REINFORCEMENT LEARNING ENABLES SCALING GENERALIST ROBOT POLICY IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

The introduction of generalist robot policies has enabled a single, large model to learn a variety of tasks from a vast pre-training corpus. While these models often display powerful zero-shot generalization capabilities, they nonetheless typically require an adaptation phase to reach sufficient performance in deployment. Iterated self-improvement via reinforcement learning (RL) has demonstrated promise in enabling this adaptation, yet such approaches have been largely explored in *single-task* settings – where the goal is to simply improve on a narrow deployment task – which lies in stark contrast to the *multi-task* capabilities of the base policy. In this work we ask whether generalist robot policies are capable of multi-task, *generalist improvement in deployment* – targeting many diverse tasks simultaneously – and what algorithmic recipe enables this. To answer this question, we first develop a set of structured evaluations to test generalist improvement capabilities, and systematically evaluate algorithmic and practical design choices regarding value estimation, policy extraction and resource allocation. We show that not only can generalist policies be improved online across diverse tasks, but that the right recipe can enable *generalization* and increase performance on tasks not considered during online training. We characterize when this generalization occurs, and show that the training task distribution as well as the use of value-based RL are both critical. To the best of our knowledge, this is the first evidence for online generalist improvement for robot foundation models.

1 INTRODUCTION

Generalist robot policies, particularly Vision-Language-Action Models (VLAs), have shown promise in enabling general-purpose robotic capabilities, allowing a single policy to be deployed across diverse tasks (Brohan et al., 2023; Octo Model Team et al., 2024; Black et al., 2025; NVIDIA et al., 2025). However, despite encoding diverse task-solving behaviors, such policies often lack sufficient reliability for robust deployment, and a degree of adaptation is typically necessary for effective performance (Kim et al., 2025; Intelligence et al., 2025b; Reuss et al., 2025). While enabling this adaptation through human supervision is often prohibitively expensive, autonomous adaptation through reinforcement learning (RL) has emerged as a promising approach to scale policy improvement, enabling robots to learn from their own experience in deployment (Intelligence et al., 2025a). Yet, such autonomous improvement approaches are typically limited to narrow *single-task* deployments. As such, they enable generalist policies to improve, but often specialize them to the particular deployment task considered, while degrading on all other tasks, in stark contrast with the generalist nature of the base policy.

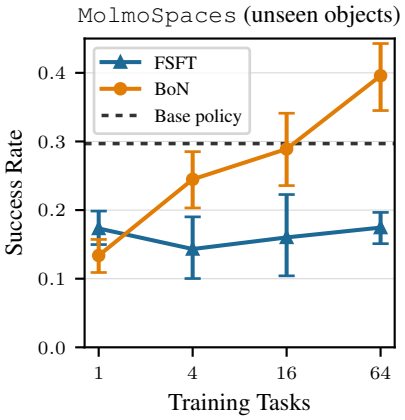

Figure 1: **Scaling the number of RL training tasks enables generalist improvement of generalist robot policies.** As training task diversity increases, critic-guided refinement (BoN) prevents forgetting and improves performance on unseen tasks.

A generalist policy, however, can produce a wide range of behaviors across diverse tasks, even if these behaviors may be far from optimal. Ideally, we might hope that, by deploying on such a diverse

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.

range of tasks, we could improve the performance of the policy *across all tasks simultaneously* - boosting its performance not just on a single deployment task, but on the entire distribution of tasks of interest. This improvement should span, first, across the diverse deployment tasks; furthermore, to obtain a truly *generalist improvement*, we must enable *generalization* – improvement beyond the training tasks, increasing performance in future novel deployment settings. This motivates the central question we consider in this work: ***how can we leverage interaction across a diverse set of tasks to enable online generalist improvement of generalist policies?*** That is, can a careful adaptation phase over multiple tasks enable both (i) improvement across these tasks simultaneously, and (ii) generalization beyond them, improving performance on tasks not encountered during post-training?

In this paper, we undertake a systematic study to **quantify generalist policy improvement** across different axes of generalization. We show that appropriate recipes, when instantiated carefully, can enable not only generalist policy improvement across a diverse set of training tasks, but also **task-generalization** – improvement on tasks not seen during training. Further, we show that this improvement *scales* with the number and diversity of tasks trained on, as we showcase in Figure 1. To the best of our knowledge, this is the first evidence that online post-training of generalist policies generalizes to novel tasks in robotic manipulation. Furthermore, our results suggest that moving beyond the single-task paradigm is not only possible but also crucial for achieving broader improvement.

We isolate the **essential ingredients** that enable multi-task learning and generalization. We find that a form of *critic-guided refinement*, i.e., adaptation based on a learned return estimate, is critical to achieve this. Specifically, we show that learning a critic network, and simply selecting robot actions by Best-of-N sampling (BoN, Nakamoto et al., 2024; Giridhar et al., 2026) with respect to that critic's induced action ranking, enables generalization across tasks. In contrast, we find that critic-free *direct policy adaptation*, instantiated by filtered supervised fine-tuning (FSFT, Peters & Schaal, 2007; Zhou et al., 2024; Qin & Springenberg, 2025) on successful trajectories, fails to support significant generalization. We also study the promises and limitations of methods which combine policy and critic learning, including algorithms that update the policy to maximize rewards predicted by the critic (Peng et al., 2019; Mark et al., 2024; Patil et al., 2026).

Lastly, we systematically evaluate the **key design choices** necessary to enable generalist policy improvement across algorithm design, data curriculum and resource allocation. Through systematic experiments, we investigate how different value learning approaches (i.e., Monte Carlo, temporal difference), architectures and objectives compare in multi-task settings, and how these values can most effectively guide the extraction and improvement of a policy. Furthermore, by controlling training and evaluation data distribution, we study which axes of diversity are most helpful for generalization. Finally, we evaluate the role of iterated improvement, and study tradeoffs in allocating compute and data collection.

2 Related Work

VLA post-training. Diffusion and flow-matching provide expressive generative policy classes that, when trained through behavior cloning at sufficient scale and diversity, can form the basis of generalist policies (Janner et al., 2022; Chi et al., 2023; Reed et al., 2022). Recent offline-to-online RL methods directly optimize such policies (Psenka et al., 2024; Park et al., 2025b; Wagenmaker et al., 2025; Patil et al., 2026; Li & Levine, 2026), but scaling them to large VLAs remains challenging. Existing VLA post-training therefore often relies on simpler, readily scalable methods. One family directly updates the policy using high-quality experience, including filtered supervised fine-tuning (Qin & Springenberg, 2025; Yang et al., 2026), conditioned behavior cloning (Intelligence et al., 2025a), and soft variants that weight demonstrations by estimated value (Peng et al., 2019; Su et al., 2026). A complementary family leaves the policy unchanged and instead uses learned value functions for test-time action selection, such as Best-of-N (Nakamoto et al., 2024; Dodeja et al., 2026). We likewise focus on simple, scalable techniques, demonstrating (cf. Section 4) that they can achieve competitive performance. Existing VLA post-training has demonstrated substantial performance gains (Zhang et al., 2024; Hu et al., 2025; Ankile et al., 2025; Myers et al., 2024; Li et al., 2025; Chen et al., 2025; Dong et al., 2026b; Guo et al., 2025; Lu et al., 2025; Sun & Song, 2025), but typically studies improvement on individual tasks (Xu et al., 2026; Pan et al., 2026; NVIDIA et al., 2025; Team et al., 2026). Related approaches distill collections of smaller specialist policies into a foundation-model

policy, thereby improving performance on the corresponding *in-distribution* tasks (Xu et al., 2025; Xiao et al., 2026). In contrast, we study joint post-training across increasingly broad task distributions and ask whether the resulting policy also improves on tasks outside the training distribution.

Multi-task generalization. A separate literature studies how robotic performance scales with the diversity of training tasks. Prior work considers multi-task scaling without foundation-model policies (Nauman et al., 2025), while meta-reinforcement learning instead develops algorithms that *learn to rapidly adapt* to unseen tasks (rather than to directly transfer a fixed policy to new tasks, the setting we consider), traditionally with relatively small policies (Duan et al., 2016; Finn et al., 2017; Yu et al., 2019). More recent work extends this perspective to VLA policies through behavior cloning (Bianchi et al., 2026). Other works study generalization arising from scaling behavior cloning during pretraining (Zheng et al., 2026; Dyna Robotics, 2026) or post-training (Lin et al., 2025; 2026). Here we focus on self-improvement via reinforcement learning. Perhaps closest to our setting is Liu et al. (2025), who compare generalization of offline and off-policy SFT with with on-policy reinforcement-learning algorithms such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024). Our study differs in two important ways. First, while we study SFT for post-training too, we focus on *online* self-improvement, rather than using more demonstration data in post-training. Second, instead of comparing generalization capabilities of off-policy SFT and on-policy RL, this work studies whether, and how, generalist RL policies acquire out-of-distribution generalization as the number and diversity of in-distribution training tasks increase. Multi-task generalization is also evident in LLM post-training: we draw parallels in Appendix A.

3 Problem Setting and Experimental Methodology

3.1 Problem Setting: Generalist Improvement of Generalist Policies

We model our environment as a contextual Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, P, R, \mu_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are continuous state and action spaces, $\mathcal{C}$ is a space of contexts encoding different tasks[1], $P : \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to \Delta(\mathcal{S})$ is a stochastic transition function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to [R_{\min}, R_{\max}]$ is a context-conditional reward function specifying each task, $\mu_0 : \mathcal{C} \to \Delta(\mathcal{S})$ is an initial distribution and $\gamma \in [0, 1)$ is a discount factor. We model a generalist agent as a context-conditional policy $\pi : \mathcal{S} \times \mathcal{C} \to \Delta(\mathcal{A})$[2], and evaluate its outcome from a state-action pair (s, a) and task c according to its Q-function:

$$Q^\pi(s, a, c) = \mathbb{E}_{P,\pi,c}\left[\sum_{t\geq 0} \gamma^t R(s_t, a_t, c) \mid s_0 = s, a_0 = a\right], \tag{1}$$

where above, $\mathbb{E}_{P,\pi,c}$ denotes expectation with respect to $a_t \sim \pi(\cdot \mid s_t, c)$ and $s_{t+1} \sim P(\cdot \mid s_t, a_t, c)$. We denote $V^\pi(s, c) = \mathbb{E}_{a\sim\pi(\cdot \mid s,c)}[Q^\pi(s, a, c)]$. Given a task distribution $\rho_{\text{test}} \in \Delta(\mathcal{C})$, the **objective of generalist improvement of generalist policies** (which we will refer to as *generalist improvement* for brevity) is to update π so as to maximize:

$$J(\pi; \rho_{\text{test}}) = \mathbb{E}_{c\sim\rho_{\text{test}}, s_0\sim\mu_0(c)}\left[V^\pi(s_0, c)\right]. \tag{2}$$

Generalist improvement protocol. In this work we assume we are given some initial generalist policy π_{gen}, achieving non-zero but imperfect zero-shot success across tasks, and can interact online with a set of training tasks drawn from a distribution ρ_{train}, with the goal of learning a fine-tuned policy π_{ft} that maximizes $J(\pi_{\text{ft}}; \rho_{\text{test}})$. Crucially, ρ_{test} need not coincide with ρ_{train}: the policy needs to be able to *transfer* to tasks it was not trained on. At the same time, these two distributions need to be related (see Figure 3), as generalization to arbitrary tasks is in practice unrealistic (see Appendix C.1, Yu et al., 2020 and Malik et al. (2021)). For simplicity, we will assume that ρ_{train} is supported on a finite set of tasks $[c_1, \ldots, c_{N_{\text{task}}}]$. We consider interaction with the training tasks through repeated iterations $m = 1, \ldots, M$, each involving online interaction and training. For each iteration m, we first collect K episodes $\tau_{1,1:K}, \ldots, \tau_{N_{\text{task}},1:K}$ from each training task $[c_1, \ldots, c_{N_{\text{task}}}]$, with the policy frozen, and then update the policy given the newly collected experience. This protocol approaches traditional, single-task online RL by choosing $\rho_{\text{train}} = \rho_{\text{test}}$ and setting $K = 1$, and tends towards multi-task offline RL (or "batch online RL", Dong et al.,

[1] In practice, a task will correspond to following a specific prompt in a specific manipulation environment.

[2] We overload notation and condition π on $c \in \mathcal{C}$ for simplicity; in practice, if π is a VLA, the task conditioning is provided in the form of a language command, and the policy outputs action chunks.

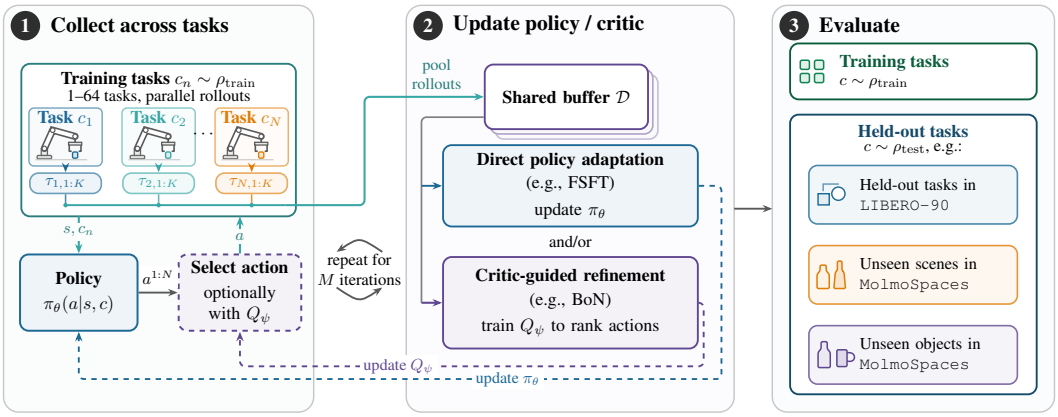

Figure 2: **Generalist improvement and evaluation. (1)** A generalist policy collects K rollouts $\tau_{n,k}$ across training tasks $c_n \sim \rho_{\text{train}}$. **(2)** Task-specific experience is pooled in a shared replay buffer, and used to enact either of two mechanisms: direct policy adaptation updates the policy, while critic-guided refinement learns critics for action ranking. This process is iterated M times. **(3)** Evaluation is carried out over training tasks ($c \sim \rho_{\text{train}}$), as well as on unseen test tasks ($c \sim \rho_{\text{test}}$) to test generalization.

2026b) when a single iteration is applied ($M = 1$, $K \gg 1$) and ρ_{train} has larger support. To simulate realistic fleet-level robot deployments, we choose a moderately large K and M, discussed in Section C.3. We visualize the protocol in Figure 2.

3.2 Mechanisms for Generalist Improvement

In this work, we aim to understand what algorithmic approaches enable generalist improvement. We distinguish between two complementary and independent mechanisms for policy improvement: *direct policy adaptation*, which updates the parameters of the robot policy alone, and *critic-guided refinement*, which instead learns a value function to "grade" robot actions. While many popular approaches combine both policy updates and critic learning (see Appendix B for a discussion), our study highlights methods that isolate each component and, in particular, we aim to study paradigmatic approaches that scale to modern VLAs.

Direct policy adaptation. As a representative and scalable approach to direct policy adaptation, we consider weighted supervised learning. At each iteration, we sample a batch of state-action-task tuples from the buffer, and define a weighting function $w : \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to \mathbb{R}$ in order to minimize a weighted negative log-likelihood under the policy:

$$\mathcal{L}_{\text{DPA}}(\theta) = -\mathbb{E}_{(s,a,c)\sim\mathcal{D}}\big[w(s,a,c)\log\pi_\theta(a|s,c)\big], \tag{3}$$

for policy parameters θ. When an explicit likelihood is not available, a tractable lower bound can be adopted (e.g., the flow matching loss in the case of modern VLAs (Intelligence et al., 2025b)). The simplest instantiation of this mechanism is referred to as %-BC (Chen et al., 2021) or filtered supervised fine-tuning (FSFT, Zhou et al., 2024), which computes $w(s_t, a_t, c) = \mathbf{1}_{G(s_t,a_t,c)>G_{\min}}$ where $G(s_t, a_t, c) = \sum_{t'\geq t} R(s_{t'}, a_{t'}, c)$ is the empirical return, and $G_{\min}$ a threshold. This mechanism implements a form of policy improvement (Qin & Springenberg, 2025) while isolating the policy update mechanism and conveniently remaining as close as possible to a purely supervised objective. Moreover, it remains representative and competitive with other algorithms recovered for different choices of w, as we discuss in Section 4.3.3.

