583*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

Lihan Zha, Apurva Badithela, Michael Zhang, Justin Lidard, Jeremy Bao, Emily Zhou, David Snyder, Allen Z Ren, Dhruv Shah, and Anirudha Majumdar. Guiding data collection via factored scaling curves. *arXiv preprint arXiv:2505.07728*, 2025.

Tony Z Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Kamyar Ghasemipour, Chelsea Finn, and Ayzaan Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024.

Appendices

A SIMULATION TASK DESCRIPTIONS

We group tasks by the object level changes required from their initial configurations. We define short horizon tasks to require one or two placements. Medium-horizon tasks require three to five direct placements. Long horizon tasks require either three or more changes involving removal or transfer of existing tray contents, or the placement of eight to ten objects. These categories describe task structure rather than measured execution time or policy success; retries and recovery can extend a rollout in any category. The OOD split contains three short-, four medium-, and five long-horizon tasks. The ID split contains three medium- and five long-horizon tasks, with no short-horizon tasks.

The subtask counts in Table 3 count one outcome for each required placement, transfer, or obstruction removal. An object already in its required location adds no subtask, and a transfer counts once from source to destination.

Split	Task	Full goal (summary)	Subtasks
Short			
OOD	Task 1	Move the gelatin box from medium to small, then pack the foam brick into small.	2
OOD	Task 2	Pack the gelatin box and foam brick into the medium tray alongside the sugar box already there.	2
OOD	Task 7	Remove the soup can obstructing the small tray while retaining the gelatin box and foam brick there.	1
Medium			
OOD	Task 9	Pack the gelatin box and foam brick into medium; pack the sugar box and tuna can into large.	4
OOD	Task 10	Pack the soup can into small, the sugar box into medium, and the cracker box and gelatin box into large.	4
OOD	Task 11	Pack the foam brick and sugar box into medium; pack the soup can, tuna can, and gelatin box into large.	5
OOD	Task 12	Pack the gelatin box into small, the foam brick and sugar box into medium, and the soup can and cracker box into large.	5
ID	Task 1	Pack the cracker box, soup can, gelatin box, and foam brick into the large tray.	4
ID	Task 3	Pack the cracker box and soup can into the medium tray; the sugar box and foam brick into the large tray.	4
ID	Task 4	Pack the cracker box, soup can, and gelatin box into the large tray; leave the foam brick and tuna can unpacked.	3
Long			
OOD	Task 3	Remove the cracker box from medium, then pack the sugar box and foam brick there; keep the gelatin box in small.	3
OOD	Task 4	Move the gelatin box and foam brick from medium to small, then pack the sugar box into medium; retain the soup can there.	3
OOD	Task 5	Five objects start in large: keep the cracker box there, move the sugar box and soup can to medium, and the gelatin box and foam brick to small.	4
OOD	Task 6	Move the soup can from small to medium and the gelatin box and foam brick from medium to small. Pack the sugar box into medium.	4
OOD	Task 8	Redistribute five misplaced objects: the cracker box to large, the sugar box and soup can to medium, and the gelatin box and foam brick to small.	5
ID	Task 2	Clear three obstructions from the medium tray, then pack the sugar box and gelatin box there.	5
ID	Task 5	Route six specified objects to the large tray; place the mug and can opener in the medium tray.	8
ID	Task 6	Sort nine objects by size and type: large items to large, cans and the canister to medium, small flat items to small.	9
ID	Task 7	Sort ten objects: food and food containers to large, the mug to medium, and the can opener and foam brick to small.	10
ID	Task 8	Distribute ten objects by explicit assignment across three trays: four to large, three to medium, and three to small.	10

Table 3: **Simulation tasks.** Tasks are grouped by the horizon categories defined above, with OOD and ID tasks identified within each group. Task numbers match the result tables within each split. Subtasks count the required object-level outcomes; the horizon category also accounts for whether the task requires removal or transfer of existing tray contents.

B ROBOT SETUPS

xArm setup. The platform uses two 7-DoF UFactory xArm 7 arms with xArm grippers fitted with scaled-down soft grippers. An overhead ZED Mini stereo camera and two wrist-mounted Intel RealSense D405 cameras provide RGB observations at 60 Hz.

