CRAFT: CONSTRAINED ROLLOUT ALLOCATION FROM FEEDBACK FOR VISION LANGUAGE ACTION REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Online reinforcement fine-tuning of Vision-Language-Action (VLA) policies requires repeated environment interaction and policy optimization, motivating effective use of a limited training budget. We investigate multi-task fine-tuning under a fixed budget of policy update iterations and a fixed number of rollout trajectories per update. Uniform task sampling ignores changing learning needs. Saturated tasks may offer little additional signal, difficult tasks may yield only failures, and previously acquired skills may deteriorate. We propose *CRAFT* (Constrained Rollout Allocation from Feedback for Vision-Language-Action Reinforcement Learning), a feedback-guided framework for adaptive task allocation. CRAFT tracks gaps to target success rates, variation in group outcomes, and differences between recent and longer-term success estimates to identify unmet learning needs, reward contrast, and performance changes. Online normalization and priority smoothing stabilize task priorities, while coverage bonuses and a probability floor preserve opportunities to revisit less-observed tasks. Under matched 30-update budgets, CRAFT improves LIBERO-Long success and episode-weighted LIBERO-PRO success by 4.0 and 7.6 percentage points, respectively. Across the four LIBERO suites, the average gain is 0.9 percentage points. Further analysis reveals how the benefits of feedback-guided allocation vary across task suites and stages of budget-constrained adaptation.

1 INTRODUCTION

Pretrained Vision-Language-Action (VLA) policies provide a strong starting point for robot learning, but fine-tuning them on target tasks still requires experience. Advances in action generation and large-scale pretraining have expanded generalist policies (Chi et al., 2023; Kim et al., 2025b; Black et al., 2025), and online reinforcement learning (RL) can improve them further through environment feedback (Chen et al., 2025). Such fine-tuning repeatedly incurs both rollout collection and policy-optimization costs. We therefore study a fixed training budget, specified by the number of policy-update rounds and rollout opportunities per round. Uniform task sampling spreads these opportunities evenly in expectation, although tasks differ in their learning needs. Saturated tasks may offer little additional signal, persistently failing tasks may provide no within-group reward contrast, and previously acquired skills may deteriorate. Under a fixed rollout budget, allocating more trajectories to one task leaves fewer for others, making task allocation a key factor in effective online fine-tuning.

Existing work addresses different parts of this problem. VLA reinforcement learning improves how collected trajectories update the policy (Chen et al., 2025), while adaptive task sampling uses learning potential, outcome variation, and staleness to guide experience selection (Jiang et al., 2021; Rutherford et al., 2024). However, a stronger learner alone does not decide what to collect next, and no single statistic is unambiguous. Low success identifies an unmet target but cannot distinguish informative mixed outcomes from persistent failure. Outcome variation identifies reward contrast but is zero for both all-success and all-failure groups. Neither statistic alone distinguishes recent changes from longer-term performance. Moreover, feedback comes from small, selectively collected groups, so priorities can fluctuate and neglected tasks can become poorly observed. *Therefore, the challenge is to distinguish unmet learning needs, usable reward contrast, and performance change*

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.

from sparse group feedback, while avoiding unstable allocation and prolonged task neglect within a fixed update budget.

To tackle this problem, we introduce *CRAFT* (Constrained Rollout Allocation from Feedback for Vision-Language-Action Reinforcement Learning), a framework that leverages feedback from completed groups to distribute upcoming rollout groups across tasks while keeping the policy-update budget fixed. Deviations from target success rates reflect remaining learning needs; variability in group results captures reward contrast; and discrepancies between short-horizon and longer-term success estimates indicate performance shifts. We apply online normalization and priority smoothing to temper volatility in the aggregated scores, and use coverage bonuses plus a minimum sampling probability to maintain chances of revisiting under-observed tasks. While these signals steer data collection, they do not ensure that any given group will be informative for learning. A shared task and starting state define every chosen group. We implement CRAFT within the π_{RL} Flow-SDE pipeline using group-relative advantages (Chen et al., 2025; Yu et al., 2026), while matching the uniform baseline in initialization, learner setup, and update budget. This paper makes the following three major contributions:

- **Task Allocation from Group Feedback.** We formulate rollout allocation under limited policy updates using competence deficits, group outcome variation, and temporal change to distinguish unmet targets, reward contrast, and changing performance when deciding what to collect next.
- **Coverage Safeguards for Adaptive Sampling.** We normalize and smooth task priorities, then use coverage bonuses and a minimum sampling probability to keep less-observed tasks eligible for future collection.
- **Gains under Matched Update Budgets.** Compared with uniform allocation at 30 updates, CRAFT raises the average success rate across four LIBERO suites from 96.1% to 97.0% and episode-weighted success across four LIBERO-PRO perturbation families of LIBERO-Long from 44.4% to 52.0%. Checkpoint and task-level analyses characterize variation across suites and training stages.

2 Related Work

Vision-Language-Action Policies. Pretrained VLA policies transfer visual and linguistic knowledge to robot control. OpenVLA combines pretrained visual representations with robot demonstrations (Kim et al., 2025b), while $\pi_{0.5}$ uses heterogeneous co-training to improve generalization to unfamiliar environments (Black et al., 2025). Beyond pretraining, OpenVLA-OFT studies how decoding, action representations, and fine-tuning objectives affect adaptation (Kim et al., 2025a). These advances improve representations and fine-tuning recipes, but do not by themselves determine which tasks should receive online experience. CRAFT complements them by allocating rollout groups from task feedback under a limited policy-update budget.

Reinforcement Learning for VLA Adaptation. Online RL turns environment feedback into policy improvement. For flow-based VLA policies, π_{RL} introduces Flow-Noise and Flow-SDE to enable policy optimization despite otherwise intractable action likelihoods (Chen et al., 2025). With binary rewards, same-outcome groups provide no centered reward contrast. In language-model RL, DAPO oversamples and filters all-correct and all-incorrect groups (Yu et al., 2025); for VLA adaptation, Prism-GRPO augments binary rewards with execution-quality scores to distinguish same-outcome trajectories (Deng et al., 2026). These strategies improve the signal from collected experience. CRAFT addresses the complementary decision of which tasks to collect next under limited policy updates, without changing rewards or discarding groups.