Critic-guided refinement. Orthogonally to direct policy adaptation, improvement can also occur through utilizing a learned value function. We consider training a parametric state-action critic Q_ψ to estimate the state-action value function expressed in Equation 1. While many approaches for value estimation exist (and are ablated in Section 4.3.2), we find that training Q_ψ together with a state value function V_η through the following standard temporal difference (TD) learning update performs reliably:

$$\mathcal{L}_Q(\psi) = \mathbb{E}_{(s,a,r,s',c)\sim\mathcal{D}}\Big[\big(Q_\psi(s,a,c) - (r + \gamma V_\eta(s',c))\big)^2\Big] \quad \text{and} \tag{4}$$

$$\mathcal{L}_V(\eta) = \mathbb{E}_{(s,a,c)\sim\mathcal{D}}\Big[\big(Q_\psi(s,a,c) - V_\eta(s,c)\big)^2\Big]. \tag{5}$$

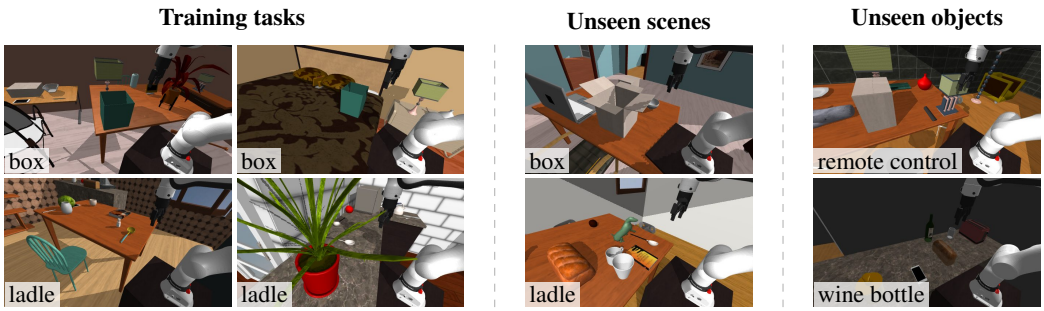

Figure 3: **Objects and scenes in `MolmoSpaces`**. Left: 4 tasks covering two objects, each with two scenes. Middle: unseen scenes of the training objects. Right: examples of unseen objects.

See Section 4.3.2 for ablations of our critic-learning objective (e.g., Monte Carlo returns and distributional critics). As our goal is to understand the mechanisms that enable generalist improvement, we wish to isolate the contribution of policy adaptation from critic refinement. To achieve this, we build on a long line of works on test-time action selection (Kalashnikov et al., 2018) that guide the policy via Best-of-N (BoN) sampling (Nakamoto et al., 2024). In particular, we keep the policy parameters frozen and re-rank sampled actions at test-time according to their estimated values. That is, for a task c and the current state s_t in a rollout, we draw N action candidates $\{a_t^i\}_{i=1}^N \sim \pi_\theta(s_t, c)$, score them and execute action chunk $a_t^{i^\star}$ with

$$i^\star = \arg\max_i Q_\psi(s_t, a_t^i, c). \tag{6}$$

This allows post-training experience to influence behavior through action selection while preserving the policy parameters inherited from pre-training; moreover, we found this recipe to be competitive with more complex schemes (see Section 4.3.3).

Combining policy adaptation and critic guidance. While our core analysis will focus on these two approaches, in conventional RL, critic values are often used to guide updates to the policy via either weighted supervised learning (Peng et al., 2019) or policy-gradient-like approaches (Lillicrap et al., 2016). To test combinations of policy- and critic-driven updates, we consider four representative approaches: (1) FSFT+BoN, which adapts the policy with $\mathcal{L}_{\text{DPA}}$ and filters action at test-time according to a critic trained with $\mathcal{L}_{\text{Q/V}}$, (2) flow-based AWR (Peng et al., 2019), where we use critic-derived estimates of the advantage function to reweight the supervised learning loss following Equation (3), (3) a variant of PARL (Mark et al., 2024), which applies the unweighted supervised objective in Equation (3) on actions re-ranked at training time, and (4) OGPO (Patil et al., 2026), which applies a policy-gradient style update through the flow process, with critic advantages as terminal rewards. We evaluate them in Section 4.3.3 and provide full details in Appendices B and D.

3.3 Experimental Setup

Our experiments are designed to answer the following core questions:

- **Q1:** As the diversity of training tasks increases, how effective are the two mechanisms (direct policy adaptation and critic-guided refinement) at driving self-improvement on the tasks they are exposed to?
- **Q2:** To what extent and at which scale does either mechanism generalize to unseen tasks?
- **Q3:** Which specific algorithmic design choices drive learning within and beyond the training task distribution, i.e., enable generalist improvement?

To answer these questions, we design a post-training and evaluation protocol centered on directly controlling the training and test task distributions ρ_{train} and ρ_{test} introduced in Section 3.1.

Training environments. To answer Q1 and Q2, we consider training distributions that differ in both the number of distinct tasks and their diversity. We base our evaluation on the `MolmoSpaces` (Kim et al., 2026) benchmark, which gives precise control over task diversity. We focus on the `pick` suite, which instructs the agent to grasp and lift various objects in different domestic environments. This benchmark exposes 18 task objects and 55-56 scenes per object (e.g., the "fork" object spans 56 distinct scenes), and we visualize examples in Figure 3. Two tasks can differ in both scene and object: while the former shift is largely visual in nature, the latter generally involves different

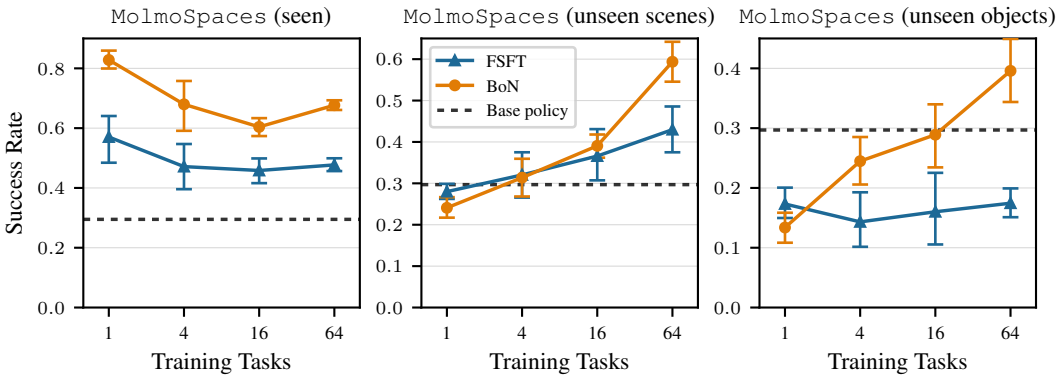

Figure 4: **Value-based generalist improvement generalizes beyond the training tasks.** Final success rate of FSFT and BoN on training tasks (left) and held-out tasks (center and right, unseen scenes and objects) in MolmoSpaces as the number of distinct training tasks increases (x-axis). BoN displays stronger generalization. We report the mean and 95% stratified bootstrap CIs across three seeds.

grasping strategies. In our main evaluation (Figure 4), we consider training task distributions over growing numbers of tasks, from 1 to 64, spanning different objects and scenes. Each set of tasks is procedurally selected to be a superset of smaller sets, while maintaining the same non-zero average performance of the base policy over tasks (≈30%). This ensures that larger sets of tasks are at least as diverse as smaller ones, as they cannot include fewer objects and scenes. Additional evaluations, e.g., in Figure 5, ablate the diversity in the set of training tasks, which may only vary in terms of scenes, or also in terms of objects. We additionally evaluate on the popular LIBERO (Liu et al., 2023) manipulation suite, for which tasks are sampled from the LIBERO-90 subset.

Evaluation tasks. To address Q1, we evaluate policy performance with respect to the training task distribution ($\rho_{\text{test}} = \rho_{\text{train}}$). To address Q2, we largely leverage MolmoSpaces's structured task diversity, varying task novelty across both the axes of task objects and scenes, described above and depicted in Figure 3. Similarly, we evaluate both training and unseen tasks in LIBERO to assess generalization capability when task diversity cannot be easily controlled. For a full description of task sets in MolmoSpaces we refer to Appendix D.1. In all experiments, we report the final mean success rate over three training seeds and 95% bootstrap CIs; see Appendix E for training curves.

Base model, critic architecture, and policy improvement protocol. Across experiments, we use the $\pi_{0.5}$ VLA (Intelligence et al., 2025b) as the base model, in its droid and libero checkpoints for MolmoSpaces and LIBERO, respectively. Method-specific hyperparameters are optimized individually, and main results are stated for the best-performing hyperparameters for each approach. We briefly recap them here, and present detailed allocation studies and ablations justifying these decisions in Section 4.3 and Appendix C. We implement our critic networks as BroNet (Nauman et al., 2024) residual MLPs whose input is the concatenation of the $\pi_{0.5}$ prefix representation, mean-pooled over the prefix token sequence, the proprioceptive state, and the flattened action chunk. We use an ensemble of two Q and V-heads and aggregate them by taking the minimum (Fujimoto et al., 2018), both when forming the TD target in Equation (4) and when scoring candidate actions at inference time via Equation (6). In each experiment, we apply $M = 10$ iterations of policy improvement, collecting $K = 20$ episodes per task per epoch. In Section C.3, we find that task performance does not improve much for larger K and M. We fine-tune policies for 100 gradient steps in single-task settings, and 500 gradient steps in multi-task settings due to greater data scale and variability. The critic MLPs, being both smaller and trained from scratch, are trained with 100× gradient steps (reaching convergence, and roughly matching the training cost of policy adaptation).

We ensure reproducibility by open-sourcing our full pipeline at this anonymized repository.

4 Generalist Improvement Scales With Task Diversity

We seek to answer our three core questions about generalist policy improvement. Our analysis centers on Figure 4, in which each point corresponds to a training split of k objects with k scenes each, evaluated on its own training tasks (left) and on a fixed set of eight unseen scenes and eight unseen objects (center and right). Appendix D.1 lists every split in full.

4.1 Q1: Can we enable generalist improvement on training tasks?

The left panel of Figure 4 evaluates to what extent post-training can improve performance on the tasks sampled from the training task distribution. For both FSFT and BoN, we observe that success rate increases with respect to the base policy, and generalist improvement succeeds. We find BoN to be generally more effective than FSFT in the single-task setting, where BoN leads by $+26$ pp. Perhaps more interestingly, we can focus on the success rates as the number of training tasks increases. We find that BoN outperforms FSFT by between $+14$ pp and $+21$ pp in the multi-task setting. Neither of the recipes displays fully convincing cross-task expertise: while potentially overfitting to a specific task, the average performance of single-task experts exceeds that of multi-task policies. We observe similar trends on training tasks for `LIBERO`, in which the performance gap with respect to single-task experts is perhaps more marked, particularly for BoN – see Appendix C.1.

4.2 Q2: Do we see generalization beyond training tasks?

A different pattern emerges when, instead of measuring performance on the tasks we train on, we turn to evaluating success rate on unseen tasks in `MolmoSpaces`. In this setting, we find, to the best of our knowledge, ***the first examples of generalization in VLA RL post-training at this scale.*** As shown in Figure 4 (center and right), single-task post-training generally leads to forgetting – while we can improve on the task we train on, performance degrades on others. However, as we scale the number of training tasks, BoN exhibits a transition from forgetting to generalization. Specifically, we observe performance gains over the base policy of up to $+30$ pp for tasks where the object remains consistent but the scene changes from the training task, and $+10$ pp for tasks where the object also changes. This indicates that BoN can even handle different objects altogether (e.g., picking up a knife after training only on other items, such as ladles, pots, and cups). More generally, these results highlight that post-training VLAs on sufficiently diverse tasks can yield improvements *beyond* the training distribution. In particular, this improvement is much more pronounced for BoN than FSFT, demonstrating that critic-guided refinement is critical for enabling generalization (see Section 4.3.3 for analysis of approaches the combine both). We additionally evaluate generalization in `LIBERO` in Appendix C.1, in which we find that this benchmark presents a much greater challenge in terms of generalization, as inter-task variability is extreme: in this case, similar patterns occur, but only in terms of forgetting.

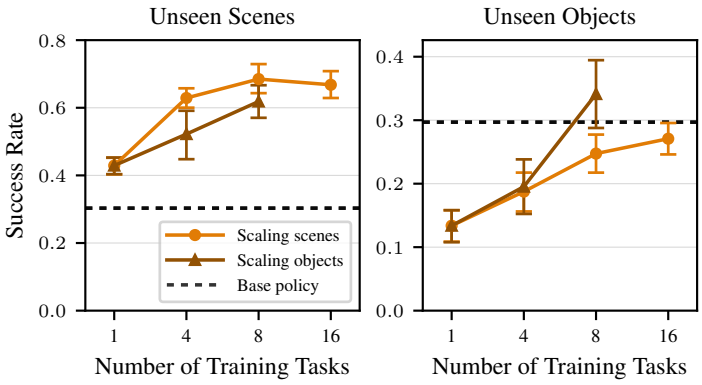

Figure 5: **Diversification transfers better along its own axis.** BoN success rate when scaling scenes of a single object, or objects with one scene each, evaluated on eight unseen scenes (*left*) and eight unseen objects (*right*). Each axis transfers best along itself, though only scaling objects surpasses the base policy on both. Mean and 95% stratified bootstrap CIs over three seeds.

To better understand where the generalization on `MolmoSpaces` comes from, we isolate the two axes – scene and object – of task diversity. In Figure 4 they grow together: a split with k^2 tasks contains k objects with k scenes each. In this experiment, we instead vary the number of scenes of a single object, and separately the number of objects with one scene each. Both are evaluated on eight unseen objects, and on eight unseen scenes of the object whose scenes the first sweep scales (Details in Appendix C.2). As shown in Figure 5, we observe greater generalization to new scenes by training on more diverse scenes, and greater generalization to new objects by training on more diverse objects. The two are not symmetric, however: scaling objects enables the post-trained policy to surpass the base policy on both unseen objects and scenes, whereas scaling scenes never exceeds it on unseen objects, even with 16 scenes. This illustrates that, to achieve generalization to novel manipulation strategies (e.g., the grasp mechanics necessary to grasp a new object), it is necessary

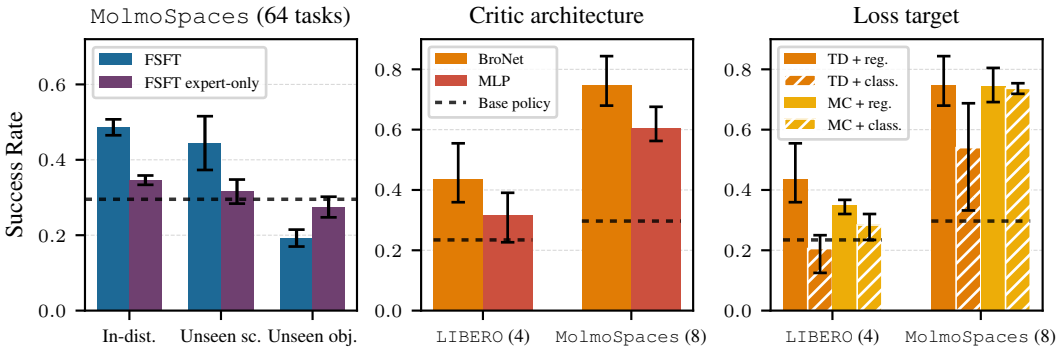

Figure 6: **Recipe ablations.** Left: full VLA vs. expert-only tuning for FSFT on MolmoSpaces (64 tasks). Full tuning is essential in-distribution and on unseen scenes, but degrades most on unseen objects. Center: architecture for BoN. Right: TD vs MC, classification vs regression for BoN.

to train on tasks requiring a diversity of manipulation behaviors, and learning on tasks requiring diverse manipulation behaviors can transfer to robustness over the scene layout and appearance.