YAM setup. The platform uses two 6-DoF I2RT YAM arms with two-finger grippers. Three global-shutter USB RGB cameras, one overhead and one on each wrist, capture 1920×1080 images at 60 Hz with fixed exposure and white balance. Robot states streamed at 200 Hz are interpolated to the shared timestamp of each camera frame set. The state and absolute action targets each contain 12 arm joint positions and two gripper positions. Two Dynamixel-based GELLO leader arms are read at 200 Hz, with their latest positions applied and recorded at 60 Hz. On both platforms, operators mark subtask boundaries with a foot pedal and label episode outcomes as success or failure; YAM uses a second pedal to end and label each episode. We label the trajectories after collection to form language-labeled subtask training episodes and resize images to 224×224 for training.

Raiden YAM setup. The platform consists of two 6-DoF I2RT YAM arms equipped with parallel grippers. RGB observations are streamed at 30 Hz using one overhead Intel RealSense D435 camera and two wrist-mounted Intel RealSense D405 cameras. A second pair of 6-DoF YAM arms serves as the leader devices for teleoperation, continuously sending their latest poses as control commands to the follower arms. During data collection, operators mark subtask boundaries using buttons integrated into the leader handles. After collection, each trajectory is segmented and annotated with language labels to form subtask-level training episodes. All RGB observations are resized to 224×224 before training.

C TASK DEFINITIONS AND PROGRESS SCORES

We report full-task success and partial task progress. Success requires completing all task requirements. For progress, we divide each trial's score by its maximum possible score and multiply by 100. Reported progress is the mean of these percentages across evaluation trials.

C.1 SIMULATED GROCERY PACKING

Two simulated xArm 7 arms pack grocery objects in the containers specified by the task instruction. Some tasks also require removing obstructions or transferring previously packed objects. Each target object receives one point for being packed into its assigned container while following the instruction for that object, and zero otherwise. For a task with N target objects, the maximum score is N. The task goals and horizon categories are listed in Appendix A.

C.2 REAL-WORLD GROCERY PACKING (XARM)

Two xArm 7 arms pack nine grocery items into two containers. We use the same per-item scoring rule as in simulated packing: an item receives one point only when it is packed into its assigned container while following its instruction. The maximum score is nine points, so each correctly packed object contributes one ninth of the total progress. Full-task success requires correctly packing all nine objects.

C.3 LEGO BRICKS DISASSEMBLY (YAM)

Two YAM arms disassemble interlocking LEGO structures and sort the individual blocks into their assigned trays. We evaluate two configurations: **YAM interlock** contains one assembly of six blocks, while **YAM double** contains two assemblies with nine blocks in total.

Each block receives half a point for being separated from the assembly and another half for placement in its correct tray. The maximum scores are therefore six points for YAM interlock and nine points for YAM double. Separating all blocks without sorting them gives 50% progress; separating and correctly sorting all blocks gives 100% and completes the task.

C.4 GARMENT FOLDING

The robot folds short or long pants, initially laid flat, and moves the folded garment to a designated clean area on the left. It first folds the legs together, then folds the garment in half. Long pants require one additional fold before placement. Short pants therefore require three stages, while long pants require four.

Each successfully completed stage receives one point. Progress is the number of completed stages divided by three for short pants or four for long pants, multiplied by 100. A trial is fully successful only when the entire sequence, including placement, is completed.

Demonstration datasets and collection budgets We collect teleoperated demonstrations across four task domains, comparing Expert data with approximately duration-matched β-DATA data. The datasets contain, respectively, 6.1 and 5.8 hours for simulated grocery packing, 4.35 and 4.76 hours for real-world grocery packing (xArm), 4.0 hours each for LEGO disassembly and sorting (YAM), and 5.6 and 5.7 hours for garment folding and placement (Raiden YAM). Demonstrations are recorded at 60 Hz, except garment demonstrations, which use 30 Hz. For simulated packing, each policy is initialized from $\pi_{0.5}$ and midtrained for 50,000 updates with batch size 256, approximately ten passes through its dataset. The separate scaling study uses matched budgets of 0.65, 1.3, 2.6, and 5.2 hours.

D PER-TASK ZERO-SHOT SIMULATION RESULTS

Table 4 gives the individual simulation results summarized in Table 1. Standard denotes the policy trained on expert demonstrations. Packing ratio is the mean, across trials, of the peak fraction of target objects correctly packed simultaneously during a trial. For c successes in 50 trials, pass@k is estimated as $1 - \binom{50-c}{k} / \binom{50}{k}$, with $\binom{50-c}{k} = 0$ when $50 - c < k$. It estimates success within k attempts with resets between attempts, rather than retries within a single rollout. The evaluation uses one training run per policy, so these results do not measure variation between training runs.