Adaptive Sampling under Limited Training Budgets. Adaptive sampling directly addresses the experience to collect. Prioritized Level Replay combines estimated learning potential with staleness-aware sampling (Jiang et al., 2021), while Sampling for Learnability uses $p(1-p)$ to prioritize levels with intermediate estimated success rates (Rutherford et al., 2024). VADE extends online selection to multimodal reasoning through Bayesian difficulty estimates, Thompson sampling, and decaying priors that track policy changes (Hu et al., 2026). These approaches establish the value of learning opportunity, changing performance, and continued coverage. For grouped VLA rollouts, however, outcome variation alone cannot distinguish saturated tasks from

persistently unsuccessful ones, and current success alone cannot capture recent change. *To the best of our knowledge, CRAFT is the first framework to use group feedback to allocate rollouts across tasks in VLA reinforcement learning under a limited training budget.*

3 Problem Formulation

We consider online RL fine-tuning over K accessible tasks, starting from a fixed policy checkpoint. Training is restricted to a budget of $U_{\max}$ outer policy-update rounds. At each collection request, the sampler allocates M groups, each containing G trajectories. Each group selects one task and one initial state; stochastic policy execution can then produce different outcomes under the same conditions. We write $p_i^{(q)}$ for the probability of task i at allocation request q. This request counter is distinct from the policy-update index, since an environment reset need not correspond to an optimizer update. Within a completed group on task i, $g \in \{1, \dots, G\}$ indexes its G trajectories, and $r_{ig} \in \{0, 1\}$ indicates whether trajectory g succeeds during the episode. The group mean is $\bar{r}_i = G^{-1} \sum_{g=1}^{G} r_{ig}$. Here, we omit the group index for readability: $\bar{r}_i$ refers to the current group means, not a task-wide average. The centered outcome $r_{ig} - \bar{r}_i$ vanishes when all group outcomes agree. This motivates tracking outcome variation; it does not fully capture policy advantage, which also depends on normalization and masking. Our goal is to improve independently evaluated task success within the available update budget by adapting future task probabilities. We use *rollout allocation* to mean task selection for subsequent collection, not action selection, reward routing, or redistribution of already collected trajectories. Training-group statistics inform the sampler, while independent evaluation under a fixed protocol determines reported performance. A fixed update budget constrains adaptation opportunities but does not fix actual simulator transitions or wall-clock time, both of which can vary by task.

4 Methodology

4.1 Overview of CRAFT

Figure 1 presents CRAFT as a collection-and-learning loop. The left panel contrasts uniform allocation with tasks whose learning needs differ under limited policy updates; the difficulty labels are illustrative, not fixed inputs to CRAFT. The center panel uses completed-group feedback to track target-success deficits D_i, outcome variation L_i, and recent success changes C_i. Online normalization and priority smoothing combine these signals, while coverage bonuses and a probability floor safeguard future collection. The right panel collects G trajectories for each selected task, starting from a shared initial state, using the Flow-SDE policy. Group rewards yield relative advantages for the policy update, and completed-group outcomes feed back to CRAFT for subsequent task allocation.

The controller retains task statistics, priorities, and coverage counters across collection requests, so allocation responds to the evolving policy rather than isolated groups. The figure's $p_i(t+1)$ denotes the next-task distribution; below, we use the allocation-request index q to distinguish task selection from policy-update rounds. CRAFT operates only during training; evaluation follows a fixed task protocol. Algorithm 1 summarizes the allocation cycle. Reset boundaries, distributed locking, and state restoration are detailed in Appendix A.

4.2 Group-Feedback Signals for Task Allocation

For the current group on task i, we abbreviate the outcomes and mean defined in Section 3 as $r_g = r_{ig}$ and $\bar{r} = \bar{r}_i$. Its binary-outcome variation is $v = 4\bar{r}(1 - \bar{r})$, which is zero for identical outcomes and maximal for an equal split of successes and failures. It measures observed reward contrast, not epistemic uncertainty or gradient quality.

For each task, s_i^f and s_i^s estimate recent and longer-term success, respectively, using fast and slow exponential moving averages (EMAs). The statistic L_i is an EMA of group variation v. Let n_i count

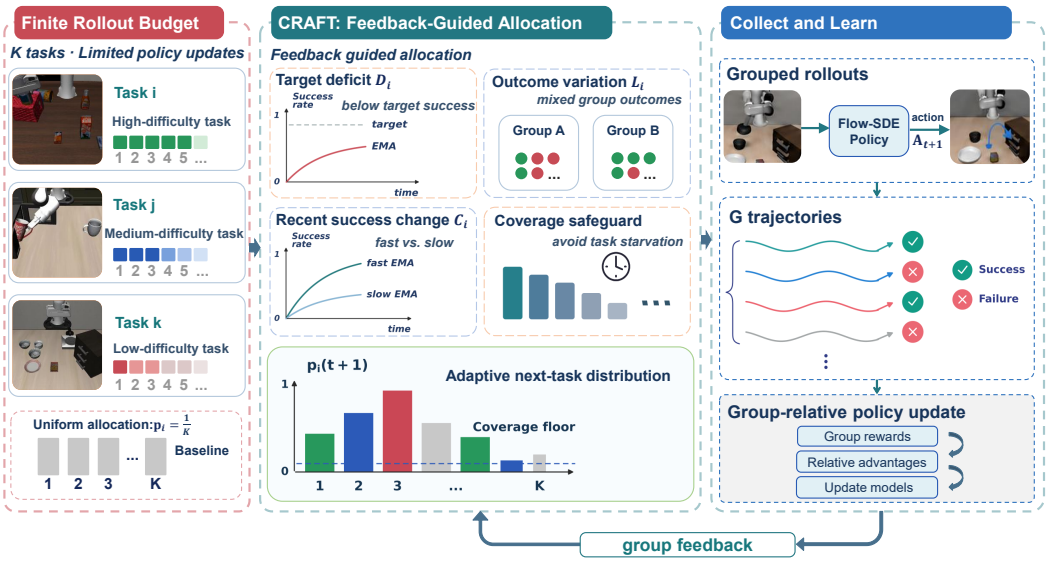

Figure 1: Overview of CRAFT for feedback-guided rollout allocation in VLA RL.

previously processed groups on task i. We define the update operator

$$\mathcal{E}_\beta(a, y; n) = \begin{cases} y, & n = 0, \\ \beta a + (1-\beta)y, & n > 0, \end{cases} \tag{1}$$

where a is the stored estimate, y is a scalar statistic from the newly completed group, and β weights the previous estimate. Superscripts "old" and "new" denote values before and after processing that group. Before incrementing n_i, we compute $s_i^{f,\text{new}} = \mathcal{E}_{\beta_f}(s_i^{f,\text{old}}, \bar{r}; n_i)$, $s_i^{s,\text{new}} = \mathcal{E}_{\beta_s}(s_i^{s,\text{old}}, \bar{r}; n_i)$, and $L_i^{\text{new}} = \mathcal{E}_{\beta_L}(L_i^{\text{old}}, v; n_i)$. Thus, y is either the group success rate $\bar{r}$ or its variation v, not an environment observation. We use $\beta_f = 0.5$ and $\beta_s = \beta_L = 0.9$, so the fast estimate responds more strongly to new feedback. The first group initializes all three estimates directly; only the observed task changes. Omitting the "new" superscripts from these updated estimates, we compute

$$D_i = \frac{\max(\tau - s_i^s, 0)}{\tau - s^{\text{random}}}, \qquad C_i = |s_i^f - s_i^s|, \tag{2}$$

where τ is a target success rate and $s^{\text{random}} < \tau$ is a reference rate. We use $\tau = 0.9$ and $s^{\text{random}} = 0$. Together with the variation estimate L_i above, D_i and C_i form the three allocation signals.