4.3 Q3: What enables generalist policy improvement?

Having evaluated our recipes at scale, we now validate key design decisions for multi-task VLA post-training, and investigate several techniques combining policy adaptation and critic guidance; see Appendix C for additional ablations including data/compute scaling and resource allocation.

4.3.1 What Makes Direct Policy Adaptation Effective?

An important fine-tuning decision concerns which parameters to update: the entire VLA or only the action expert, i.e., the flow model mapping tokens from the backbone to continuous actions. We compare both settings for FSFT on the MolmoSpaces (64 tasks) split, evaluating in-distribution, on unseen scenes, and on strongly OOD unseen objects. The left panel of Figure 6 shows that ***full VLA tuning is beneficial***: it outperforms expert-only tuning by more than $+12$ pp both in-distribution and on unseen scenes, whereas expert-only tuning improves only marginally over the base policy. This comes at a cost: on unseen objects, full VLA tuning degrades performance by 10.6 pp relative to the base policy, against -2.2 pp for expert-only tuning. This suggests that full finetuning increases overfitting risks. The two settings therefore trade in-distribution gains against OOD performance.

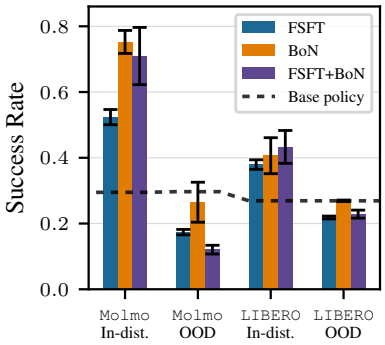

Figure 7: **Combining FSFT and BoN is not straightforward.** FSFT+BoN can bring marginal gains on the training tasks, but not on unseen ones. Mean success rate over three seeds, averaged across two task splits.

4.3.2 What Makes a Good Value Function in VLA Post-Training?

Value function learning admits many possible implementations. To find a scalable recipe for multi-task VLA post-training, we conduct controlled ablations along multiple design axes, focusing on training-tasks improvement. Figure 6 (center, right) compares two network architectures, regression and classification objectives, as well as TD and MC targets. Compared to standard MLP critics, adding **residual connections** (BroNet; Nauman et al., 2024) increases performance by more than $+10$ pp. Regarding the training objective, combining **TD targets with a simple regression loss** yields the largest and most consistent improvements in our experiments. Notably, we find distributional critic formulations to scale poorly to multi-task settings; we hypothesize this is due to return distributions of different tasks varying significantly. Finally, Best-of-N introduces one further hyperparameter, the number of sampled candidates N. We ablate its impact on performance and inference time in Appendix C.7, and use $N = 32$ throughout.

4.3.3 Combining Policy Adaptation and Critic Guidance

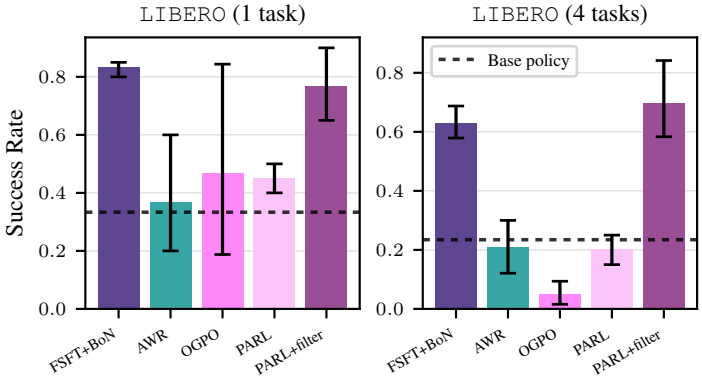

Figure 8: **Filtering matters more than the extraction objective.** FSFT+BoN outperforms AWR and PARL on LIBERO (1 task, *left*; 4 tasks, *right*), while adding FSFT's filtering step to PARL matches the FSFT+BoN performance. Mean training-task success rate over three seeds, 95% stratified bootstrap CIs. Full learning curves in Appendix E.3.

Direct policy adaptation (FSFT) and critic-guided refinement are both capable of driving policy improvement, and can be naturally combined. As a straightforward instantiation of such an algorithm, we consider FSFT+BoN, which simultaneously fine-tunes the policy on successful rollouts and trains a critic for inference-time action selection. Figure 7 shows the in-distribution and out-of-distribution performance in both benchmarks, averaged across task splits $N_{\text{task}} \in \{1, 64\}$ (in the case of MolmoSpaces, we evaluate unseen objects). The combination of FSFT and BoN can bring marginal gains on training tasks, but not on unseen ones: its generalization capabilities do not exceed those of pure FSFT and BoN.

While FSFT+BoN is perhaps the simplest instantiation of a recipe that uses the value estimates to update the policy, it also seems to be the most stable in the multi-task setting, as shown in Figure 8, which compares it to other post-training techniques leveraging both policy adaptation and critic guidance (see Appendix B for a full description) in the setting in which hybrid techniques appear the most promising (1 or 4 LIBERO tasks). It outperforms AWR, OGPO and PARL by more than $+35$ pp, both in single-task and multi-task settings. The poor performance of AWR, OGPO, and PARL suggests that, while critic estimates may be sufficiently accurate for test-time action ranking, they may not be accurate enough for weighting actions in policy updates, e.g., in Equation (3), particularly for arbitrary trajectories. Moreover, we often find that first or zero order critic gradients do not help resolve issues of language instruction following, as evidenced by findings in Appendix C.4. We additionally evaluate a version of PARL which only updates its policy on successful trajectories, much like FSFT. This version closes much of this gap, indicating that value estimates on unsuccessful trajectories are indeed less reliable, and filtering them out is an important component for stability.

5 Discussion

This work studies the adaptation regimes of generalist policies as the diversity of their set of fine-tuning tasks increases. We isolate two orthogonal techniques for self-improvement, centered respectively on updating the policy parameters, and on training value estimates. While both mechanisms lead to improvements over training tasks, we find that, at a sufficient scale, value-based RL is instrumental in inducing a degree of generalization beyond observed tasks, and to achieve generalist policy improvement.

Limitations. Our contributions are empirical in nature. As such, while we dedicated significant effort and compute to broadening the scope of our evaluation, it remains limited to manipulation environments, which currently form a major domain of application for VLAs. Further studies on loco-manipulation and whole-body control (NVIDIA et al., 2025) remain important directions for future work. Moreover, the evaluation was designed with the constraint that the base policy (in our case, $\pi_{0.5}$) would make non-trivial progress on training tasks out of the box: for this reason, including tasks of arbitrary complexity would be less informative. Finally, given the large scale of our evaluation, and the number of diverse tasks required, it remains constrained to simulated environments.

The main goal of this work is thus to contribute to the understanding of mechanisms and trends of online self-improvement in dynamical systems. We believe that our results represent a promising starting point for future works, which can bootstrap off the simple and scalable RL recipes we derived, and study the benefits of multi-task post-training at scale on real hardware.

AI USE STATEMENT

Among mandatory disclosure tasks, we used generative AI tools for coding assistance, both for implementations and for supporting qualitative and thematic data analysis. We have not used generative AI tools to generate, clean or reformat datasets, to help develop conceptual frameworks, to design or provide feedback on research methodology or experiments, nor to interpret results. Other categories do not apply: our work does not involve translation, and we did not develop theoretical models nor formulate mathematical claims and proofs.

Additionally, we used generative AI tools for the following tasks with recommended disclosure: creating and modifying scientific figures, identifying relevant literature, suggesting edits for improved readability. We did not use generative AI for tasks in other categories with recommended disclosure.

We have reviewed all AI-assisted work: code for both implementations and plotting was reviewed systematically, and suggestions regarding related works and readability were only incorporated upon human examination. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our research abides by the ICLR Code of Ethics, and does not involve human subjects or potentially harmful physical deployment. While we focus on algorithmic components, we acknowledge that robot learning is generally entangled with topics of greater societal concern, particularly regarding large-scale automation, and the consequences that it involves.

REPRODUCIBILITY STATEMENT

We ensure reproducibility by releasing code for our full training pipeline at the URL listed at the end of Section 3.3. All data it uses is collected online, and no offline dataset is necessary to reproduce results. Moreover, we thoroughly report training details, hyperparameters and task splits in Appendix D.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *NeurIPS*, 2021.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement – residual RL for precise assembly. *ICRA*, 2025.

Christian Bianchi, Siamak Yousefi, Alessio Sampieri, Andrea Roberti, Luca Rigazio, Fabio Galasso, and Luca Franco. Robotic policy adaptation via weight-space meta-learning. *arXiv preprint arXiv:2606.07217*, 2026.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *RSS*, 2025.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. *RSS*, 2023.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *NeurIPS*, 2021.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. ConRFT: A reinforced fine-tuning method for VLA models via consistency policy. *RSS*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *RSS*, 2023.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *ICML*, 2025.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *JMLR*, 2024.

Lakshita Dodeja, Ondrej Biza, Shivam Vats, Stephen Hart, Stefanie Tellex, Robin Walters, Karl Schmeckpeper, and Thomas Weng. When life gives you bc, make q-functions: Extracting q-values from behavior cloning for on-robot reinforcement learning. *arXiv preprint arXiv:2605.05172*, 2026.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies. *ICLR*, 2026a.

Perry Dong, Suvir Mirchandani, Dorsa Sadigh, and Chelsea Finn. What matters for batch online reinforcement learning in robotics? *ICLR*, 2026b.

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.

Dyna Robotics. Dyna-2: A 1-million-hour scaling law for world-action models. August 2026. URL https://dyna.co/dyna-2.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *ICML*, 2017.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *ICML*, 2018.

Varun Giridhar, Anant Khandelwal, Jeremy A. Collins, Ignat Georgiev, and Animesh Garg. Beyond imitation: Self-improving robot policies via off-policy q-planning. *arXiv preprint arXiv:2608.21204*, 2026.

Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. *arXiv preprint arXiv:2501.16664*, 2025.

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

Jiaheng Hu, Rose Hendrix, Ali Farhadi, Aniruddha Kembhavi, Roberto Martin-Martin, Peter Stone, Kuo-Hao Zeng, and Kiana Ehsani. FLaRe: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning. *ICRA*, 2025.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *CoRL*, 2025b.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, Xian Li, Brian O'Horo, Gabriel Pereyra, Jeff Wang, Christopher Dewan, Asli Celikyilmaz, Luke Zettlemoyer, and Ves Stoyanov. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. *arXiv preprint arXiv:2212.12017*, 2022.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *ICML*, 2022.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRL*, 2018.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *RSS*, 2025.

Yejin Kim, Wilbert Pumacay, Omar Rayyan, Max Argus, Winson Han, Eli VanderBilt, Jordi Salvador, Abhay Deshpande, Rose Hendrix, Snehal Jauhri, Shuo Liu, Nur Muhammad Mahi Shafiullah, Maya Guru, Ainaz Eftekhar, Karen Farley, Donovan Clay, Jiafei Duan, Arjun Guru, Piper

Wolters, Alvaro Herrasti, Ying-Chun Lee, Georgia Chalvatzaki, Yuchen Cui, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmospaces: A large-scale open ecosystem for robot navigation and manipulation. *RSS*, 2026.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *ICLR*, 2022.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. *ICLR*, 2026.

Yunfei Li, Xiao Ma, Jiafeng Xu, Yu Cui, Zhongren Cui, Zhigang Han, Liqun Huang, Tao Kong, Yuxiao Liu, Hao Niu, Wanli Peng, Jingchao Qiao, Zeyu Ren, Haixin Shi, Zhi Su, Jiawen Tian, Yuyang Xiao, Shenyu Zhang, Liwei Zheng, Hang Li, and Yonghui Wu. GR-RL: Going dexterous and precise for long-horizon robotic manipulation. *arXiv preprint arXiv:2512.01801*, 2025.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ICLR*, 2016.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. *ICLR*, 2025.

Sixu Lin, Yunpeng Qing, Litao Liu, Ming Zhou, Ruixing Jin, Xiaoyi Fan, and Guiliang Liu. Dygrovla: Cross-task scaling of vision-language-action models via dynamic grouped residual optimization. *ICML*, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *NeurIPS*, 2023.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *NeurIPS*, 2025.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and Adam Roberts. The flan collection: Designing data and methods for effective instruction tuning. *ICML*, 2023.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. VLA-RL: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.

Dhruv Malik, Yuanzhi Li, and Pradeep Ravikumar. When is generalizable reinforcement learning tractable? *NeurIPS*, 2021.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.

Vivek Myers, Bill Chunyuan Zheng, Oier Mees, Sergey Levine, and Kuan Fang. Policy adaptation via language optimization: Decomposing tasks for few-shot imitation. *CoRL*, 2024.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *NeurIPS*, 2023.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *CoRL*, 2024.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. *NeurIPS*, 2024.

Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. *NeurIPS*, 2025.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *RSS*, 2024.

Mingjie Pan, Siyuan Feng, Qinglin Zhang, Xinchen Li, Jianheng Song, Chendi Qu, Yi Wang, Chuankang Li, Ziyu Xiong, Zhi Chen, Yi Liu, and Jianlan Luo. Sop: A scalable online post-training system for vision-language-action models. *arXiv preprint arXiv:2601.03044*, 2026.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. *ICLR*, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *ICML*, 2025b.

Sarvesh Patil, Mitsuhiko Nakamoto, Manan Agarwal, Shashwat Saxena, Jesse Zhang, Giri Anantharaman, Cleah Winston, Chaoyi Pan, Douglas Chen, Nai-Chieh Huang, Zeynep Temel, Oliver Kroemer, Sergey Levine, Abhishek Gupta, Hongkai Dai, Paarth Shah, and Max Simchowitz. Ogpo: Sample efficient full-finetuning of generative control policies. *arXiv preprint arXiv:2605.03065*, 2026.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. *ICML*, 2007.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *ICML*, 2024.

Chongli Qin and Jost Tobias Springenberg. Supervised fine tuning on curated data is reinforcement learning (and can be improved). *arXiv preprint arXiv:2507.12856*, 2025.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *TMLR*, 2022.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *ICLR*, 2025.

Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. FLOWER: Democratizing generalist robot policies with efficient vision-language-action flow policies. *CoRL*, 2025.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. *ICLR*, 2026.

Zhian Su, Weijie Kong, Haonan Dong, and Huixu Dong. Ig-rft: An interaction-guided rl framework for vla models in long-horizon robotic manipulation. *IEEE Transactions on Automation Science and Engineering*, 2026.

Zhanyi Sun and Shuran Song. From prior to pro: Efficient skill mastery via distribution contractive RL finetuning. *ICML*, 2025.

Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, Optimus Qin, Yifei Su, Nan Sun, Qiao Sun, Runze Suo, Heyun Wang, Yunhong Wang, Rujie Wu, Caoyu Xia, Lina Zhang, Jack Zhao, Guoliang Chen, Wenlong Chen, Xinze He, Bin Li, Qing Li, Zhuorong Li, Heng Qu, Wenxuan Song, Diyun Xiang, Yifan Xie, Peiran Xu, Hangjun Ye, Wen Ye, Han Zhao, and Quanyun Zhou. Xiaomi-robotics-1: Scaling vision-language-action models with over 100k hours of real-world trajectories. *arXiv preprint arXiv:2607.15330*, 2026.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *CoRL*, 2025.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *ICLR*, 20222.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Linxi Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual rl. *ICLR*, 2026.

Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. Rldg: Robotic generalist policy distillation via reinforcement learning. *RSS*, 2025.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. Rl token: Bootstrapping online rl with vision-language-action models. *arXiv preprint arXiv:2604.23073*, 2026.

Rushuai Yang, Hecheng Wang, Zhichao Wu, Chiming Liu, Xiaohan Yan, Xuan Du, Shuoyu Yue, Chuheng Zhang, Yunlong Wang, Yongcheng Liu, Lizhe Qi, Yi Chen, Wei Shan, and Maoqing Yao. Aloe: Action-level off-policy evaluation for vision-language-action model post-training. *arXiv preprint arXiv:2602.12691*, 2026.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *CoRL*, 2019.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *NeurIPS*, 2020.

Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. GRAPE: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*, 2024.

Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, Trevor Darrell, Furong Huang, Yuke Zhu, Danfei Xu, and Linxi Fan. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models. *CoRL*, 2024.

A Extended Related Work

Connections to LLM post-training. A closely related pattern to the ones we observe appears in LLM post-training. For instance, increasing the diversity of instruction-finetuning tasks improves generalization to held-out tasks (Wei et al., 20222; Iyer et al., 2022; Longpre et al., 2023; Chung et al., 2024), with evidence of approximately log-linear scaling with task coverage (Longpre et al., 2023) and stronger gains for larger models (Chung et al., 2024). These works primarily study supervised fine-tuning on fixed datasets, whereas we consider *online self-improvement* through reinforcement learning and study how its benefits scale with task diversity—a question that, to our knowledge, has not been systematically investigated in robotics. More directly, recent work comparing RL and supervised fine-tuning finds that RL can generalize better to unseen task variants (Chu et al., 2025) and better preserve previously acquired capabilities during post-training (Shenfeld et al., 2026). These observations relate to our results in Figure 4, where FSFT similarly exhibits weaker generalization and degraded performance on the training tasks compared to BoN.

B A Taxonomy of Post-Training Algorithms

Many algorithms have been proposed for finetuning pretrained generative control policies, such as Vision-Language-Action (VLA) models. We review a few representative approaches and connect them to the two policy improvement mechanisms identified in Section 3: direct policy adaptation and critic-guided refinement. As displayed in Figure 9, we distinguish whether these methods use value estimates and perform policy updates, and whether they use an unweighted supervised loss for policy extraction. As FSFT, BoN and FSFT+BoN are already introduced in Section 3, we focus on the remaining methods listed in Figure 9. This list is not meant to be exhaustive, but rather a representative comparison of existing methods. For simplicity, we will absorb the task conditioning in the state.

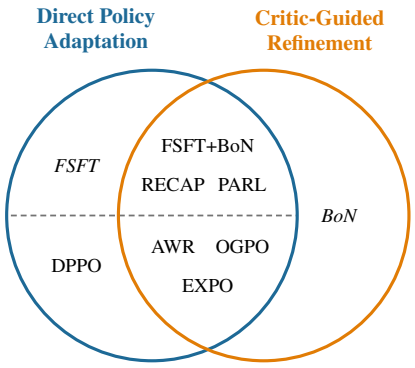

Figure 9: **A taxonomy of a few VLA post-training approaches.** Approaches above the dashed line adopt a purely supervised objective for policy extraction. We italicize the two mechanisms at the center of the analysis in Sections 4.1 and 4.2.

DPPO. Diffusion Policy Policy Optimization (Ren et al., 2025) is designed for generative policies involving a denoising process $a_{K:0} = (a_K, \dots, a_0)$: starting from $a_K \sim \mathcal{N}(0, \mathbf{I})$, the generative model samples $a_{k-1} \sim \pi_\theta(\cdot \mid s, a_k, k, c) \quad \forall k = K, \dots, 1$, until it arrives at the generated chunk a_0. DPPO applies on-policy PPO to a two-layer MDP, embedding the inner denoising MDP within the outer environment MDP. Policy gradients use tractable Gaussian likelihoods at each denoising step, with environment-step advantages estimated from on-policy rollouts and a state-value baseline. This allows policy extraction through simple PPO updates. For $k = 1, \dots, K$,

$$\omega_k(\theta) = \frac{\bar{\pi}_\theta(a_{k-1} \mid s, a_k, k)}{\bar{\pi}_{\theta_{\text{EMA}}}(a_{k-1} \mid s, a_k, k)}, \quad \widehat{A}_k = \gamma_{\text{den}}^{k-1} \widehat{A}^{\text{env}}, \tag{7}$$

$$\mathcal{L}_{\text{DPPO}}(\theta) = -\mathbb{E}\left[\sum_{k=1}^{K} C_\epsilon\Big(\omega_k(\theta), \widehat{A}_k\Big)\right], \tag{8}$$

where θ_{EMA} is an exponential moving average of the policy parameters, $\widehat{A}^{\text{env}}$ is estimated using environment rewards and a learned value baseline, γ_{den} discounts credit to earlier denoising transitions and the PPO surrogate is $C_\epsilon(\omega, A) = \min\{\omega A, \text{clip}(u, 1-\epsilon, 1+\epsilon)A\}$. We do not classify this method as leveraging critic-based guidance, as the learned value function is not action-conditional, and is mostly used for variance reduction rather than for evaluating actions. Nevertheless, it updates the policy parameters, and thus performs direct policy adaptation.

RECAP. RL with Experience and Corrections via Advantage-conditioned Policies (Intelligence et al., 2025a) learns a state-only distributional critic from Monte Carlo (MC) returns and converts estimated advantages into binary conditioning labels. The policy is then *conditioned* on these labels, and trained with a supervised flow matching loss on both successful and unsuccessful trajectories. With the original method's undiscounted n-step estimation, the advantage and the advantage labels for some state s_0 are expressed as

$$\widehat{A} = \sum_{j=0}^{n-1} r_{t+j} + V_\eta(s_n) - V_\eta(s_0), \quad I_t = \mathbf{1}\{\widehat{A} > \varepsilon\}, \tag{9}$$

for some threshold ϵ. As such, this method combines a form of critic-based guidance (through advantage conditioning) with policy adaptation through conditional behavior cloning.

AWR. Advantage Weighted Regression (Peng et al., 2019) is an actor-critic algorithm which avoids differentiation through the critic, and updates the actor through a weighted supervised objective. Its early instantiations rely on a critic trained through Monte Carlo return estimation, but

AWR has since been applied widely, e.g., with temporal-difference critics in offline RL (Park et al., 2025a). With the notation introduced in Section 3, the advantage-weighted policy extraction objective is

$$\mathcal{L}_{\text{AWR}}(\theta) = -\,\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[e^{\frac{\widehat{A}(s,a)}{\beta}}\log\pi_\theta(a|s)\right], \tag{10}$$

where $\widehat{A}$ is the advantage, often computed from $Q_\psi(s,a)$, and β is a temperature parameter that recovers pure behavior cloning as $\beta \to \infty$.

PARL. Policy Agnostic RL (Mark et al., 2024) trains a state-action critic $Q_\psi(s,a)$, originally with either IQL (Kostrikov et al., 2022) or Cal-QL (Nakamoto et al., 2023), while policy extraction relies on two steps: action optimization, and policy updates. The former consists of sampling action candidates from the current policy, and leveraging first- and zeroth-order optimization to approximately find $a^\star = \arg\max_{\mathcal{A}} Q_\psi(s,a)$ for each state and task in the training batch. The latter simply encourages the policy to regress the best candidate through unweighted behavior cloning (see Equation 3). PARL is thus a close neighbor of FSFT+BoN; PARL optimizes over actions at training-time instead of test-time, and trains its policy on both successful and unsuccessful trajectories.

EXPO. Expressive Policy Optimization (Dong et al., 2026a) combines a critic trained through TD learning with a policy improvement scheme that integrates an expressive base policy with a lightweight edit policy. The edit policy is trained to maximize Q-values with a SAC-style objective (Haarnoja et al., 2018), while the base policy is updated according to a simple behavior cloning loss. Crucially, EXPO integrates a BoN-like mechanism, both for selecting actions during evaluation, and for computing TD targets. Thus, EXPO is also closely connected to FSFT+BoN, mostly differing in the introduction of an edit policy, and in the slightly different recipe used for value learning.

OGPO. Off-Policy Generative Policy Optimization (OGPO) (Patil et al., 2026) decouples the bi-level MDP formulation proposed in Ren et al. (2025) and uses off-policy Q functions with TD-learning for the environment MDP, while using on-policy PPO-style policy extraction for the denoising MDP of the generative control policy. For each replay state from the replay buffer, OGPO rolls out a group of stochastic actions G to obtain group-relative advantages. This mitigates the need for first-order optimization through the Q-functions and allows zeroth-order updates by conditioning the likelihoods via the PPO objective keeping the critics frozen during policy extraction:

$$\widehat{A_i} = Q_{\text{targ}}(s, a_0^{(i)}) - \frac{1}{G}\sum_{j=1}^{G} Q_{\text{targ}}(s, a_0^{(j)}), \quad \omega_i(\theta) = \prod_{k=1}^{K} \frac{\bar{\pi}_\theta(a_{k-1} \mid s, a_k, k)}{\bar{\pi}_{\theta_{\text{EMA}}}(a_{k-1} \mid s, a_k, k)} \tag{11}$$

and

$$\mathcal{L}_{\text{OGPO}}(\theta) = -\mathbb{E}_{\mathcal{D},\theta_{\text{ref}}}\left[\frac{1}{B}\sum_{i=1}^{B} C_\epsilon\Big(\omega_i(\theta), \widehat{A_i}\Big)\right]. \tag{12}$$

Here, $\widehat{A_i}$ is the group relative advantage, $\omega_i(\theta)$ is the annealed importance sampling ratio, Q_{targ} is the EMA version of the online Q function, and similarly environment rollouts are conducted via π_{EMA}. OGPO proposes an ODE-to-SDE conversion of pretrained flow policies that allows the log probability updates on pretrained VLAs. Finally, OGPO proposes two key stabilization techniques: (1) Use of a success buffer for FSFT on the base objective, and (2) Use of *conservative advantages* that stabilize the gradient updates through underfit Q-functions in the beginning of RL-finetuning.

C Additional Empirical Results

C.1 Generalist improvement on LIBERO: does generalization occur for completely novel tasks?

We extend the evaluation reported in Figure 4 to LIBERO, in terms of performance on seen and unseen tasks. When evaluated on seen tasks (Figure 10, left), both direct policy adaptation (FSFT) and critic-guided refinement (BoN) are generally able to improve success rates with respect to the base policy. As observed in Figure 4, performance of multi-task models does not reach that of single-task experts, particularly for BoN, which barely improves over the base policy when trained over 64 tasks.

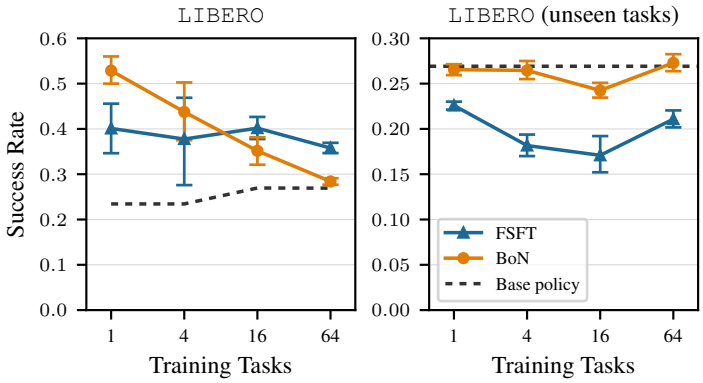

Figure 10: **Value-based generalist improvement is limited when tasks are loosely related.** Performance of FSFT and BoN on training tasks (left) and held-out tasks (right) in LIBERO as the training tasks grow in number. The single-task lead of BoN reverses in larger multi-task experiments, but FSFT exhibits substantial forgetting on held-out tasks, unlike BoN.

The evaluation on unseen tasks (Figure 10, right) is instead distinct from the one in Figure 4, and highlights the limits of generalization to unseen tasks. Here, the degree of diversity of tasks is harder to control, and the variability across tasks is higher. Unseen tasks often involve completely different setups, motion patterns and goals compared to the training tasks. At the scale we train on ($N_{\text{task}} \leq 64$), generalization does not emerge. Nevertheless, BoN remains preferable to FSFT, as it only displays minimal forgetting.

C.2 Extended Scaling Task Diversity Ablation

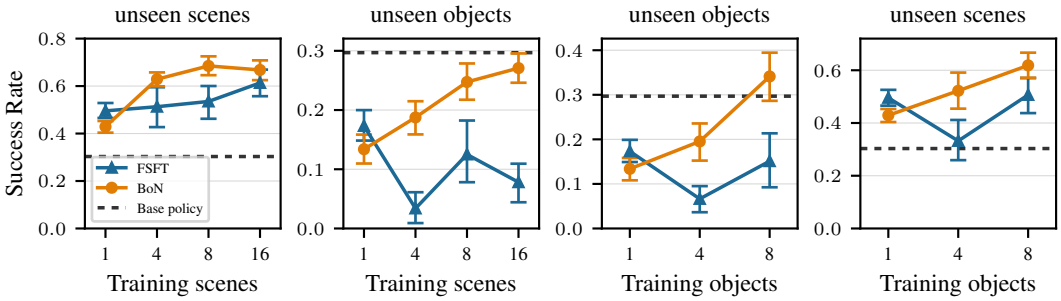

Figure 11: **Diversify along the axis of desired generalization.** Scaling scenes of a single object (*left two*) or objects with one scene each (*right two*), evaluated on unseen scenes and unseen objects. Each axis transfers best along itself, and BoN scales better than FSFT, which falls below the base policy on unseen objects. Mean over three seeds, 95% stratified bootstrap CIs.

We extend Figure 5 with FSFT and specify the task splits in full. We scale the number of scenes of a single object, $\{1, 4, 8, 16\}$, and separately the number of objects, $\{1, 4, 8\}$, with one scene each, evaluating both on eight held-out scenes of that object and on eight unseen objects. MolmoSpaces has 18 objects, of which we reserve 8 for evaluation, so at most 10 remain for training, explaining why the object sweep has no 16-object split. Table 1 lists the splits.

As shown in Figure 11, scaling scenes improves performance on unseen scenes by $+36.5$ pp for BoN and $+31$ pp for FSFT, but does not transfer across objects: both fall below the base policy there, although BoN recovers with more scenes while FSFT does not. Scaling objects eventually

transfers: BoN falls below the base policy at 1 and 4 objects, but overtakes it at 8. At a matched budget of eight tasks each axis transfers best along itself, by $+6.6$ pp on unseen scenes and $+9.4$ pp on unseen objects. It is therefore beneficial to diversify along the axis of desired generalization, though object diversity is more effective overall.

Table 1: **Task splits for the diversity ablation.** Task IDs are given without the `molmo_` prefix and grouped by object; *Init. SR* is the mean success rate of the base policy over a split. Scaling scenes varies the number of scenes of a single object; scaling objects varies the number of objects, with one scene each. Both regimes are evaluated on the same two held-out splits: eight unseen scenes of box, the one object common to every training split, and one scene of each of eight objects never seen during training. We note that the `MolmoSpace` (8 tasks) split is used in Figures 6 and 15.