	Standard				β-DATA			
Task	Success	Packing ratio	Pass@5	Pass@10	Success (Δ)		Packing ratio	Pass@5	Pass@10
Out-of-distribution (OOD)									
Short									
Task 1	**100**	100.0	100	100	**100**	(0)	100.0	100	100
Task 2	**98**	99.3	100	100	**98**	(0)	99.3	100	100
Task 7	74	87.0	100	100	**98**	(+24)	99.0	100	100
Medium									
Task 9	92	96.0	100	100	**100**	(+8)	100.0	100	100
Task 10	64	83.5	100	100	**86**	(+22)	90.5	100	100
Task 11	50	66.4	97	100	**90**	(+40)	92.8	100	100
Task 12	78	93.6	100	100	**90**	(+12)	95.6	100	100
Long									
Task 3	70	86.7	100	100	**90**	(+20)	95.3	100	100
Task 4	64	87.5	100	100	**86**	(+22)	92.5	100	100
Task 5	4	26.8	19	36	**16**	(+12)	55.2	60	86
Task 6	56	81.5	99	100	**86**	(+30)	90.0	100	100
Task 8	2	37.6	10	20	**10**	(+8)	55.2	42	69
OOD mean	62.7	78.8	85	88	**79.2**	(+16.5)	88.8	92	96
In-distribution (ID)									
Medium									
Task 1	76	85.5	100	100	**96**	(+20)	96.0	100	100
Task 3	74	83.0	100	100	**82**	(+8)	87.5	100	100
Task 4	92	96.7	100	100	**96**	(+4)	99.3	100	100
Long									
Task 2	90	92.0	100	100	**96**	(+6)	98.0	100	100
Task 5	10	36.8	42	69	**42**	(+32)	65.5	94	100
Task 6	0	24.2	0	0	**18**	(+18)	48.7	65	89
Task 7	0	30.6	0	0	**10**	(+10)	45.2	42	69
Task 8	0	23.4	0	0	**4**	(+4)	37.2	19	36
ID mean	42.8	59.0	55	59	**55.5**	(+12.8)	72.2	78	87
Overall	54.7	70.9	73	76	**69.7**	(+15.0)	82.1	86	92

Table 4: **Zero-shot performance with subtask instructions.** A high-level planner provides each policy with subtask instructions. Within each split, tasks follow the structural categories in Appendix A. Each task uses 50 trials per policy and a 21,600-step limit. All metrics are percentages; higher is better. Bold marks the higher observed success rate in each row, with both values bold on ties. Packing ratio measures peak packing progress, as defined in the text. Parentheses show the success difference from Standard in percentage points. Pass@k estimates success within k attempts, as defined in the text. We weight tasks equally in the split and overall averages. Each policy has 600 OOD trials, 400 ID trials, and 1,000 trials overall. Each policy has one training run.

E POLICY TRAINING SETTINGS

Setting	Midtraining	Online finetuning
Initialization	Released $\pi_{0.5}$ base checkpoint	Final midtraining checkpoint from the same corpus
Trainable modules	Image encoder; action expert; rank-16 backbone adapters; history network and projections	Rank-32 action-expert adapters; rank-16 backbone adapters; history network and projections
Frozen weights	Base language-backbone weights	Image encoder and base weights of the language backbone and action expert
Trainable parameters	874M of 3.38B	53.6M: 22.1M action-expert adapters, 27.9M backbone adapters, and 3.6M history network and projections
Optimizer	AdamW: $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 10^{-10}; gradient-norm clipping at 1	Same as midtraining
Learning rate	Linear warmup over 2,000 updates to 10^{-4}; cosine decay to 10^{-6} at 50,000 updates	Linear warmup over ten updates to 3×10^{-5}; then constant
Training budget	50,000 gradient updates	Ten updates per round; 50 rounds and 500 updates in total
Batch	256 examples from the training corpus	128 offline, 64 newly admitted, and 64 earlier online examples; first round uses 128 newly admitted examples
Weight averaging	None	Per-round exponential moving average; decay $\min\{0.95, (k+1)/(k+10)\}$ after $k \geq 1$ rounds (0.85 at round 50)
Actions and history	60-action chunks at 60 Hz; 14 action dimensions; 120-action history encoded as four tokens	Same as midtraining
Normalization	Per-coordinate 1st and 99th percentiles of the corpus mapped to -1 and 1	Fixed midtraining statistics
Online replay	Not applicable	Equal trajectory weights within the fresh and earlier groups; uniform sampling of eligible timesteps within each trajectory
Eligible online frames	Not applicable	At least one state coordinate changes by 10^{-3} or more; use all frames if fewer than 10% qualify

Table 5: **Training settings for simulated packing.** Standard and β-DATA share these settings, with separate corpora and normalization statistics. LoRA adapts attention and feed-forward weights. The online column describes one-success rounds; the hours-matched midtraining schedule is given in Appendix F.