The signals answer different questions. The *success deficit* D_i identifies competence that remains below target. The *outcome variation* L_i captures recent contrast within groups. The *temporal change* C_i highlights a discrepancy between recent and longer-term performance. A persistently unsuccessful task can have a large deficit but little variation, whereas a task with mixed outcomes can provide contrast before reaching the target. Because C_i is an absolute difference, it captures discrepancies in either direction without distinguishing improvement from deterioration. Together, these signals define a task-priority heuristic, not additional environmental rewards.

4.3 Stable Task Priorities and Coverage Safeguards

Raw signals have different scales. For each signal $x \in \{D, L, C\}$, CRAFT maintains global Welford statistics $\mathcal{M}_x = (c_x, \mu_x, V_x)$, where c_x is the observation count and V_x is the sum of squared deviations. The sample variance is $\sigma_x^2 = V_x / \max(c_x - 1, 1)$. After incorporating the current observation, CRAFT standardizes the signal as

$$\widehat{x}_i = \text{Norm}(x_i; \mathcal{M}_x) = \begin{cases} \dfrac{x_i - \mu_x}{\sqrt{\sigma_x^2 + \epsilon}}, & c_x \geq 2 \text{ and } \sigma_x^2 > \epsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

We use $\epsilon = 10^{-8}$. The moments are shared across tasks, rather than estimated separately for each task, and are updated in the order in which groups are processed.

The normalized signals determine a target priority, followed by a smoothed update:

$$h_i = w_D \widehat{D}_i + w_L \widehat{L}_i + w_C \widehat{C}_i, \qquad Q_i^{\text{new}} = (1-\alpha) Q_i^{\text{old}} + \alpha h_i. \tag{4}$$

We initialize $Q_i = 0$ and use $(w_D, w_L, w_C) = (0.65, 0.25, 0.10)$ and $\alpha = 0.15$. Hereafter, Q_i denotes the stored priority after the update. Only the observed task changes; others retain their scores. Thus, Q_i summarizes normalized feedback over time, rather than re-standardizing all tasks simultaneously. It is a sampling priority, not an action-value estimate.

To preserve coverage, we augment each priority with a visit-count bonus and a staleness bonus: $z_i = Q_i + b_i^{\text{visit}} + b_i^{\text{age}}$. The former favors tasks with fewer completed groups; the latter favors tasks not observed recently, measured in global group reservations. Equation 6 in Appendix A specifies both bonuses. After warmup, the distribution at request q is

$$\widetilde{p}_i^{(q)} = \frac{\exp(z_i/\vartheta)}{\sum_j \exp(z_j/\vartheta)}, \qquad p_i^{(q)} = p_{\min} + (1 - K p_{\min}) \widetilde{p}_i^{(q)}. \tag{5}$$

Here $\vartheta = 0.5$ is the sampling temperature, $p_{\min} = 0.03$, and $K p_{\min} < 1$. The distribution sums to one and satisfies $p_i^{(q)} \geq p_{\min}$. For ten tasks, 30% of its mass provides uniform coverage, and the remaining 70% follows the adaptive priorities. This floor limits task starvation in expectation but does not guarantee a visit within any particular finite batch.

4.4 Implementation of CRAFT

Algorithm 1 organizes each allocation call into three stages at a worker's full training-environment reset. Its inputs are task-consistent completed groups $\mathcal{B}$ and a request for M new groups.

Algorithm 1 CRAFT rollout allocation.

Input: i) completed groups $\mathcal{B}$; ii) K tasks, G trajectories per group, M requested groups; iii) shared state store and hyperparameters in Table 5.
Output: group reset plan $\mathcal{R} = ((i_b, \xi_b))_{b=1}^M$ and updated shared state.

/ Stage 1: Priority Update */*
$\mathcal{S} \leftarrow$ Begin
for $(i, \{r_g\}_{g=1}^G) \in \mathcal{B}$ in the supplied order **do**
  $\bar{r} \leftarrow G^{-1} \sum_{g=1}^G r_g, \quad v \leftarrow 4\bar{r}(1-\bar{r})$
  $s_i^f \leftarrow \mathcal{E}_{\beta_f}(s_i^f, \bar{r}; n_i), \quad s_i^s \leftarrow \mathcal{E}_{\beta_s}(s_i^s, \bar{r}; n_i)$ *// Eq. 1*
  $L_i \leftarrow \mathcal{E}_{\beta_L}(L_i, v; n_i), \quad n_i \leftarrow n_i + 1, \quad a_i^{\text{last}} \leftarrow \sum_j m_j$
  $D_i \leftarrow \max(\tau - s_i^s, 0)/(\tau - s^{\text{random}}), \quad C_i \leftarrow |s_i^f - s_i^s|$ *// Eq. 2*
  $\mathcal{M}_x \leftarrow$ Welford$(\mathcal{M}_x, x_i), \quad \forall x \in \{D, L, C\}$
  $\widehat{x}_i \leftarrow \text{Norm}(x_i; \mathcal{M}_x), \quad \forall x \in \{D, L, C\}$ *// Eq. 3*
  $h_i \leftarrow \sum_{x \in \{D,L,C\}} w_x \widehat{x}_i, \quad Q_i \leftarrow (1-\alpha) Q_i + \alpha h_i$ *// Eq. 4*
end for
Commit$(\mathcal{S})$
/ Stage 2: Task Reservation */*
$\mathcal{S} \leftarrow$ Begin, $\quad U \leftarrow$ Ordered$(\{i : m_i < 1\})$
if $U \neq \varnothing$ **then**
  $(i_1, \ldots, i_M) \leftarrow \text{Shuffle}_q(\text{Cycle}(U, M))$
else
  $z_i \leftarrow Q_i + b_i^{\text{visit}} + b_i^{\text{age}}, \quad \forall i$ *// Eq. 6*
  $p_i^{(q)} \leftarrow p_{\min} + (1 - K p_{\min}) \dfrac{\exp(z_i/\vartheta)}{\sum_j \exp(z_j/\vartheta)}, \quad \forall i$ *// Eq. 5*
  $i_b \overset{\text{iid}}{\sim} \text{Categorical}(p^{(q)}), \quad b = 1, \ldots, M$
end if
$m_i \leftarrow m_i + \sum_{b=1}^M \mathbb{I}[i_b = i], \quad \forall i; \qquad q \leftarrow q + 1$
Commit$(\mathcal{S})$
/ Stage 3: Group Initialization */*
$\xi_b \sim \rho_{0, i_b}, \quad b = 1, \ldots, M$
return $\mathcal{R} = ((i_b, \xi_b))_{b=1}^M$

The shared state $\mathcal{S}$ stores the task estimates, priorities, completed counts n_i, reservation counts m_i, last-observation counters a_i^{last}, moments $\mathcal{M}_x$, and allocation counter q.

In the algorithm, $\leftarrow$ denotes assignment: the right-hand side uses the stored value and the left-hand side receives the updated value, corresponding to the "old" and "new" notation above. Assignments update the corresponding fields of $\mathcal{S}$. A fresh run sets these quantities to zero; a resumed run restores a compatible state. Reservations include in-flight groups, so m_i and n_i have different roles.