Split	Tasks	Objects	Init. SR (%)
Scaling scenes			
1 task *box*: 267, 366, 138, 143	4	1	29.2
4 tasks *box*: 267, 366, 138, 143	4	1	29.2
8 tasks *box*: 164, 86, 70, 267, 366, 138, 143, 212	8	1	29.2
16 tasks *box*: 207, 393, 146, 55, 164, 86, 70, 267, 366, 138, 143, 212, 351, 333, 289, 122	16	1	33.9
Scaling objects			
1 task *box*: 267, 366, 138, 143	4	1	29.2
4 tasks *box*: 138, *cup*: 24, *tissuepaper*: 221, *spatula*: 373	4	4	29.2
8 tasks — `MolmoSpaces` (8 tasks) *boiler*: 369, *box*: 138, *cup*: 24, *ladle*: 187, *pot*: 325, *tissuepaper*: 221, *spatula*: 373, *spoon*: 148	8	8	29.7
Evaluation			
Unseen scenes *box*: 666, 602, 543, 273, 916, 505, 769, 675	8	1	30.3
Unseen objects *atomizer*: 140, *bowl*: 356, *fork*: 345, *knife*: 155, *remotecontrol*: 357, *salt/pepper shaker*: 163, *soapdispenser*: 208, *winebottle*: 226	8	8	29.7

C.3 Allocating resources: data and compute

Online post-training involves three important design axes when it comes to resource allocation: (i) how much data should be collected, (ii) how many gradient steps should be applied and (iii) how data collection and gradient updates should be interleaved. We thus consider these three axes, and validate the main experimental protocol outlined in Section 3.3: post-training for 10 iterations, each alternating data collection (20 episodes per task) and model updates (100-500 gradient steps for the policy, $100\times$ for the critic). While we present this ablation for BoN in `LIBERO`, we observed similar trends across algorithms and environments.

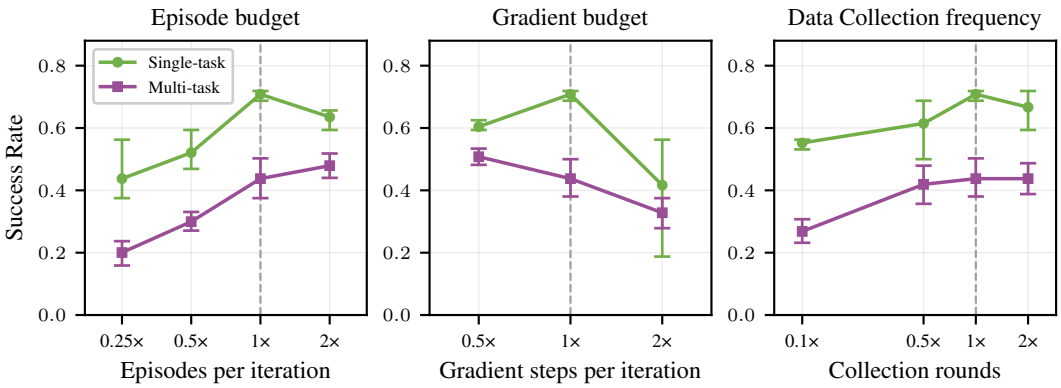

Figure 12: **Post-training benefits from more data and more frequent iterations.** Final success rates for critic-guided refinement (BoN) on LIBERO. Each subplot changes one hyperparameter: the number of episodes collected per iteration (left), the number of gradient steps per iteration (middle) and the number of iterations (right) over which a data and compute budget is distributed. The other hyperparameters are fixed per plot. $1\times$ marks the values used across the standard experimental protocol described in Section 3.3. Collecting more data is beneficial with diminishing gains, while more updates may be harmful; frequent iterations beat a single update phase after data collection.

As shown in the first column of Figure 12, we first study how performance changes with the amount of data collected per iteration, while keeping the number of iterations and of gradient steps fixed. We observe that in both single- and multi-task settings (adopting the 4-task split in this case), performance increases at a decreasing rate. We thus adopt the highest budget which brings significant benefits — 20 episodes per task. The second column keeps the number of episodes and iterations fixed instead, and ablates the amount of compute that should be allocated to training. In this case, we observe that more training is not always helpful, suggesting that the underlying training dynamics of online RL are not entirely stable. For these reasons, we settle on an intermediate budget. Finally, in the third column, we keep the amount of data and compute fixed, and we purely validate the number of iterations that should be applied. The leftmost point of the plot collects all the data at once, and follows this single data collection phase by a lengthy update phase, exhausting the entire gradient budget. The right most point instead distributes data and compute budgets across a large number of shorter iterations, each collecting $1/20$th of the data and applying $1/20$th of the gradient steps. Overall, we observe a gentle scaling trend which plateaus at high iteration frequencies, suggesting that frequent iterated online fine-tuning is beneficial. Accordingly, we post-train our base model over 10 iterations.

C.4 Instruction following as a Failure Mode

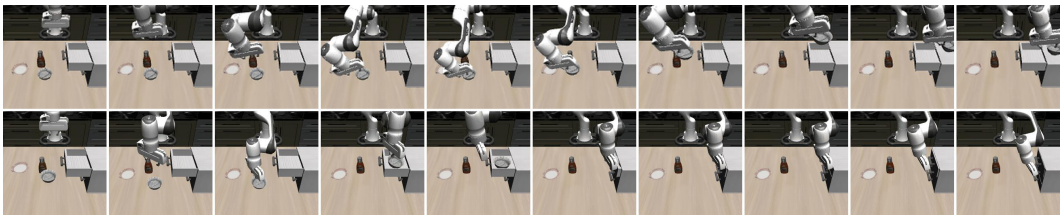

Figure 13: **Instruction following failure mode.** Two rollouts on LIBERO-31, where the instruction is to place the cup on top of the cupboard. *Top:* a successful rollout. *Bottom:* the dominant failure mode, in which the policy places the cup inside the drawer and closes it.

As shown in Figure 13, the OGPO policy reliably grasps the cup but then places it inside the drawer and closes it, executing a coherent but different task from the instructed one. This suggests that the backbone representations do not sufficiently disambiguate the language instruction.

C.5 Are multi-task critics beneficial for single-task learning?

A natural question in our setting is whether a critic pretrained on many tasks can accelerate value learning on unseen ones. We compare BoN with a critic pretrained on the `MolmoSpaces` (64 tasks) split against one trained from scratch, evaluating on unseen scenes of the pretrained critic's training objects and on unseen objects. Figure 14 shows that pretraining helps only initially: the pretrained critic starts at a substantially higher success rate, but the gap closes in the next iteration, after which it trails the randomly initialized critic on both splits. These results indicate that pretrained critics offer little benefit for accelerating value learning and can even be detrimental. One possible explanation is a loss of plasticity in the pretrained critic, though we leave the identification of the precise mechanism to future work.

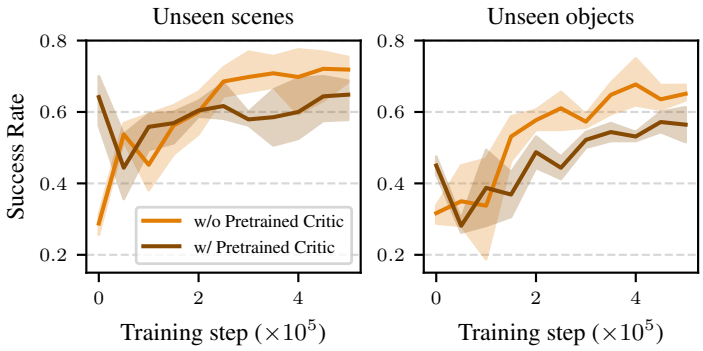

Figure 14: **Pretrained critic for accelerating value learning.** A critic pretrained on `MolmoSpaces` (64 tasks) starts higher than one trained from scratch, but the gap closes within one iteration and then reverses, on both unseen scenes (*left*) and unseen objects (*right*). Mean and 95% stratified bootstrap CIs across three seeds.

C.6 Stability in direct policy adaptation

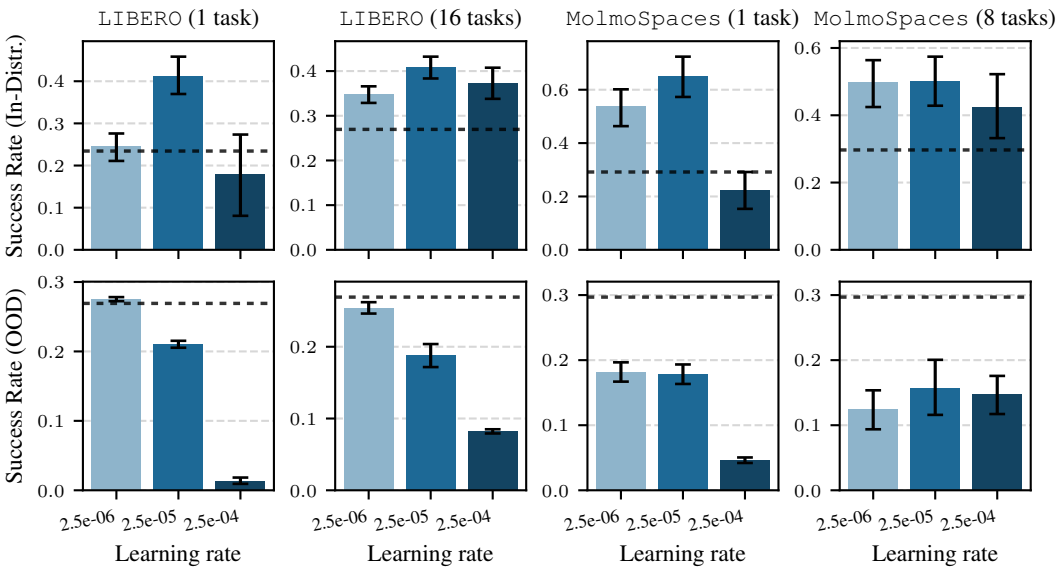

Figure 15: **Learning rate ablation for FSFT.** We sweep three learning rates across four task splits: `LIBERO` (1 task), `LIBERO` (16 tasks), `MolmoSpaces` (1 task), and `MolmoSpaces` (8 tasks). We evaluate performance on the training tasks (In-Distr.) and on the validation tasks (OOD). Bars show the mean over three seeds, with error bars denoting 95% stratified bootstrap confidence.

FSFT has few hyperparameters, of which the learning rate is among the most impactful. To assess its importance, we measure its influence on in-distribution and out-of-distribution (OOD) performance across four single- and multi-task splits: `Libero` (1 task), `Libero` (16 tasks), `MolmoSpaces` (1 task), and `MolmoSpaces` (8 tasks). We sweep three learning rates, $\{2.5 \cdot 10^{-6}, 2.5 \cdot 10^{-5}, 2.5 \cdot 10^{-4}\}$, over three seeds each, leaving all other hyperparameters unchanged. Figure 15 presents the results. The smallest value, $2.5 \cdot 10^{-6}$, degrades in-distribution performance, while the largest, $2.5 \cdot 10^{-4}$, is detrimental OOD. These results suggest that the learning rate mediates a trade-off between in-distribution and OOD performance. The middle value, $2.5 \cdot 10^{-5}$, balances the two, and we adopt it as the default in our FSFT experiments.

C.7 Number of action candidates N in Best-of-N

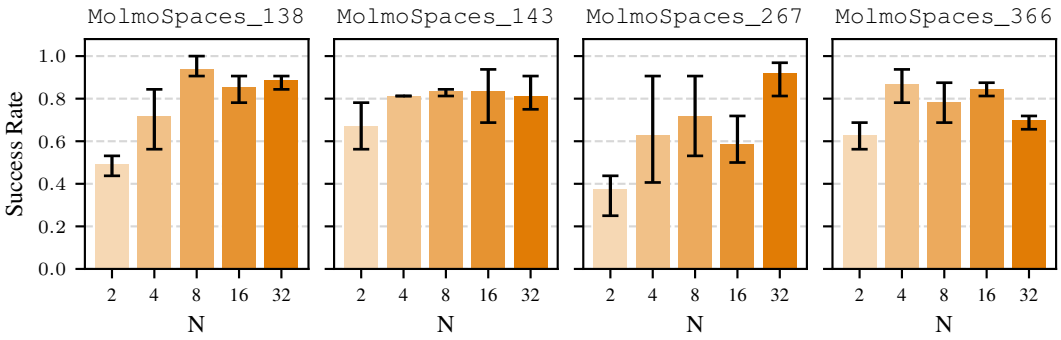

Figure 16: **Impact of the number of candidates N in Best-of-N on performance.** Success rate of BoN on each of the four single-task `MolmoSpaces` runs as N grows. On average, performance grows with N and peaks at $N = 32$, though the gains over $N \in \{8, 16\}$ are marginal. We report mean and 95% stratified bootstrap CIs over three seeds.

Beyond critic learning, Best-of-N introduces a single hyperparameter: the number of sampled actions N. At each step it draws N candidates $\{a_t^i\}_{i=1}^N \sim \pi_\theta(s_t, c)$, scores them, and executes $a_t^\star = \arg\max_i Q_\psi(s_t, a_t^i, c)$. Larger N gives the critic more candidates to choose from, at the cost of slower inference. To assess the impact of N on performance, we ablate $N \in \{2, 4, 8, 16, 32\}$ on four single-task `MolmoSpaces` runs. As shown in Figure 16, performance roughly grows with N on all tasks and peaks on average at $N = 32$, which we adopt as our default. The gains are diminishing, however: $N = 32$ improves only marginally over $N \in \{8, 16\}$.

N	Time (ms)	Rel.
1	146.3 ± 2.1	1.00×
2	159.5 ± 2.0	1.09×
4	183.9 ± 1.5	1.26×
8	232.9 ± 2.6	1.59×
16	325.2 ± 2.2	2.22×
32	510.7 ± 2.8	3.49×

Table 2: **BoN inference cost.** Wall-clock time of one Best-of-N step on a single GH200 GPU, relative to N=1.

Table 2 reports the wall-clock time of one BoN step as a function of N. Our default $N = 32$ costs 3.49× the time of $N = 1$, against 1.59× for $N = 8$, which offers a better balance between performance and inference cost. Since our experiments run in simulation, we keep $N = 32$ to prioritize performance; on a real robot, $N = 8$ would likely be preferable.

D EXPERIMENTAL DETAILS

D.1 ENVIRONMENTS AND TASK SPLITS

This section describes the environments we use and how we construct task splits. In all of our experiments, we utilize two benchmarks: `LIBERO-90` and `MolmoSpaces`. `LIBERO` is a widely used manipulation benchmark, whereas `MolmoSpaces` offers precise control over task diversity, which makes it better suited to studying generalization. We use its `pick` suite, which exposes 18 objects with 55–56 scenes each, allowing us to vary scene and object diversity independently.

Table 3: **`MolmoSpaces` task splits.** For each split we list the task IDs it contains, grouped by object and given without the `molmo_` prefix. *Scenes/obj.* is the number of scenes per object; *Init. SR* the mean success rate of the base policy over the tasks of the split. The two evaluation splits hold out one scene of each of the eight training objects, and one scene of each of eight objects never seen during training.