F DATA AND ROLLOUT-TIME SCALING DETAILS

Rollout time at every data budget. Table 6 summarizes the curves on the right of Figure 5. At 1.3, 2.6, and 5.2 hours, β-DATA succeeds more often than Standard at every limit from one to six minutes; at 0.65 hours, Standard leads at every limit. Six-minute packing ratio follows the same ordering across data budgets, reaching 82.8% for β-DATA and 76.8% for Standard at 5.2 hours.

		Success (%) by limit			Packing ratio (%)	Time used (min)	Late completion (%)
Data	Method	2 min	4 min	6 min			
0.65 h	Standard	14.0	15.8	16.5	33.7	5.20	0.8
0.65 h	β-DATA	4.7	5.7	6.0	27.8	5.74	0.4
1.3 h	Standard	21.3	25.3	26.8	46.8	4.73	2.0
1.3 h	β-DATA	32.8	40.2	43.2	64.7	4.00	5.0
2.6 h	Standard	36.3	44.0	46.5	65.8	3.88	4.5
2.6 h	β-DATA	49.7	58.0	60.3	74.9	3.12	5.6
5.2 h	Standard	47.8	56.7	58.7	76.8	3.28	4.6
5.2 h	β-DATA	55.5	65.2	69.3	82.8	2.79	12.0

Table 6: Test-time statistics of the hours-matched policies. Packing ratio and mean time used are measured under the six-minute limit, with failed trials counted at the limit for time used. Late completion is the percentage of trials still running at four minutes that succeed by six minutes. Each row is one training run evaluated on 600 trials

G ONLINE FINETUNING PROTOCOL AND TASK-LEVEL RESULTS

Training batches and replay. Each gradient update uses 256 training examples: 128 from offline demonstrations and 128 from admitted online trajectories. Once earlier online trajectories are available, half of the online examples come from the newly admitted trajectories and half from the earlier buffer. During the first update round, all 128 online examples come from the newly admitted trajectories. For ten-success rounds, the fresh examples are allocated approximately equally among the ten new trajectories rather than in proportion to trajectory length. Over the ten updates, each new trajectory contributes the same number of examples. Earlier trajectories are sampled uniformly with replacement, so each of the m trajectories in the earlier success buffer has probability $1/m$ of being selected. Ten percent of the offline episodes are held out from training. These choices keep recent successes represented during learning while retaining earlier online experience and offline demonstrations.

Optimization and frame sampling. The learning rate reaches 3×10^{-5} after ten warmup updates and remains constant thereafter. The policy used for collection and saved checkpoints uses exponentially averaged parameters, updated once per ten-update block. After $k \geq 1$ completed rounds, the averaging decay is $\min\{0.95, (k+1)/(k+10)\}$. Online replay samples timesteps uniformly among those where at least one robot-state coordinate changes by 10^{-3} or more between consecutive frames. If fewer than 10% of an episode's timesteps meet this motion criterion, sampling uses all timesteps instead.