Stage 1. Priority Update. CRAFT uses completed-group feedback to update task priorities. The EMA operator handles first observations before n_i increases. Each raw signal then enters its global Welford statistics before normalization, and only the observed task's priority changes. The routines BEGIN and COMMIT load the latest state under a lock and atomically save it before releasing that lock. This transaction preserves the updated feedback for subsequent allocation.

Stage 2. Task Reservation. CRAFT reloads the shared state in a separate transaction to include intervening worker updates, then reserves the next M groups. During warmup, $\text{Cycle}(U, M)_b = U_{1+((b-1) \bmod |U|)}$ fills the batch from the fixed ordered candidate list, and Shuffle_q permutes it using the request-specific RNG. Otherwise, tasks are sampled with replacement from Equation 5. The indicator $\mathbb{I}[i_b = i]$ counts reservations for each selected task. We save these counts and the updated request counter before execution.

Stage 3. Group Initialization. CRAFT samples uniformly from each selected task's available initial states via ρ_{0,i_b}. All G trajectories in a group share (i_b, ξ_b), and the returned plan controls subsequent collection rather than policy-update timing. The learner retains its existing optimization schedule within $U_{\max}$ rounds, and evaluation bypasses CRAFT. All collected groups remain available to the learner, including all-success and all-failure groups, while their outcomes inform future allocation. Appendix A gives the seed and persistence conventions. CRAFT therefore changes what to collect without introducing reward-group rejection or an additional policy objective.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We evaluate CRAFT on two benchmarks, LIBERO (Liu et al., 2023) and LIBERO-PRO (Zhou et al., 2025). On LIBERO, we evaluate the Spatial, Object, Goal, and Long task suites. On LIBERO-PRO, we evaluate perturbed versions of the ten LIBERO-Long tasks under four perturbation types: Object, Semantic, Position, and Task. The baseline, denoted π_{RL}, uses the $\pi_{0.5}$ policy backbone and the π_{RL} Flow-SDE training pipeline in RLinf (Chen et al., 2025; Yu et al., 2026). Our configuration uses group-relative advantages. Consequently, these are continuation experiments, not reproductions of the published PPO results. We denote the same pipeline augmented with CRAFT as $\pi_{\text{RL}}+\text{CRAFT}$; its matched control, π_{RL}, samples tasks uniformly.

Both conditions start from the RLinf-Pi05-PPO-LIBERO-130 checkpoint. This name identifies the initialization, not the advantage estimator used during continuation. Standard LIBERO training uses 64 parallel environments; LIBERO-PRO uses 32 environments and two collection epochs per update, yielding 64 rollouts per update. Both use groups of eight trajectories, a global batch size of 256, a micro-batch size of eight, a learning rate of 5×10^{-7}, and a KL coefficient of 0.001. We use full-shard FSDP and actor/rollout offloading, with a 480-step training horizon. CRAFT's hyperparameters are fixed across the reported routing runs (Table 5).

We use a 30-update budget to study short-horizon adaptation from a pretrained policy, rather than training to convergence. Both methods collect 64 rollout trajectories per update, giving each run the same budget of 1,920 trajectories. The standard LIBERO evaluation uses five denoising steps and 50 episodes per task, for a total of 500 per suite. LIBERO-PRO uses the protocol below. We report success rates in percent; all stated improvements are absolute differences in success rate, rather than relative percentage increases.

5.2 PERFORMANCE COMPARISON

Task Success Rates. Table 1 reports the performance of the four LIBERO suites after 30 policy updates. Under matched initialization and rollout budgets, CRAFT increases average success from

96.1% to 97.0%. The largest gain occurs on LIBERO-Long, where success rises from 90.2% to 94.2%, while LIBERO-Object improves from 99.2% to 99.8% despite its near-saturated baseline. Spatial and Goal decrease by 0.4 and 0.6 percentage points, respectively. These results show an aggregate improvement within the same adaptation budget, with benefits that vary across task suites. Figure 3 in Appendix B visualizes the complete four-suite comparison.

Table 1: Performance comparison on the LIBERO benchmark. Success rates (%) are reported for the Spatial, Object, Goal, and Long task suites. Bold marks column maxima, including ties. Difference (pp) denotes the success-rate difference between $\pi_{\mathrm{RL}} + \mathrm{CRAFT}$ and π_{RL}, in percentage points.

Method	Spatial	Object	Goal	Long	Average
SpatialVLA	88.2	89.9	78.6	55.5	78.1
CoT-VLA	87.5	91.6	87.6	69.0	81.1
DreamVLA	97.5	94.0	89.5	89.5	92.6
OpenVLA-OFT	97.6	98.4	97.9	94.5	97.1
VLA-Adapter	**97.8**	99.2	97.2	**95.0**	**97.3**
π_{RL}	97.0	99.2	**98.0**	90.2	96.1
$\pi_{\mathrm{RL}} + \mathrm{CRAFT}$	96.6	**99.8**	97.4	94.2	97.0
Difference (pp)	-0.4	$+0.6$	-0.6	$+4.0$	$+0.9$

Table 1 also includes published results from SpatialVLA (Qu et al., 2025), CoT-VLA (Zhao et al., 2025), DreamVLA (Zhang et al., 2025), OpenVLA-OFT (Kim et al., 2025a), and VLA-Adapter (Wang et al., 2026b). CRAFT achieves the highest Object success rate among the listed methods, although some methods obtain higher average success. Because their training and evaluation protocols differ, these results provide context rather than a direct comparison of training efficiency. Appendix C documents the corresponding protocols.

Generalization across Perturbations. We evaluate four official LIBERO-PRO perturbation families on the ten LIBERO-Long tasks (LIBERO-10). We adapt policies on the Object perturbation and evaluate them at updates 10, 20, and 30 on the Object, Semantic, Position, and Task perturbations. Evaluation uses a maximum episode length of 520 steps, five denoising steps, and action chunks of five. Each condition covers all ten tasks, with ten episodes per task for Object and five for each remaining condition. Success means reaching the goal at least once within the episode. The overall rate is episode-weighted across 250 episodes. All scheduled evaluations are complete.

Table 2: Success rates (%) on LIBERO-PRO after 10, 20, and 30 policy updates. Evaluation uses the ten LIBERO-Long tasks under four perturbation types: Object, Semantic, Position, and Task. Parentheses give differences from π_{RL} at the same update budget, in percentage points. Bold marks column maxima, including ties.