Split	Tasks	Scenes/obj.	Init. SR (%)
`MolmoSpaces` (1 task) *box*: 267, 366, 138, 143	4	4	29.2
`MolmoSpaces` (4 tasks) *box*: 366, 138 *spatula*: 373, 93	4	2	30.2
`MolmoSpaces` (16 tasks) *box*: 267, 366, 138, 143 *spatula*: 176, 373, 93, 394 *cup*: 48, 185, 24, 62 *tissuepaper*: 4, 260, 221, 348	16	4	29.2
`MolmoSpaces` (64 tasks) *boiler*: 332, 33, 242, 369, 126, 383, 295, 81 *box*: 164, 86, 70, 267, 366, 138, 143, 212 *cup*: 18, 292, 48, 185, 24, 62, 377, 78 *ladle*: 28, 311, 50, 56, 399, 187, 121, 27 *pot*: 309, 227, 265, 325, 277, 127, 254, 166 *tissuepaper*: 11, 361, 103, 4, 260, 221, 348, 305 *spatula*: 22, 210, 176, 373, 93, 394, 6, 47 *spoon*: 19, 296, 82, 39, 162, 148, 150, 7	64	8	29.5
Unseen scenes *boiler*: 8, *box*: 273, *cup*: 222, *ladle*: 73, *pot*: 346, *tissuepaper*: 49, *spatula*: 21, *spoon*: 379	8	1	29.7
Unseen objects *atomizer*: 140, *bowl*: 356, *fork*: 345, *knife*: 155, *remotecontrol*: 357, *salt/pepper shaker*: 163, *soapdispenser*: 208, *winebottle*: 226	8	1	29.7

Studying multi-task training and generalization requires training splits of increasing size but comparable difficulty, together with held-out tasks. For `MolmoSpaces`, we first subsample 400 tasks and estimate the initial success rate of each by rolling out 48 episodes with the base policy. From these we construct four training splits of 1, 4, 16 and 64 tasks, each a superset of the smaller ones, with a mean initial success rate near 30%. A split of k^2 tasks contains k objects with k scenes each. The single-task split is run on four different scenes of the same object, and we report the average, to reduce variance. For evaluation we hold out two further splits: eight unseen scenes of the objects

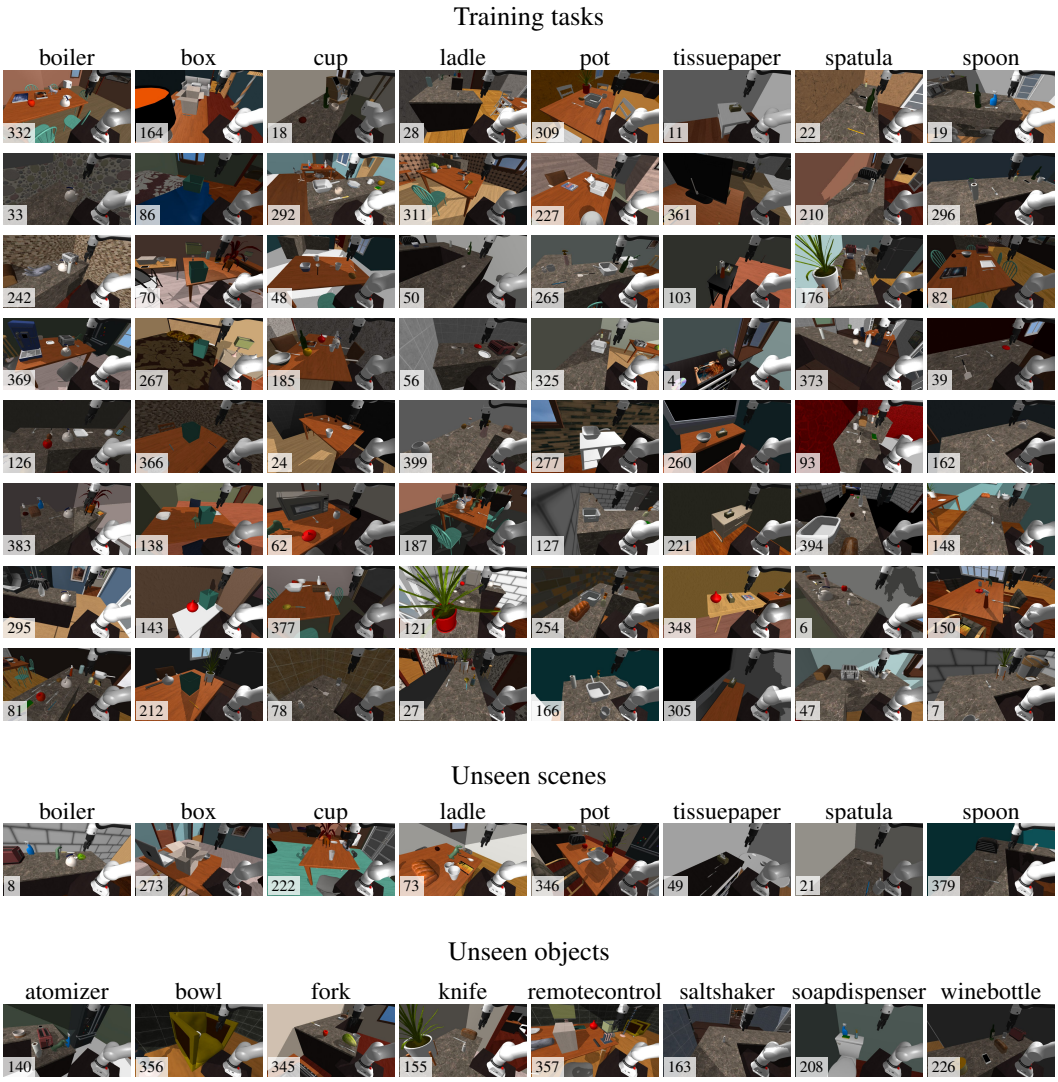

Figure 17: **`MolmoSpaces` task splits – visualization.** Initial exterior-camera frame of each task, ordered as in Table 3. In the training split, each column is one object, and the unseen scene of that object sits directly below its column. Task IDs are shown in the lower-left corner of each frame.

in the 64-task split, and eight objects that appear in no training split. Table 3 lists the resulting splits together with their objects and initial success rates.

We construct the `LIBERO` splits in the same way, estimating initial success rates from 48 rollouts on 89 of the 90 tasks, and again running the single-task split on four tasks. `LIBERO` is smaller than `MolmoSpaces`, so its splits cannot be balanced as tightly: mean initial success rate ranges from 23.4% to 27.0%. A further 25 tasks are held out to measure generalization. Table 4 lists the splits and their initial success rates.

Table 4: **LIBERO task splits.** All tasks are drawn from LIBERO-90; task IDs are given without the libero_90_ prefix. *Init. SR* is the mean success rate of the base policy over the tasks of a split. The training splits are nested, and the unseen tasks are disjoint from all of them.

Split	Tasks	Init. SR (%)
LIBERO (1 task) *Task IDs*: 79, 31, 82, 38	4	23.4
LIBERO (4 tasks) *Task IDs*: 79, 31, 82, 38	4	23.4
LIBERO (16 tasks) *Task IDs*: 36, 49, 87, 86, 21, 85, 53, 64, 31, 79, 38, 82, 30, 47, 56, 24	16	27.0
LIBERO (64 tasks) *Task IDs*: 5, 32, 6, 36, 74, 39, 40, 76, 41, 43, 45, 49, 89, 65, 62, 42, 78, 8, 1, 88, 16, 17, 3, 87, 86, 58, 21, 84, 83, 25, 80, 27, 85, 73, 35, 53, 63, 14, 44, 64, 31, 79, 38, 82, 28, 70, 59, 60, 30, 12, 29, 47, 54, 56, 33, 9, 46, 50, 67, 51, 24, 20, 55, 61	64	26.9
Unseen tasks *Task IDs*: 34, 37, 4, 71, 66, 52, 75, 11, 18, 23, 81, 10, 69, 48, 7, 13, 0, 26, 77, 19, 72, 22, 2, 68, 57	25	26.9

D.2 Implementation Details

Common Setup. All methods start from the same pretrained $\pi_{0.5}$ checkpoint, i.e., pi05_libero on LIBERO and pi05_droid_jointpos on MolmoSpaces, and share the same data collection loop. As in the released checkpoint, the policy sees only camera images and the language instruction; we verified that omitting proprioceptive state helps. Each iteration begins with a collection phase, in which the policy collects 20 episodes per task into a replay buffer; training then proceeds and differs by method. After the last iteration, a final evaluation phase begins. Collection and evaluation both use an exponential moving average of the policy parameters for stability. The policy outputs a chunk of 15 actions (10 actions on LIBERO) but executes only the first 5 before being queried again. We nonetheless store one sample per environment step, pairing each observation with the full action chunk executed from that step onward, stitched across consecutive queries rather than taken from a single predicted chunk. The buffer is FIFO and sampled uniformly.

FSFT. FSFT stores only successful episodes and fine-tunes all policy parameters, i.e., VLM backbone, action expert, and vision encoder, with the standard $\pi_{0.5}$ flow-matching loss.

BoN. BoN keeps the VLA's parameters frozen and instead learns a Q-function conditioned on observations and action chunks. At every query, it samples $N{=}32$ candidate chunks with independent initial noise and executes the one maximizing the minimum over the critic's ensemble, both during data collection and at evaluation. The critic never sees raw images as it takes the mean-pooled final-layer token embeddings of the policy's backbone, the proprioceptive state, and the flattened action chunk. Since the policy is frozen, this embedding is computed once at collection time and cached, which makes critic training substantially faster. We use a time-to-success reward (-1 per step, 0 on success), with episodes terminating only on success and time limits treated as truncation. We train Q and V on all collected episodes with H-step TD targets from slowly updated target networks: the H discounted rewards accumulated over the chunk, plus γ^H times the bootstrapped value of the state reached at its end. Because V regresses Q at buffer actions rather than at a maximum or an upper expectile, this is a SARSA-style evaluation of the data-collecting policy.

FSFT+BoN. FSFT+BoN collects data with BoN and additionally fine-tunes the policy on the successful transitions of the shared buffer. Critic training and action selection are unchanged. The policy is updated once every 100 critic steps, so it receives exactly as many gradient steps per round

as in FSFT. Fine-tuning the full policy makes the cached mean-pooled token embeddings stale, since those stored in earlier iterations come from earlier versions of the backbone. We therefore consider an expert-only variant that freezes the vision encoder and language model and trains only the action expert, keeping the embeddings stationary, and find that it works well in practice.

AWR. The critic training pipeline of AWR is entirely shared with FSFT+BoN. The sole difference is in the policy training objective. While both rely on a weighted supervised loss (see Equation 3), AWR weights are computed from the critic's output and are non-zero for unsuccessful trajectories. Advantages are computed as the difference between the minima of the two Q-functions and the two value functions, then scaled, clipped and exponentiated according to the tuned hyperparameters in Table 5. Finally, weights are scaled to sum to 1 in each batch.

PARL. PARL also shares its pipeline with FSFT+BoN, with two differences: the supervised loss has weight 1 for states in unsuccessful trajectories, and the target action is not necessarily the one executed by the policy during data collection. For computational reasons, we select target actions by optimizing against the critic through sampling alone. We store 32 candidate actions during inference, rerank them according to the critic for each state in the batch, and clone the best one. This approximates the optimization process of PARL while remaining computationally manageable.

PARL+filter. PARL+filter simply sets the weights of the policy loss in PARL to 0 if the state belongs to an unsuccessful trajectory, thus moving closer to FSFT+BoN.

OGPO. OGPO shares its critic learning pipeline with FSFT + BoN. As in FSFT + BoN, we keep the vision encoder and the language model frozen and train only the action expert. As discussed in Appendix B, we use group relative conservative advantages and BC loss over trajectories in the success buffer as an additional regularization. We use the hyperparameters specified in Table 5 for our runs.

D.3 HYPERPARAMETERS

Table 5: **Hyperparameters.** *Training budget*: each run performs M collection rounds of K episodes per task, with the given number of gradient steps per round (single-task / multi-task). *Tuned weights*: whether the full VLA or only the action expert is fine-tuned. *Trained on*: whether the SFT loss uses only successful episodes. *Update interval*: critic steps per policy update. *Buffer re-ranking*: whether the current critic re-selects the best of the N stored candidates each time a batch is sampled, rather than fixing the choice at collection time. The replay buffer holds 250k transitions in all runs except for the 64-task split, where larger 500k buffers are used for FSFT and BoN.

	FSFT	BoN	FSFT +BoN	AWR	OGPO	PARL	PARL +filter
Training budget							
Collection rounds M	10	10	10	10	10	10	10
Episodes per task K	20	20	20	20	20	20	20
Steps per round	100/500	10k/50k	10k/50k	10k/50k	10k/50k	10k/50k	10k/50k
Policy							
Learning rate	$2.5{\cdot}10^{-5}$	–	$2.5{\cdot}10^{-5}$	$2.5{\cdot}10^{-5}$	$2.5{\cdot}10^{-5}$	$2.5{\cdot}10^{-5}$	$2.5{\cdot}10^{-5}$
Batch size	256	–	256	256	256	256	256
EMA decay	0.999	–	0.999	0.999	0.999	0.999	0.999
Tuned weights	full	–	expert	full	expert	full	full
Trained on	successes	–	successes	all	all	all	successes
Update interval	–	–	100	100	100	100	100
Critic							
Learning rate	–	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Architecture	–	BroNet	BroNet	BroNet	BroNet	BroNet	BroNet
Hidden width	–	1024	1024	1024	1024	1024	1024
Batch size	–	1024	1024	1024	1024	1024	1024
EMA decay	–	0.995	0.995	0.995	0.995	0.995	0.995
Bootstrap	–	TD	TD	TD	TD	TD	TD
Discount γ	–	0.995	0.995	0.995	0.995	0.995	0.995
Replay buffer							
Capacity	250/500k	250/500k	250k	250k	250k	250k	250k
Action selection							
Candidates N	–	32	32	–	32	32	32
Buffer re-ranking	–	–	no	–	no	yes	yes
Method-specific							
AWR temperature β	–	–	–	0.1	–	–	–
AWR weight clip	–	–	–	20	–	–	–
OGPO PPO clip ratio	–	–	–	–	0.1	–	–
OGPO denoising steps	–	–	–	–	10	–	–
OGPO Group size	–	–	–	–	8	–	–
OGPO BC loss weight	–	–	–	–	1.0	–	–

D.4 EVALUATION PROTOCOL

In this section we describe the evaluation protocol. On both `LIBERO` and `MolmoSpaces` we evaluate each policy in distribution and on unseen tasks. For `LIBERO`, the unseen set comprises 25 tasks; for `MolmoSpaces`, eight unseen scenes of objects from the 64-task split and eight unseen objects (see Appendix D.1). Our metric throughout is success rate.

During training, the in-distribution success rate is measured on the 20 episodes the policy collects per task in each round, so no extra mid-evaluation rollouts are needed. After the last training phase, we run a separate evaluation phase of 32 episodes per task, both in distribution and on unseen tasks. We repeat every experiment with three seeds and report the mean success rate over seeds, together with 95% stratified bootstrap confidence intervals computed with `rliable` (Agarwal et al., 2021).

D.5 Computational Resources

Our experiments ran on nodes with four NVIDIA GH200 GPUs each. A single-task run took about 2-5 hours of wall-clock time depending on the task, and runs on the largest multi-task splits took 100-200 hours.

E TRAINING CURVES

We report success curves over training for the experiments in Figures 4, 10 and 8.

E.1 MAIN RESULTS — MOLMOSPACES

E.1.1 MOLMOSPACES (1 TASK) IN-DISTRIBUTION

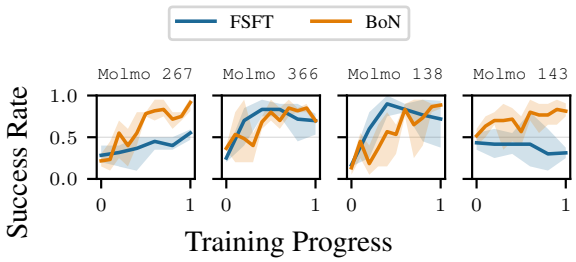

Figure 18: **MolmoSpaces (1 task): training tasks.** Success rate on the training tasks as a function of normalized training progress, where 1 denotes the end of post-training. Lines show the mean over three seeds. Shaded regions denote 95% stratified bootstrap CIs.