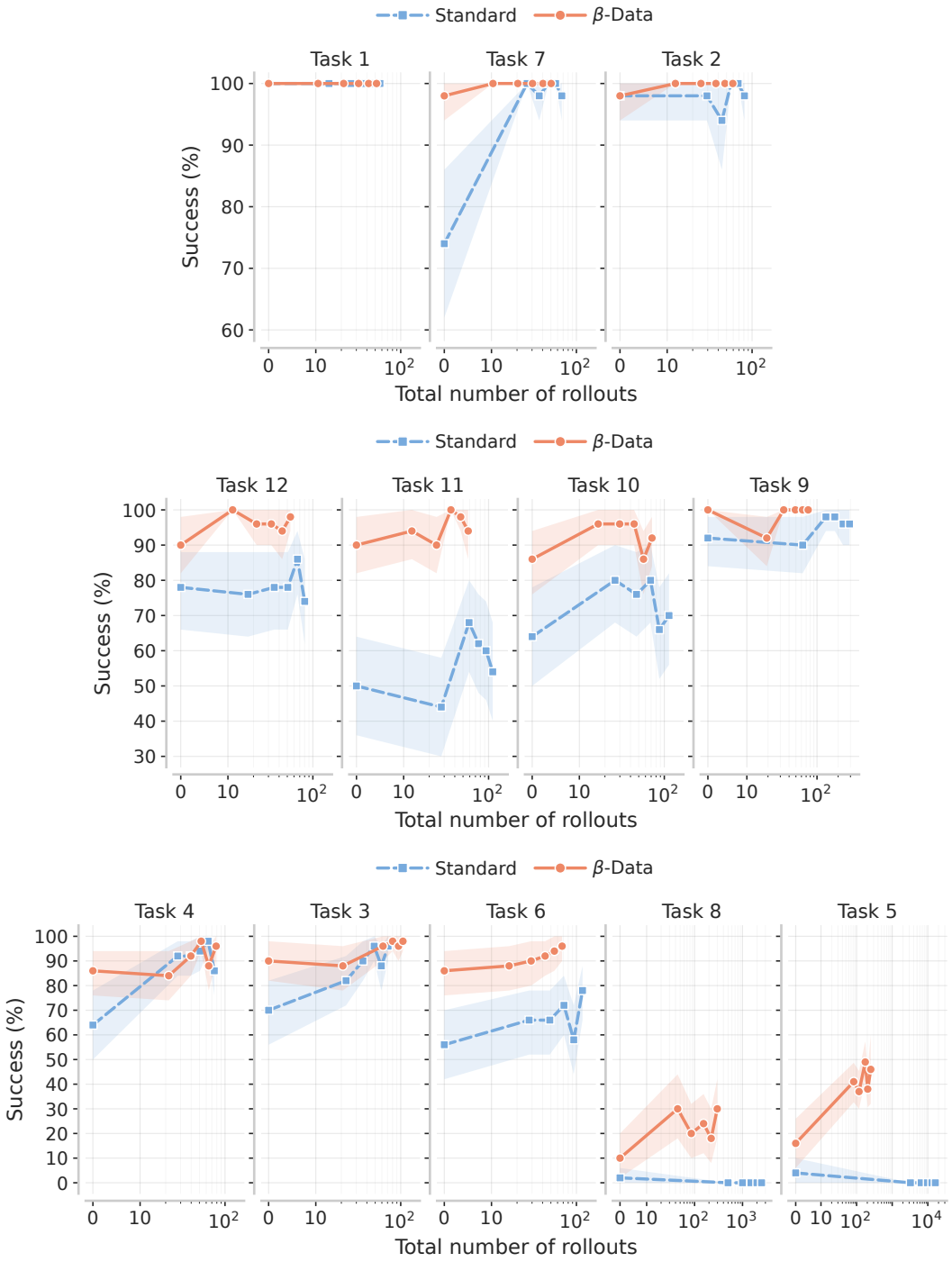

Figure 8: **Online finetuning on individual OOD tasks.** Rows show Short, Medium, and Long tasks. Each panel shows one task, ordered within its row by the larger of the two methods' total number of rollouts at the final checkpoint, from smallest to largest. The horizontal axes are linear from 0 to 10 rollouts and logarithmic above 10. Shading shows pointwise 95% bootstrap intervals for success. Standard is blue and dashed; β-DATA is orange and solid.