Updates	Method	Object	Semantic	Position	Task	Overall
10	π_{RL}	55.0	70.0	**22.0**	40.0	48.4
10	$\pi_{\mathrm{RL}} + \mathrm{CRAFT}$	56.0 (+1.0)	68.0 (-2.0)	14.0 (-8.0)	38.0 (-2.0)	46.4 (-2.0)
20	π_{RL}	55.0	66.0	16.0	**44.0**	47.2
20	$\pi_{\mathrm{RL}} + \mathrm{CRAFT}$	55.0 (+0.0)	62.0 (-4.0)	20.0 (+4.0)	40.0 (-4.0)	46.4 (-0.8)
30	π_{RL}	53.0	64.0	**22.0**	30.0	44.4
30	$\pi_{\mathrm{RL}} + \mathrm{CRAFT}$	**61.0** (+8.0)	**76.0** (+12.0)	**22.0** (+0.0)	40.0 (+10.0)	**52.0** (+7.6)

At 30 updates, CRAFT achieves 130 successful episodes out of 250, compared with 111 for π_{RL} (Table 2), raising overall success from 44.4% to 52.0%. In Table 2, parenthesized values indicate absolute success-rate differences between $\pi_{\mathrm{RL}} + \mathrm{CRAFT}$ and π_{RL} under the same evaluation condition and update budget, measured in percentage points. We compute the overall difference from the episode-weighted success rates. Object success increases from 53.0% to 61.0%, Semantic from 64.0% to 76.0%, and Task from 30.0% to 40.0%, while Position remains at 22.0% (Figure 2). The gains on Semantic and Task show transfer beyond the Object perturbation used for fine-tuning. Together with the four-suite LIBERO average, these results support aggregate benefits in both standard adaptation and perturbation transfer under matched update budgets. Bold values indicate the highest success rate in each column of Table 2, including ties.



Figure 2: Success rates on four LIBERO-PRO perturbation families of LIBERO-Long after 30 policy updates. Both methods use matched training and evaluation settings; Overall is weighted by episode count.

Table 3: Comparison of VLA methods on LIBERO-PRO. Success rates (%) are reported for the ten LIBERO-Long tasks under four perturbation types. Bold indicates the best result in each column, including ties.

Method	Object	Semantic	Position	Task
$\pi_{0.5}$ (benchmark)	**92.0**	93.0	8.0	1.0
SimVLA	61.0	98.0	3.0	10.0
π_0-FAST + SCALE	51.8	84.0	3.4	15.8
QuoVLA	82.0	**99.0**	21.0	24.0
SUREFlow	21.0	78.0	0.0	0.0
π_{RL}	53.0	64.0	**22.0**	30.0
π_{RL} + CRAFT	61.0	76.0	**22.0**	**40.0**

The checkpoint trajectory shows when these benefits emerge. Although CRAFT trails uniform allocation at updates 10 and 20, its overall success rises from 46.4% at both early checkpoints to 52.0% at update 30. Over the same checkpoints, the baseline declines from 48.4% to 47.2% and then 44.4%. CRAFT therefore yields a stronger final policy within the 30-update budget, with its advantage emerging over adaptation rather than at every stage.

Table 3 compares results on four LIBERO-PRO perturbation families of the Long tasks with published results from the LIBERO-PRO benchmark (Zhou et al., 2025), SimVLA (Luo et al., 2026), SCALE (Choi et al., 2026), QuoVLA (Wang et al., 2026a), and SUREFlow (Islam et al., 2026). We initialize our policies from RLinf-Pi05-PPO-LIBERO-130 and further adapt them on LIBERO-PRO Object. The benchmark's published $\pi_{0.5}$ result is not a matched evaluation of this continuation setup; allocation gains are therefore measured against our identically initialized π_{RL} baseline. After 30 updates, CRAFT reaches 40.0% Task and 22.0% Position success, above the listed external results, while its Object and Semantic scores remain below the best published values. The external protocols differ (Appendix C), so these differences cannot be attributed to training budget alone. In the matched comparison (Table 2), CRAFT improves Task success from 30.0% to 40.0% and Semantic success from 64.0% to 76.0%, while retaining the baseline's 22.0% Position success. The Task and Semantic gains indicate transfer beyond the Object perturbation used for fine-tuning.

Allocation Dynamics and Task-Level Gains. Task-level results reveal where the aggregate gains arise. On LIBERO-Long, CRAFT improves five tasks, matches four, and slightly reduces success on one; T3 and T9 gain 16 and 14 percentage points, respectively. Under the Object perturbation of LIBERO-PRO, four tasks improve, and six remain unchanged, with the largest gains on T1 and T9. The eight additional Object successes comprise three each on T1 and T9 and one each on T3 and T4 (Appendix F).

On LIBERO-Long, T8 receives the largest share but improves by only 2 percentage points, whereas T3 improves by 16 points with a smaller share. Allocation priorities reflect training-time feedback, whereas task-level gains measure final evaluation outcomes under a shared policy. These comparisons therefore characterize allocation behavior but do not isolate the causal effect of additional rollouts on an individual task. A detailed analysis is provided in Appendix F.

Qualitative Comparison. Appendix E complements the quantitative results with rollout examples from LIBERO-Long and its LIBERO-PRO Object-perturbed counterparts. The examples cover stove operation and moka-pot placement, drawer manipulation, and two-mug placement. For each task, initial, intermediate, and final frames contrast a failed π_{RL} rollout with a successful $\pi_{\mathrm{RL}} + \mathrm{CRAFT}$ rollout. These examples illustrate execution outcomes on the corresponding tasks rather than estimate the frequency of success.

5.3 Analysis of Early Adaptation

Task-Level Performance. To examine an earlier adaptation stage, we evaluate the step-10 Spatial checkpoints on the full 500-episode protocol. CRAFT achieves 485 successes (97.0%) versus 478 (95.6%) for π_{RL} (Table 4). It improves four tasks, ties five, and decreases one. Task 4 contributes the largest gain, while the weakest task, Task 5, declines slightly. The number of failures falls from 22 to 15, a 31.8% relative reduction. This evaluation covers all ten tasks.

Table 4: Task-level comparison on LIBERO-Spatial after 10 policy updates. Success counts (episodes) are reported over 50 evaluation episodes per task.

Method / task	0	1	2	3	4	5	6	7	8	9	Total
π_{RL}	50	48	50	49	45	40	50	47	49	50	478/500
$\pi_{\mathrm{RL}} + \mathrm{CRAFT}$	50	50	50	50	49	39	50	48	49	50	485/500

The primary comparison uses a common endpoint of 30 policy updates, while the 10- and 20-update results characterize adaptation before that endpoint. CRAFT's advantage is checkpoint-dependent. It emerges later on LIBERO-PRO, whereas the early advantage on LIBERO-Spatial reverses by update 30.

Performance across Training Seeds. After 10 policy updates, CRAFT reaches a mean LIBERO-Long success rate of $92.1 \pm 0.5\%$ across three training seeds, each evaluated over 500 episodes; $\pm$ denotes the population standard deviation across seeds (Appendix B, Table 6).

Comparison under Identical Initial Conditions. A separate paired evaluation of ten-update checkpoints uses the same ten initial states for each of the ten Long tasks. CRAFT succeeds in 95 of these 100 episodes, compared with 91 for uniform sampling (Appendix B, Table 7). These supplementary experiments characterize variation across training seeds and paired outcomes during early adaptation, separately from the primary 30-update comparison.