E.1.2 MOLMOSPACES (1 TASK) UNSEEN SCENES

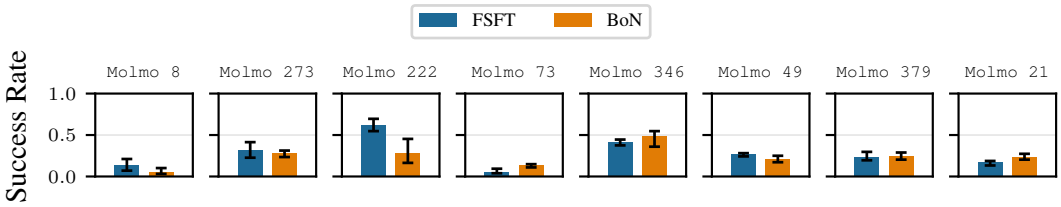

Figure 19: **MolmoSpaces (1 task): unseen scenes.** Final success rate on unseen scenes. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.1.3 MOLMOSPACES (1 TASK) UNSEEN OBJECTS

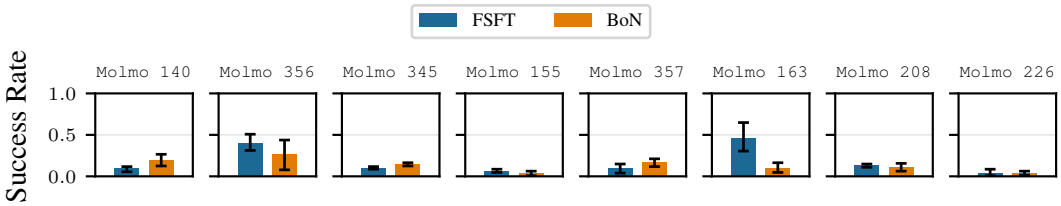

Figure 20: **MolmoSpaces (1 task): unseen objects.** Final success rate on unseen objects. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.1.4 MolmoSpaces (4 Tasks) In-Distribution



Figure 21: **MolmoSpaces (4 tasks): training tasks.** Success rate on the training tasks as a function of normalized training progress, where 1 denotes the end of post-training. Lines show the mean over three seeds. Shaded regions denote 95% stratified bootstrap CIs.

E.1.5 MolmoSpaces (4 Tasks) Unseen Scenes

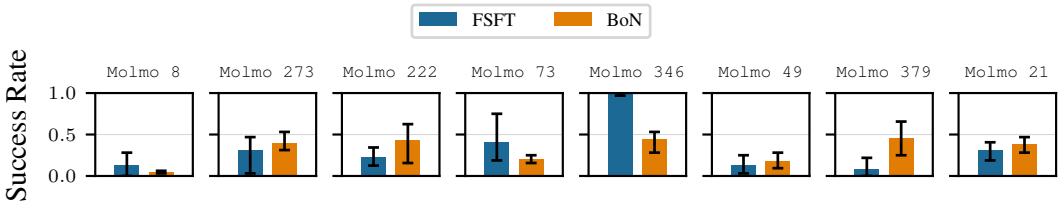

Figure 22: **MolmoSpaces (4 tasks): unseen scenes.** Final success rate on unseen scenes. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.1.6 MolmoSpaces (4 Tasks) Unseen Objects

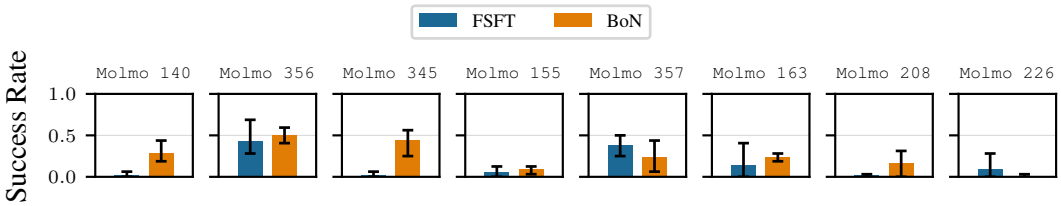

Figure 23: **MolmoSpaces (4 tasks): unseen objects.** Final success rate on unseen objects. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.1.7 MolmoSpaces (16 Tasks) In-Distribution

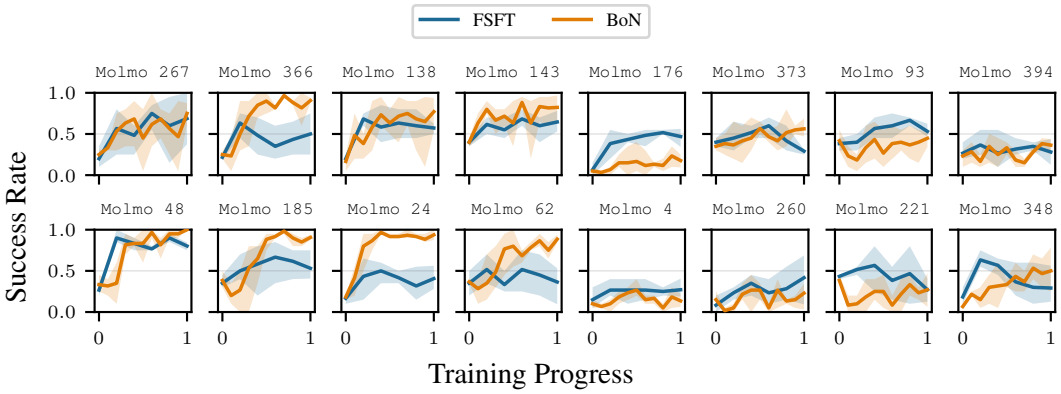

Figure 24: **`MolmoSpaces` (16 tasks): training tasks.** Success rate on the training tasks as a function of normalized training progress, where 1 denotes the end of post-training. Lines show the mean over three seeds. Shaded regions denote 95% stratified bootstrap CIs.

E.1.8 MolmoSpaces (16 Tasks) Unseen Scenes

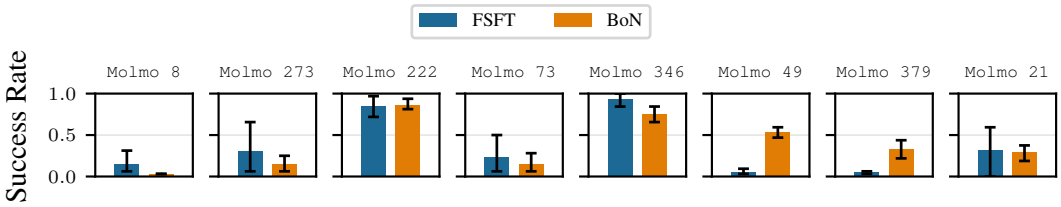

Figure 25: **`MolmoSpaces` (16 tasks): unseen scenes.** Final success rate on unseen scenes. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.1.9 MolmoSpaces (16 Tasks) Unseen Objects

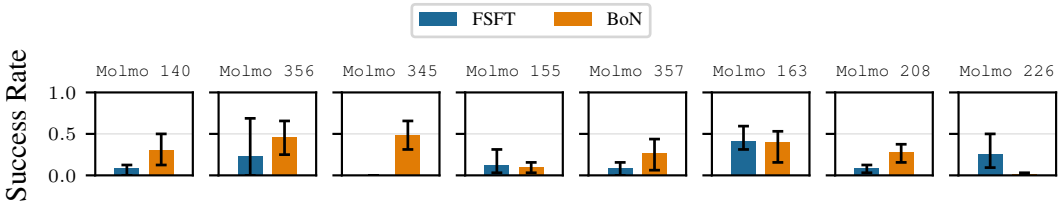

Figure 26: **`MolmoSpaces` (16 tasks): unseen objects.** Final success rate on unseen objects. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.1.10 MolmoSpaces (64 Tasks) In-Distribution

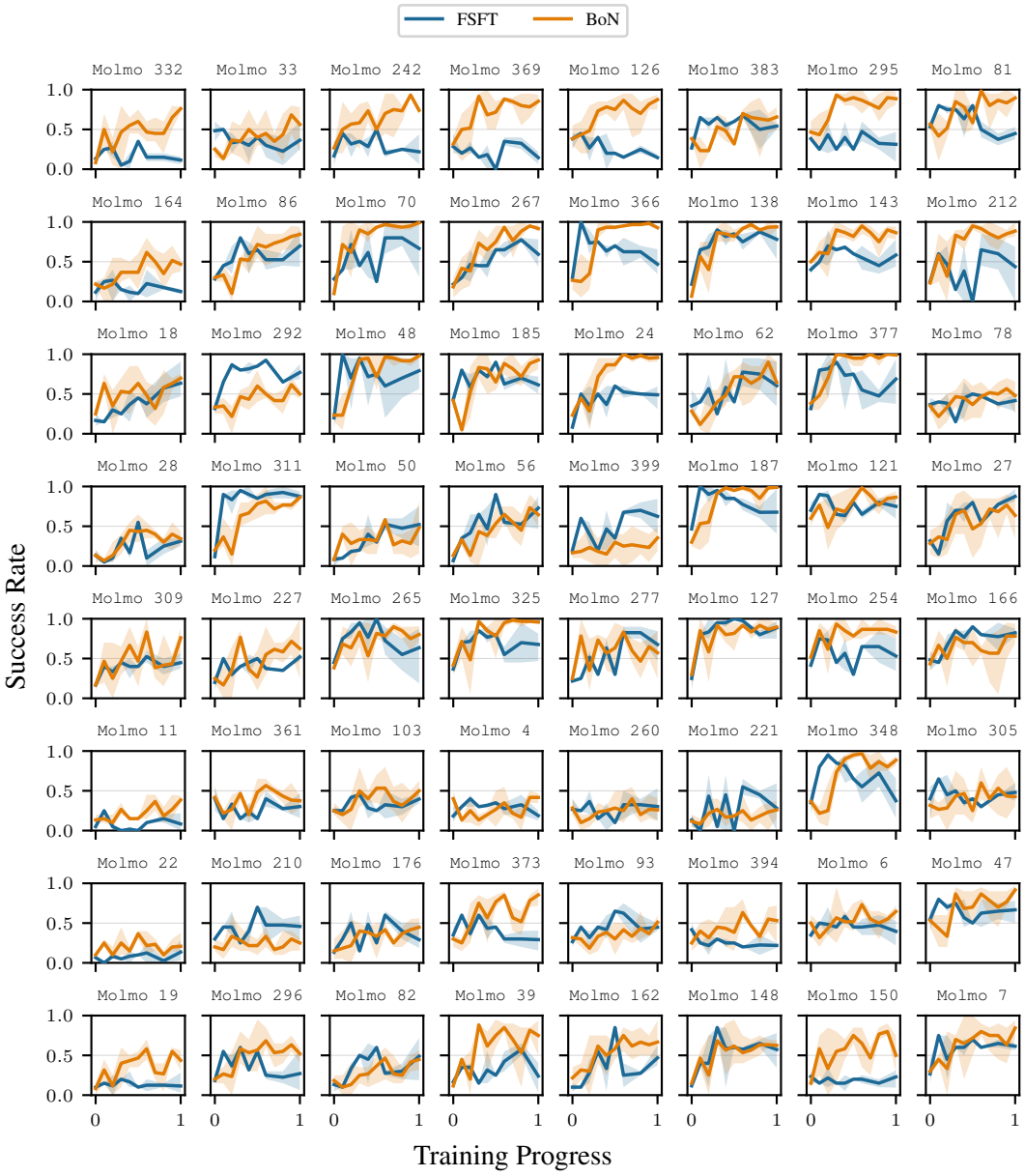

Figure 27: **`MolmoSpaces` (64 tasks): training tasks.** Success rate on the training tasks as a function of normalized training progress, where 1 denotes the end of post-training. Lines show the mean over three seeds. Shaded regions denote 95% stratified bootstrap CIs.

E.1.11 MolmoSpaces (64 Tasks) Unseen Scenes

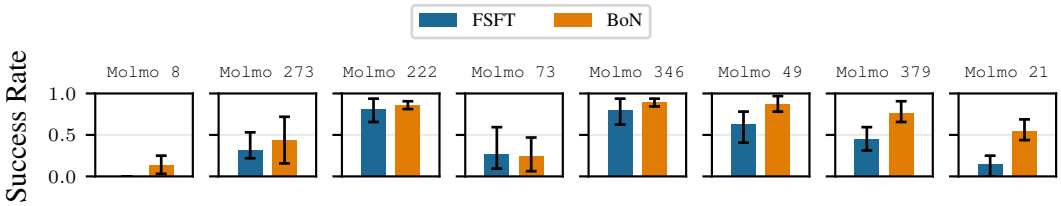

Figure 28: **MolmoSpaces (64 tasks): unseen scenes.** Final success rate on unseen scenes. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.1.12 MolmoSpaces (64 Tasks) Unseen Objects

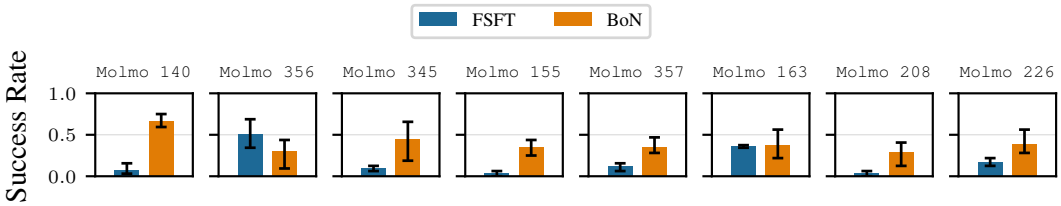

Figure 29: **MolmoSpaces (64 tasks): unseen objects.** Final success rate on unseen objects. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.2 Main Results — LIBERO

E.2.1 LIBERO (1 Task) In-Distribution

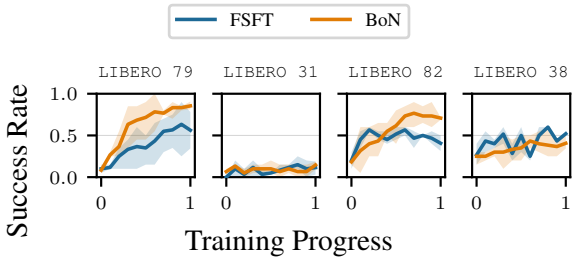

Figure 30: **LIBERO (1 task): training tasks.** Success rate on the training tasks as a function of normalized training progress, where 1 denotes the end of post-training. Lines show the mean over three seeds. Shaded regions denote 95% stratified bootstrap CIs.

E.2.2 LIBERO (1 Task) Unseen Tasks

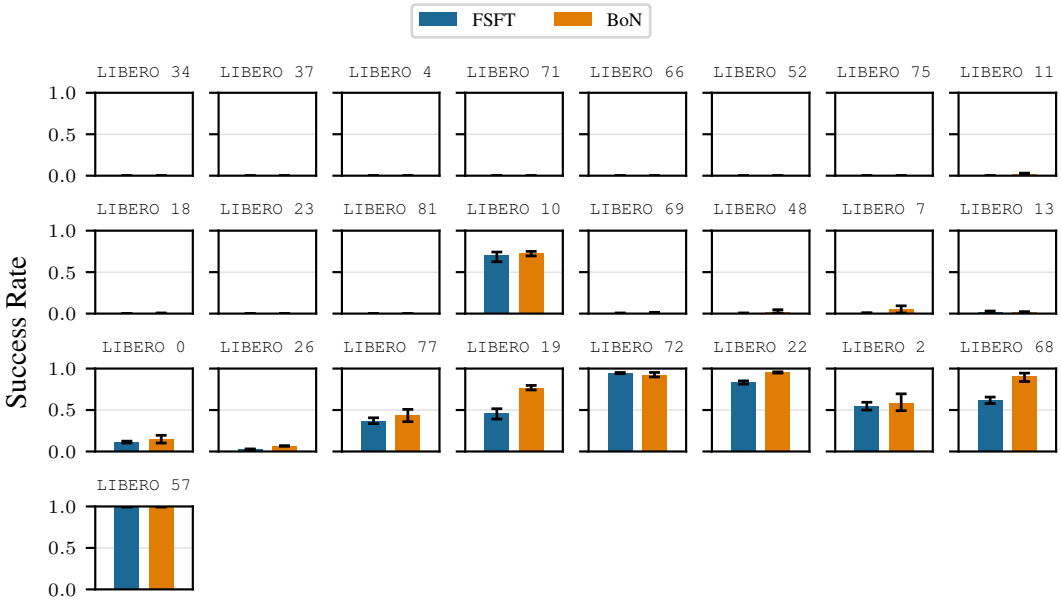

Figure 31: **LIBERO (1 task): unseen tasks.** Final success rate on unseen tasks. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.2.3 LIBERO (4 TASKS) IN-DISTRIBUTION

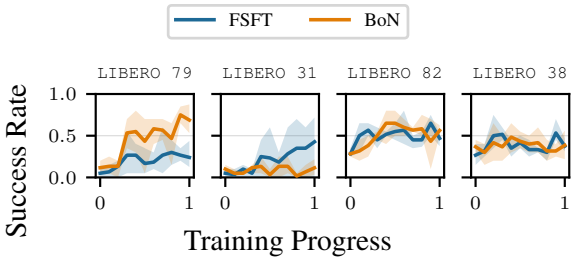

Figure 32: **LIBERO (4 tasks): training tasks.** Success rate on the training tasks as a function of normalized training progress, where 1 denotes the end of post-training. Lines show the mean over three seeds. Shaded regions denote 95% stratified bootstrap CIs.