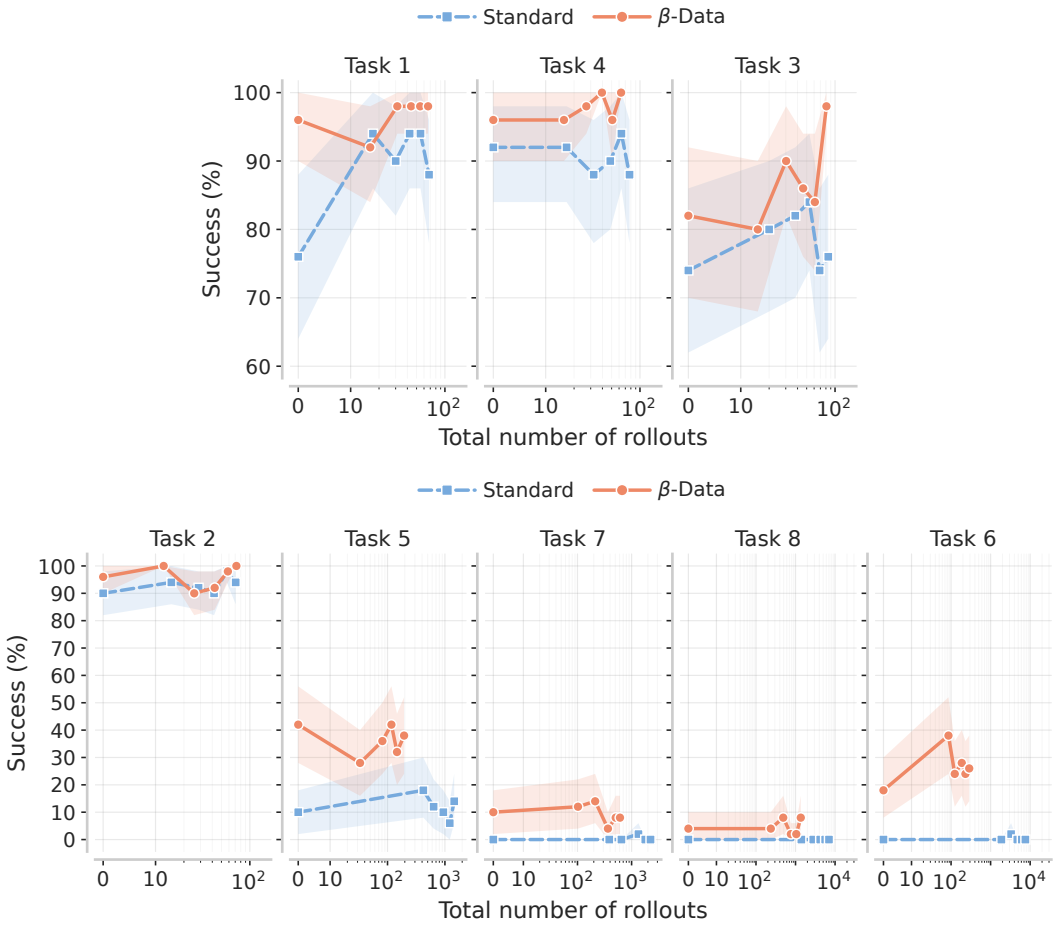

Figure 9: **Online finetuning on individual ID tasks.** Rows show Medium and Long tasks. Each panel shows one task, using the ordering, axis scales, and pointwise 95% bootstrap intervals of Figure 8. Standard is blue and dashed; β-DATA is orange and solid.

H SIMULATED PACKING DATA COLLECTION AND POLICY ROLLOUTS

Figure 10 shows how the operator responds to interventions during simulated data collection. Figure 4 compares the expert-data and β-DATA policies from the same initial state.



Figure 10: **Data collection in simulation.** A human operator continues packing as simulated perturbations add objects and displace packed objects onto the table. The operator responds by repacking the cracker box and lifting the cup, then completes the seven-object goal. Dashed boxes identify relevant objects; curved arrows indicate movement direction rather than measured trajectories.

I REAL-ROBOT PACKING DATA COLLECTION AND POLICY ROLLOUTS

Adversarial demonstration collection. Figure 11 illustrates the operator–adversary protocol during real-world packing.