6 Conclusion

CRAFT treats task selection as a control point for online VLA adaptation with limited policy updates. Its feedback loop combines gaps to target success rates, variation in group outcomes, and changes in estimated success, then stabilizes the resulting task distribution while preserving coverage. Evaluations across all four LIBERO suites and four LIBERO-PRO perturbation families of LIBERO-Long show that this intervention can improve aggregate performance at a fixed update budget. Checkpoint analyses further show that its benefits depend on the suite and adaptation stage. These results show that task allocation can improve the use of a fixed interaction budget. CRAFT obtains the reported aggregate gains without additional policy updates or rollout trajectories relative to the matched baseline. CRAFT therefore complements policy optimization by changing which tasks supply training experience. Future work will evaluate CRAFT across additional VLA backbones and more challenging benchmarks such as RoboDojo, examining how feedback-guided allocation scales with task diversity and difficulty.

AI USE STATEMENT

Generative AI tools assisted with writing and language polishing. The authors take responsibility for the final manuscript and its scientific claims.

ETHICS STATEMENT

We adhere to the ICLR Code of Ethics. This work uses simulated manipulation benchmarks only and involves no human subjects or physical robot deployment. Benchmark performance does not imply real-world safety.

REPRODUCIBILITY STATEMENT

Section 4 and Algorithm 1 specify the task allocation rule. Appendix A provides implementation details. We will release the source code and configuration files upon acceptance.

REFERENCES

Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M. R., Finn, C., Fusai, N., Galliker, M. Y., Ghosh, D., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., LeBlanc, D., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Ren, A. Z., Shi, L. X., Smith, L., Springenberg, J. T., Stachowicz, K., Tanner, J., Vuong, Q., Walke, H., Walling, A., Wang, H., Yu, L., and Zhilinsky, U. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, volume 305, pp. 17–40, 2025.

Chen, K., Liu, Z., Zhang, T., Guo, Z., Xu, S., Lin, H., Zang, H., Li, X., Zhang, Q., Yu, Z., Fan, G., Huang, T., Wang, Y., and Yu, C. π_{RL}: Online RL fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025.

Chen, T., Chen, Y., Li, Z., Tang, J., Su, K., Lu, H., Wan, W., Chen, B., Liu, S., Yan, H., Su, H., Dou, Z., Wang, K., Zhang, D., Liu, Y., Qin, Y., Liang, Q., Wu, Q., Lin, Z., Lin, W., Wang, Y., He, M., Wu, T., Wu, R., Zhou, J., Lei, K.-C., Yu, H., Ji, Y., Jin, W., Lin, G., Li, X., Xiong, Q., Xu, R., Li, Z., Chai, W., Xie, E., Wang, Z., Mu, Y., Dong, H., Matusik, W., Ding, M., Ding, W., Luo, P., and Tomizuka, M. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026.

Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Choi, H., Ahn, D., Lee, Y., Kang, T., Cho, S., and Choi, J. SCALE: Self-uncertainty conditioned adaptive looking and execution for vision-language-action models. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Deng, Z., Lu, Y., Wang, Y., Liu, L., Ping, Q., Ding, H., Wu, G., Xu, P., and Huan, J. Prism-GRPO: Faster VLA policy optimization via splitting same-outcome groups. *arXiv preprint arXiv:2608.17423*, 2026.

Hu, Z., Qiu, J., Bai, T., Yang, H., Yuan, B., Jing, Q., He, C., and Zhang, W. VADE: Variance-aware dynamic sampling via online sample-level difficulty estimation for multimodal reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9846–9855, 2026.

Islam, M. T., Peddapalli, S. N., Lee, S., and Ahn, S. SUREFlow: State-space uncertainty-aware REsidual flow matching for robust robot manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.

Jiang, M., Grefenstette, E., and Rocktäschel, T. Prioritized level replay. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pp. 4940–4950, 2021.

Kim, M. J., Finn, C., and Liang, P. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025a.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E. P., Sanketi, P. R., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., and Finn, C. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, volume 270, pp. 2679–2713, 2025b.

Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 44776–44791, 2023.

Luo, Y., Chen, W., Liang, T., Wang, B., and Li, Z. SimVLA: A simple VLA baseline for robotic manipulation. *arXiv preprint arXiv:2602.18224*, 2026.

Qu, D., Song, H., Chen, Q., Yao, Y., Ye, X., Gu, J., Wang, Z., Ding, Y., Zhao, B., Wang, D., and Li, X. SpatialVLA: Exploring spatial representations for visual-language-action models. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

Rutherford, A., Beukman, M., Willi, T., Lacerda, B., Hawes, N., and Foerster, J. No regrets: Investigating and improving regret approximations for curriculum discovery. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 16071–16101, 2024.

Wang, X., Wu, Y., Duan, H., and Han, J. QuoVLA: Quotient space for vision-language-action models. *arXiv preprint arXiv:2605.24890*, 2026a.

Wang, Y., Ding, P., Li, L., Cui, C., Ge, Z., Tong, X., Song, W., Zhao, H., Zhao, W., Hou, P., Huang, S., Tang, Y., Wang, W., Zhang, R., Liu, J., and Wang, D. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18638–18646, 2026b.

Yu, C., Wang, Y., Guo, Z., Lin, H., Xu, S., Zang, H., Zhang, Q., Wu, Y., Zhu, C., Hu, J., Huang, Z., Wei, M., Xie, Y., Yang, K., Dai, B., Xu, Z., Du, J., Wang, X., Fu, X., Shi, L., Liu, Z., Chen, K., Liu, W., Liu, G., Li, B., Yang, J., Yang, Z., Dai, G., and Wang, Y. RLinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. In *Proceedings of the 20th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 829–846, 2026.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, J., Liu, L., Liu, X., Lin, H., Lin, Z., Ma, B., Sheng, G., Tong, Y., Zhang, C., Zhang, M., Zhang, R., Zhang, W., Zhu, H., Zhu, J., Chen, J., Chen, J., Wang, C., Yu, H., Song, Y., Wei, X., Zhou, H., Liu, J., Ma, W.-Y., Zhang, Y.-Q., Yan, L., Wu, Y., and Wang, M. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pp. 113222–113244, 2025.

Zhang, W., Liu, H., Qi, Z., Wang, Y., Yu, X., Zhang, J., Dong, R., He, J., Wang, H., Zhang, Z., Yi, L., Zeng, W., and Jin, X. DreamVLA: A vision-language-action model dreamed with comprehensive world knowledge. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

Zhao, Q., Lu, Y., Kim, M. J., Fu, Z., Zhang, Z., Wu, Y., Li, Z., Ma, Q., Han, S., Finn, C., Handa, A., Lin, T.-Y., Wetzstein, G., Liu, M.-Y., and Xiang, D. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1702–1713, 2025.

Zhou, X., Xu, Y., Tie, G., Chen, Y., Zhang, G., Chu, D., Zhou, P., and Sun, L. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

A Implementation Details

A.1 Feedback Statistics and Allocation Hyperparameters

Table 5 records the fixed parameters from the persisted configurations of the reported runs, rather than generic controller defaults.

Table 5: Task-allocation hyperparameters and reward-group retention settings used in CRAFT.