E.2.4 LIBERO (4 TASKS) UNSEEN TASKS

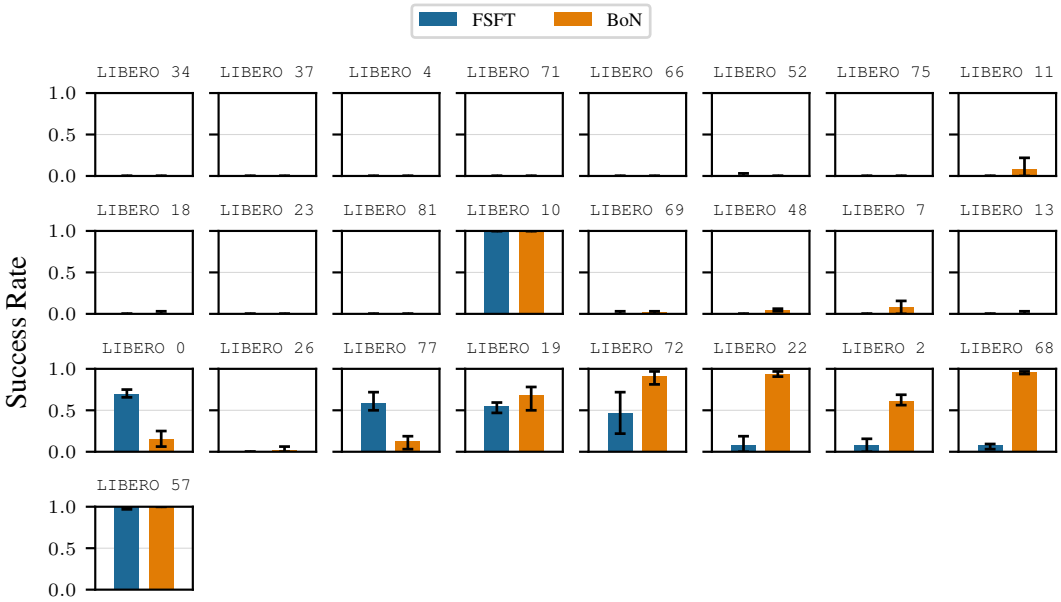

Figure 33: **LIBERO (4 tasks): unseen tasks.** Final success rate on unseen tasks. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.2.5 LIBERO (16 TASKS) IN-DISTRIBUTION

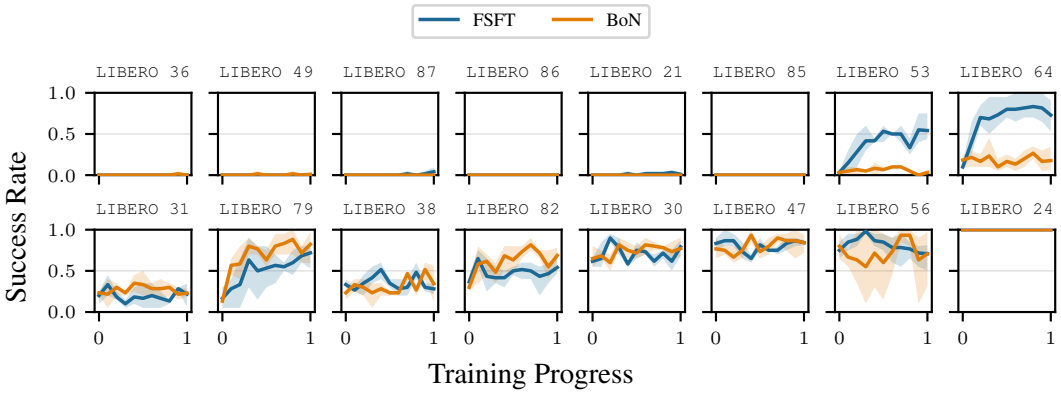

Figure 34: **LIBERO (16 tasks): training tasks.** Success rate on the training tasks as a function of normalized training progress, where 1 denotes the end of post-training. Lines show the mean over three seeds. Shaded regions denote 95% stratified bootstrap CIs.

E.2.6 LIBERO (16 TASKS) UNSEEN TASKS

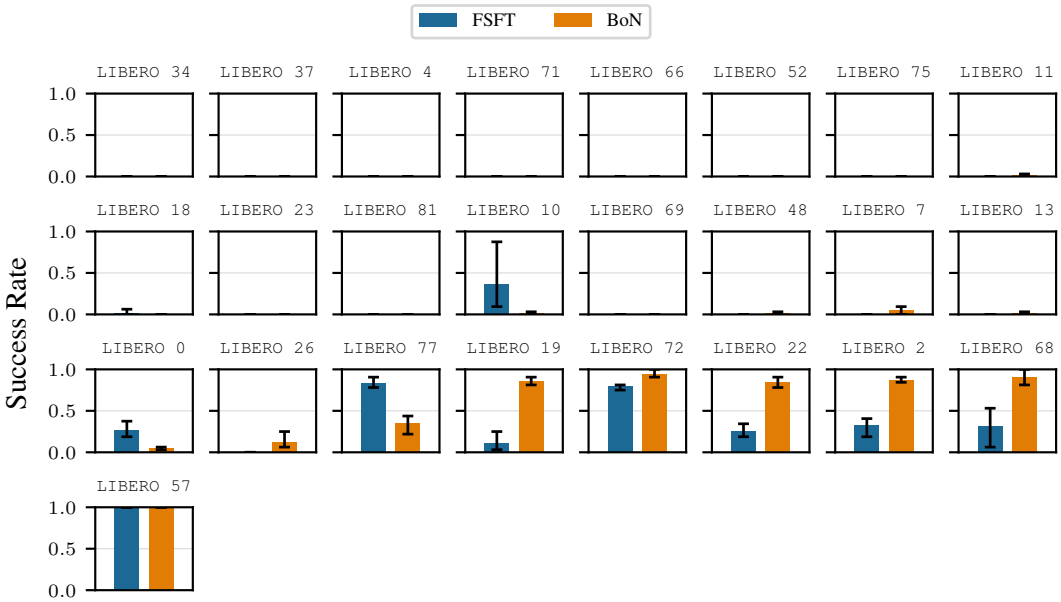

Figure 35: **LIBERO (16 tasks): unseen tasks.** Final success rate on unseen tasks. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.2.7 LIBERO (64 TASKS) IN-DISTRIBUTION

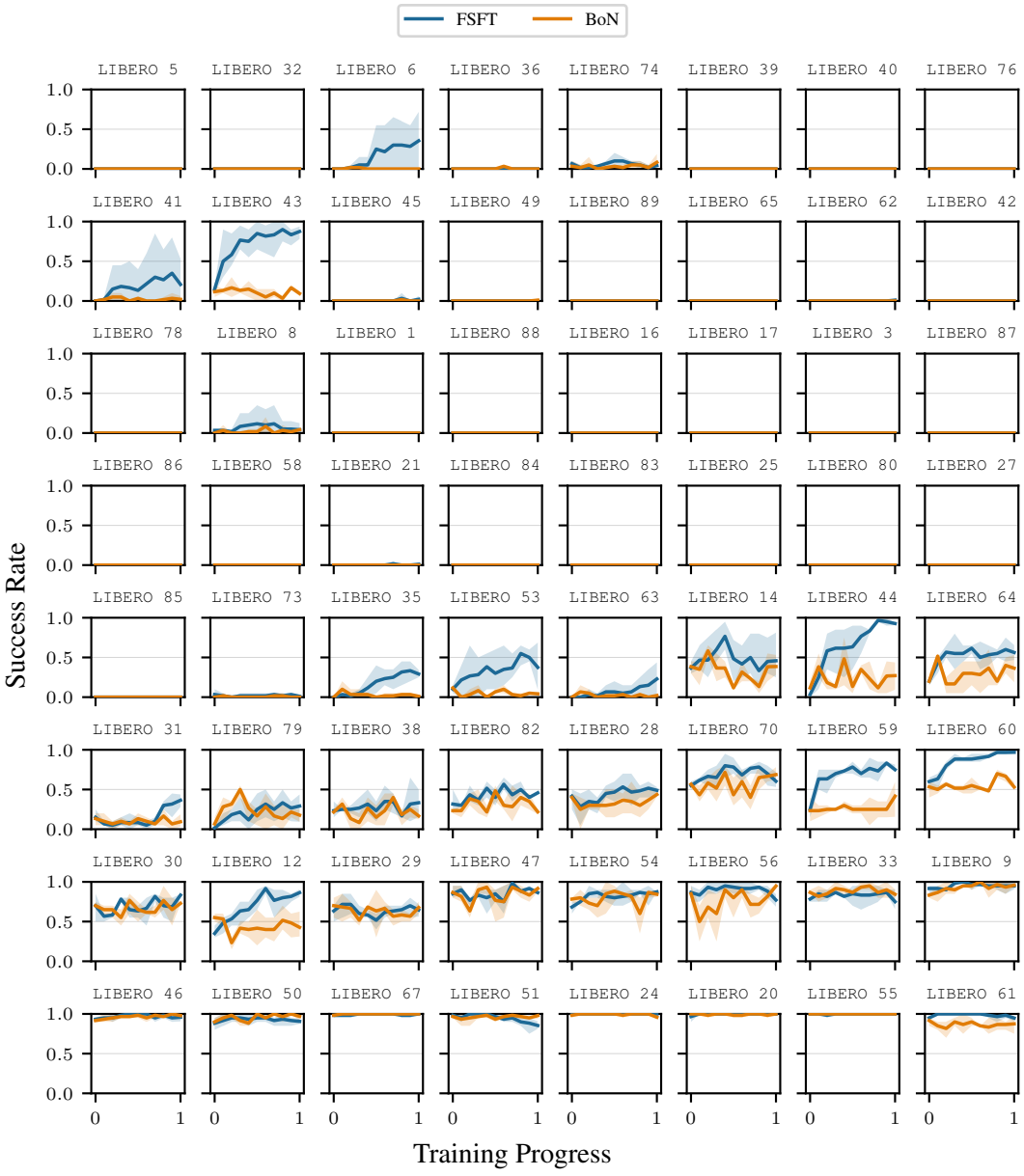

Figure 36: **LIBERO (64 tasks): training tasks.** Success rate on the training tasks as a function of normalized training progress, where 1 denotes the end of post-training. Lines show the mean over three seeds. Shaded regions denote 95% stratified bootstrap CIs.

E.2.8 LIBERO (64 TASKS) UNSEEN TASKS

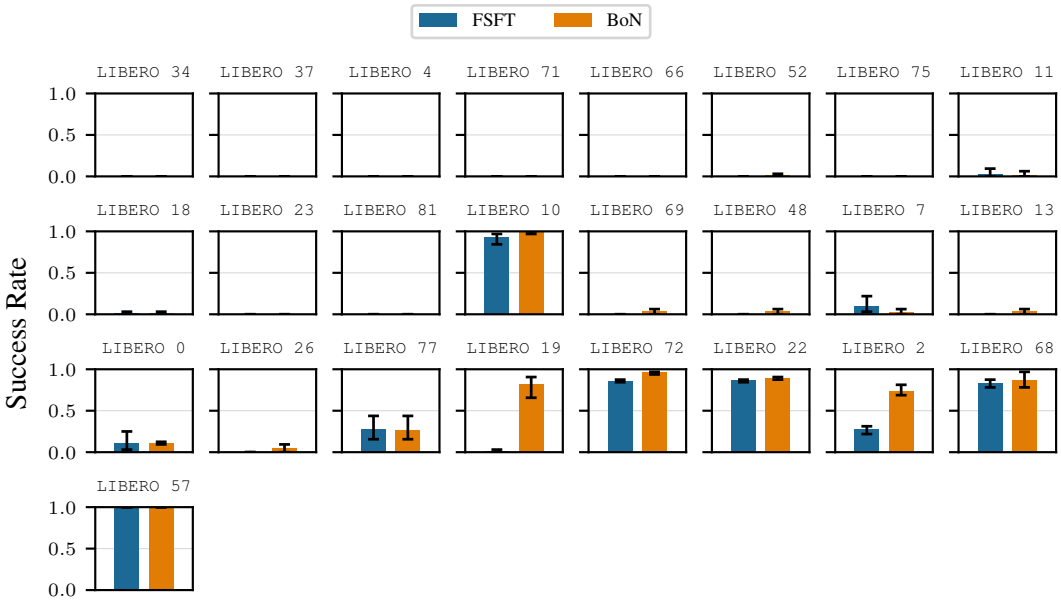

Figure 37: **LIBERO (64 tasks): unseen tasks.** Final success rate on unseen tasks. Bars show the mean over three seeds with 95% stratified bootstrap CIs.

E.3 Combining Policy Adaptation and Critic Guidance Results

E.3.1 LIBERO (1 Task)

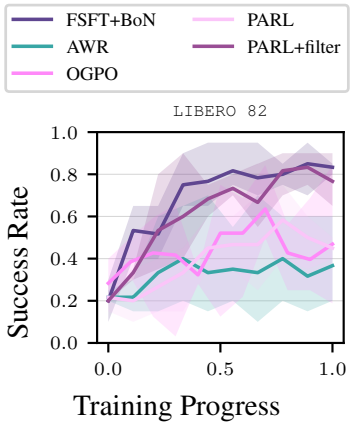

Figure 38: **LIBERO (1 task): combining policy adaptation and critic guidance.** Success rate on the training tasks as a function of normalized training progress, where 1 denotes the end of post-training. Lines show the mean over three seeds. Shaded regions denote 95% stratified bootstrap CIs.

E.4 LIBERO (4 Tasks)

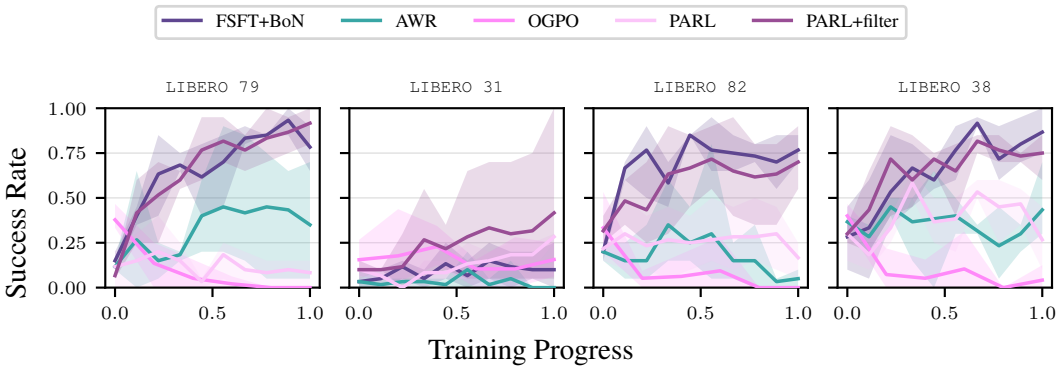

Figure 39: **LIBERO (4 tasks): combining policy adaptation and critic guidance.** Success rate on the training tasks as a function of normalized training progress, where 1 denotes the end of post-training. Lines show the mean over three seeds. Shaded regions denote 95% stratified bootstrap CIs.