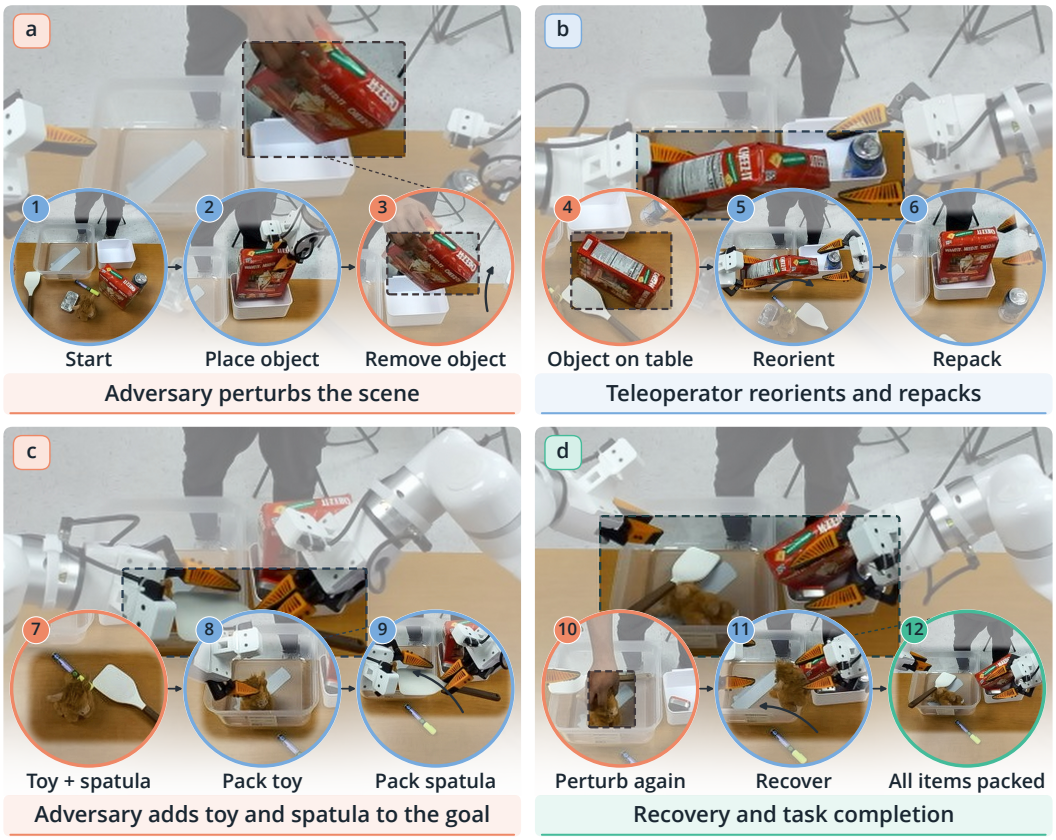

Figure 11: **Perturbation and recovery during real-world data collection.** In one teleoperated episode, the adversary removes packed objects and adds the toy and spatula to the goal. The teleoperator recovers and completes the updated packing task. Numbered circles show selected moments in chronological order.

Before and after online finetuning. Figure 13 compares selected rollouts of β-DATA at initialization and after online finetuning; both rows use the same midtraining source.

Online training and trajectory selection. Each run collects twenty accepted successful trajectories and performs twenty gradient updates after each pair. The batch size is 256: 192 examples come from the policy's own demonstrations and 64 from online successes. Once earlier successes are available, the online portion contains 32 examples from the newest pair and 32 from the earlier buffer; the first update block draws all online examples from the new pair. The learning rate increases to 3×10^{-5} over ten warmup updates and remains constant thereafter.



Figure 12: **Expert-data and β-DATA policies on real-xArm packing.** Both policies are evaluated zero shot. We observe the expert data policy making repetititve grasp attempt on the chip bag while it is instructed to pack the yak, ending with the yak unpacked. On the other hand, β-DATA (bottom) packs the yak on its first attempt, recovers from chip-bag drops, and completes the task. Dashed boxes highlight relevant objects; arrows indicate schematic movement directions.

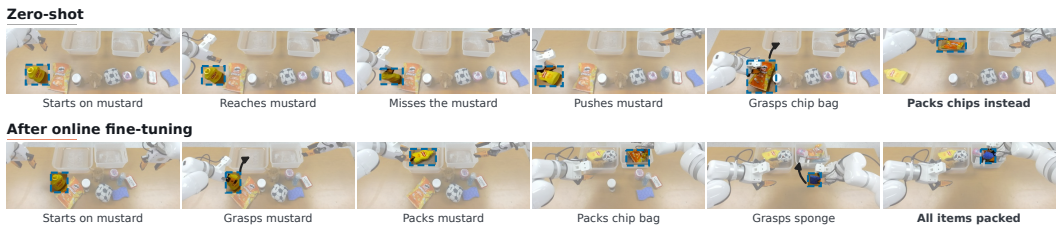

Figure 13: **Real-xArm packing before and after online finetuning.** Both rows show β-DATA. The zero-shot policy (top) fails to grasp the mustard and packs the chip bag while instructed to pack the mustard. After online updates, we observe that the policy (bottom) packs the mustard and continues until all items are packed.

J LEGO DATA COLLECTION AND POLICY ROLLOUTS

Figure 14 shows how the operator responds when the adversary rejoins or relocates bricks during collection. Figure 15 illustrates the policies' different grasping and disassembly behavior in selected evaluation episodes.