Setting	Value
Target / reference success rate	$0.9,\ 0.0$
Fast / slow / variation EMA decay	$0.5,\ 0.9,\ 0.9$
Deficit / variation / change weights	$0.65,\ 0.25,\ 0.10$
Priority update rate	0.15
Temperature / probability floor	$0.5,\ 0.03$
Visit-count / staleness coefficients	$0.15,\ 0.02$
Warmup reservations per task	1
Normalizer minimum count / ϵ	$2,\ 10^{-8}$
Reward-group rejection	Disabled

Outcome Variation and Estimate Initialization. The implementation computes $v = 2G^{-1}\sum_g |r_g - \bar{r}|$, which equals $4\bar{r}(1-\bar{r})$ for binary outcomes. On a task's first observation, its two success EMAs are initialized to $\bar{r}$ and its variation EMA to v. Subsequent observations apply the standard recurrence $s^{\text{new}} = \beta s^{\text{old}} + (1-\beta)y$, where β is the relevant decay and y is the new success or variation measurement. Thus, the decay weights the previous estimate. Unobserved tasks retain their state.

Online Normalization and Priority Updates. Global moments are updated using Welford's algorithm before applying Equation 3. Active signals are not clipped. Only the observed task's priority changes; other tasks retain scores computed under earlier moment estimates so that observation order can affect allocation.

Visit-Count and Staleness Bonuses. Let n_i count completed groups on task i and m_i count reserved groups, including outstanding reservations. Write $N = \sum_i n_i$ and $A = \sum_i m_i$. When a group observation is processed, set $a_i^{\text{last}} = A$. At allocation time, the bonuses are

$$b_i^{\text{visit}} = \lambda_v \sqrt{\frac{\log(N+K+1)}{n_i+1}}, \qquad b_i^{\text{age}} = \lambda_a \frac{\max(A - a_i^{\text{last}}, 0)}{\max(A, 1)}, \tag{6}$$

with $(\lambda_v, \lambda_a) = (0.15, 0.02)$ and initial $a_i^{\text{last}} = 0$. We measure staleness in global reservations since the latest received observation, not in seconds, optimizer updates, or time since rollout start. We recompute both bonuses from current counters when tasks are sampled.

A.2 Rollout Collection and Shared-State Management

Task-Consistent Group Collection. At a worker's full training-environment reset, CRAFT processes the preceding completed groups, updates task statistics, and constructs the next-task distribution. It reserves the requested groups before selecting a shared initial state for each group. The consistency check rejects mixed-task groups. Automatic per-environment resets are disabled so that task changes occur at the intended group boundary.

Warmup Reservations and Random Seeds. At the beginning of a request, the controller forms the ordered list $U = \{i : m_i < 1\}$. If it is nonempty, it fills all M requested slots cyclically from that fixed list and shuffles the batch. A task may receive more than one group when $M > |U|$; eligibility is checked again only on the next request. Otherwise, tasks are drawn with replacement using Equation 5. The experiment seed and allocation counter seed task sampling; initial states are sampled separately for each group.

Shared-State Transactions and Persistence. Workers share a file-locked controller store. Observation processing and task reservation use separate transactions that load, update, and atomically save the state. Reservations are saved before each worker's environment reset, without a global synchronization barrier. Restoration requires a compatible run and the corresponding policy and optimizer checkpoints; different worker schedules need not produce identical replay. Group feedback is processed at subsequent resets, and reservations persist after worker failures.

Group Retention and Policy Updates. CRAFT retains all collected groups and changes only task allocation, leaving the learner's group-retention rule unchanged. Homogeneous binary outcomes have zero-centered outcome signal, but this does not imply that every enabled loss term vanishes. Likewise, sampled-group proportions do not measure gradient contribution: trajectory lengths, masks, advantage normalization, and cancellation also affect the update.

B Additional Evaluation Results

B.1 Matched Comparison across LIBERO Task Suites

Figure 3 visualizes the complete matched comparison in Table 1. The point estimates use 500 evaluation episodes per suite after 30 updates. We show all four suites and their unweighted average, including the small decreases in Spatial and Goal. Both bar charts use a zero-based success-rate axis; the bars are point estimates, not uncertainty intervals.

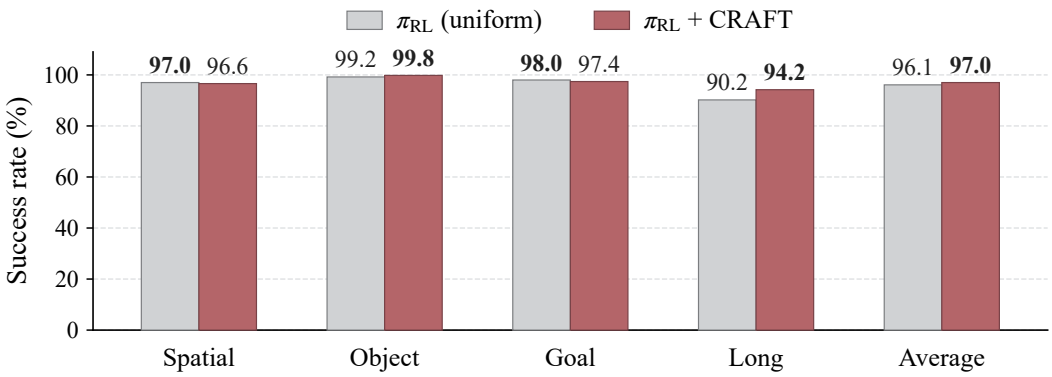

Figure 3: Success rates across the four LIBERO task suites after 30 policy updates under matched training and evaluation settings. Average is the unweighted mean across suites.

B.2 Training-Seed Variation on LIBERO-Long

At ten updates, we evaluated CRAFT over training seeds 0, 1, and 2 using the 500-episode Long protocol. Table 6 reports each run. Mean success is $92.1 \pm 0.5\%$, with the population standard deviation across seeds. The corresponding means of weakest-task and bottom-20% success are 68.7% and 75.3%. Bottom-20% success is the average of the two lowest task rates within each policy, computed before averaging across seeds.

Table 6: CRAFT success rates (%) on LIBERO-Long across three training seeds after 10 policy updates. Bottom 20% is the mean of the two lowest task success rates within each run.

Training seed	Mean success	Weakest task	Bottom 20%
0	91.8	68.0	75.0
1	92.8	64.0	74.0
2	91.6	74.0	77.0

A matched ten-update baseline aggregate is unavailable, so these runs do not estimate a multi-seed treatment effect. The suite-level improvements in the main 30-update comparison are reported in Section 5.2.

B.3 Paired Pilot Evaluation on LIBERO-Long

We separately evaluate ten-update checkpoints on ten fixed initial states per Long task. Both methods use five denoising steps, a ten-action chunk, and shared per-episode seeds. This yields 100 strictly paired episodes (Table 7), separate from the 500-episode benchmark.

Table 7: Per-task success counts in the paired LIBERO-Long pilot after 10 policy updates. Both methods use the same ten evaluation initial states per task.

Method / task	0	1	2	3	4	5	6	7	8	9	Total
π_{RL}	8	10	10	9	10	10	10	10	7	7	91/100
π_{RL} + CRAFT	10	10	10	10	10	10	10	10	9	6	95/100

CRAFT succeeds alone in eight pairs and the baseline alone in four; 87 pairs are joint successes, and one is a joint failure. An exact two-sided McNemar test gives $p = 0.3877$.