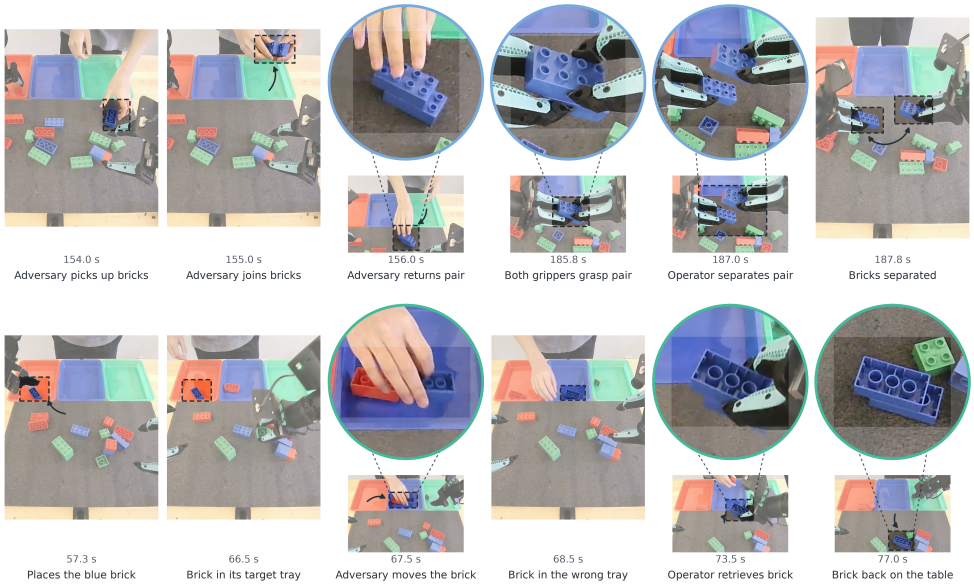

Figure 14: **Adversarial collection for LEGO disassembly and sorting.** Top: the adversary joins two bricks, and the teleoperator separates them using both arms. Bottom: the adversary moves a placed brick into another tray, and the operator retrieves it. The rows show separate selected sequences. Dashed boxes identify relevant bricks; arrows indicate schematic movement directions.

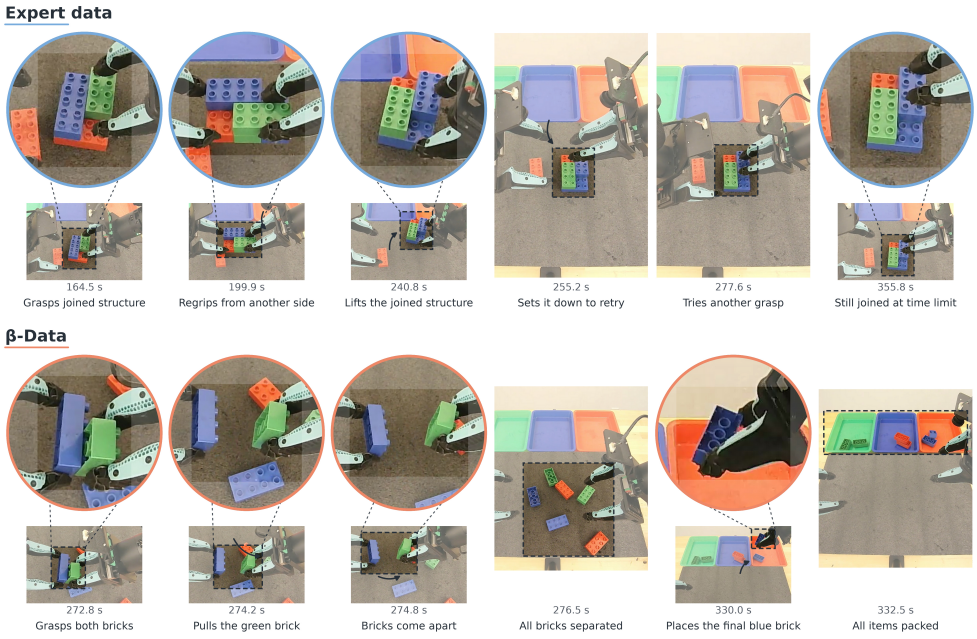

Figure 15: **Selected LEGO disassembly and sorting rollouts.** The expert-data policy (top) repeatedly regrasps the structure but leaves its bricks connected. β-DATA (bottom) separates the bricks and places the final brick to complete sorting. The episodes have different initial configurations. Dashed boxes highlight relevant bricks; arrows indicate schematic movement directions.