B.4 Evaluation Coverage and Score Aggregation

Table 4 reports the completed 500-episode Spatial evaluation using ten-update checkpoints, five denoising steps, and action chunk five.

The LIBERO-PRO table contains 250 episodes per method at each of three checkpoints. Its weighted overall rate is $(2R_{\mathrm{obj}}+R_{\mathrm{sem}}+R_{\mathrm{pos}}+R_{\mathrm{task}})/5$. Repeated checkpoints measure a training trajectory, not independent training seeds. We keep them separate from earlier evaluations conducted under different execution setups.

C Sources for Published Comparisons

The external rows in Table 1 reproduce published results for context. They differ in policy initialization, pretraining, observations, demonstrations, and inference, and are not re-evaluated controls for CRAFT.

Sources of Published LIBERO Results. SpatialVLA results come from Table III of the RSS 2025 paper (Qu et al., 2025); CoT-VLA results come from Table 1 of the CVPR 2025 paper (Zhao et al., 2025), with its published average rounded from 81.13 to 81.1.

DreamVLA's results come from Table 2 of the NeurIPS 2025 paper and include LIBERO-90 pretraining (Zhang et al., 2025). OpenVLA-OFT uses the additional-input variant in Table I of the RSS 2025 paper, including wrist images and state inputs, with filtered demonstrations (Kim et al., 2025a). VLA-Adapter results come from Table 5 of the AAAI 2026 paper (Wang et al., 2026b).

Sources of Published LIBERO-PRO Results. Table 3 uses only the LIBERO-10/Long results in each source, not LIBERO-Object or an average across base suites. The $\pi_{0.5}$ row comes from Table 4 of LIBERO-PRO v1 (Zhou et al., 2025); SimVLA from Table 3 of v1 (Luo et al., 2026); QuoVLA from Table 1(b) of v2 (Wang et al., 2026a); π_0-FAST + SCALE from Table 4 of v3 (Choi et al., 2026); and SUREFlow from Table II of v1 (Islam et al., 2026). We use the benchmark's official Object, Semantic, Position, and Task terminology throughout. We multiply normalized rates by 100; we leave values already reported as percentages unchanged. The table reports success, not error rates.

SimVLA and QuoVLA train on the four original LIBERO suites and evaluate PRO without further fine-tuning. SCALE uses inference-time adaptation with a LIBERO-trained backbone; SUREFlow uses a different state-space policy architecture. None matches our 30-update Object-adaptation protocol. We omit an external overall column because source macro-averages and multi-suite totals do not match our episode-weighted score.

D PRELIMINARY ROBODOJO INTEGRATION

We have implemented a RoboDojo evaluation pipeline (Chen et al., 2026) and begun preliminary integration trials of CRAFT using a block-level task-allocation adapter. These trials explore extending feedback-guided allocation beyond the LIBERO settings studied in the main experiments. Figure 4 shows diagnostic rollouts from the initial $\pi_{0.5}$ ARX X5 joint-action baseline used during pipeline development. We extract frames from each task's episodes directly from head-camera videos at 5%, 35%, 65%, and 95% of the video duration.

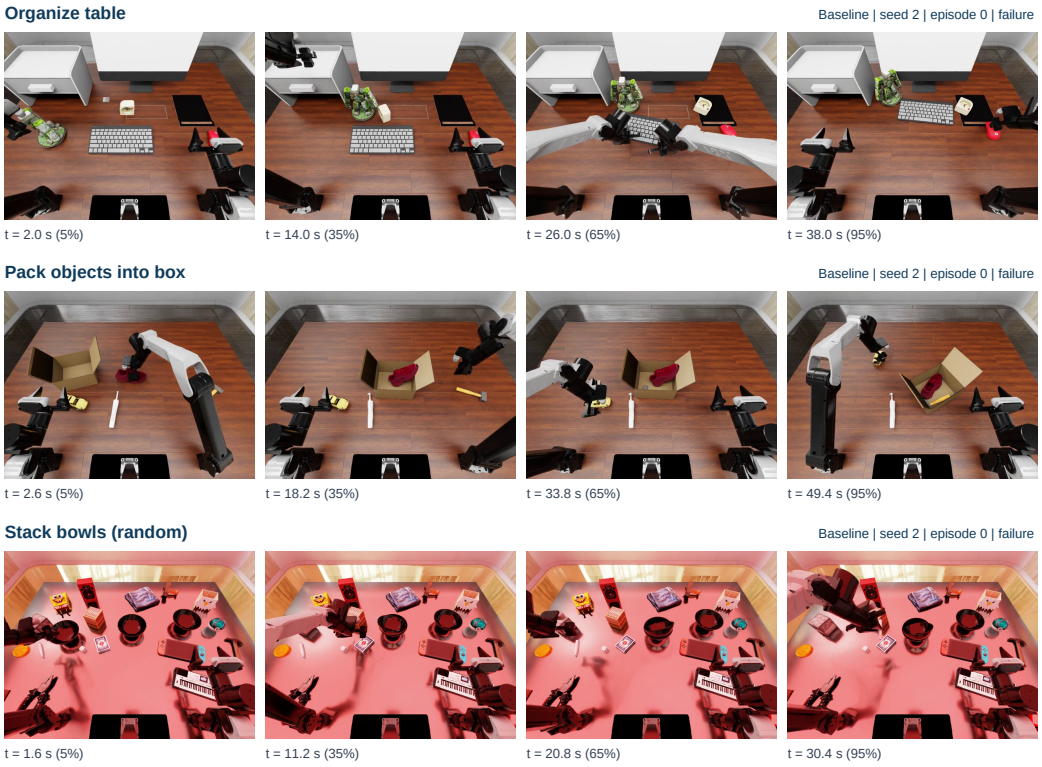

Figure 4: RoboDojo pipeline diagnostics using the initial $\pi_{0.5}$ baseline. Frames are sampled at 5%, 35%, 65%, and 95% of each video.

The recordings document pipeline validation rather than a matched comparison of allocation methods. We will continue the CRAFT integration trials with systematic held-out evaluation against matched uniform-allocation controls across a wider range of tasks.

E QUALITATIVE ROLLOUT EXAMPLES

Figures 5 and 6 contrast verified uniform-sampling failures with CRAFT successes on the same tasks from the matched 30-update runs. For each row, we select one failed trajectory from uniform sampling and one successful trajectory from CRAFT, then show frames at the beginning, middle, and end of each rollout. We take the success and failure labels from the recorded environment outcomes. The examples visualize representative behaviors; they are not paired estimates of success frequency and do not replace the aggregate results in the main experiment tables.

LIBERO examples. The three rows in Figure 5 correspond to distinct LIBERO-Long instructions. Example 1 asks the robot to turn on the stove and place the moka pot on it. Example 2 requires placing the black bowl in the bottom cabinet drawer and closing the drawer. Example 3 requires placing the white mug on the left plate and the yellow-and-white mug on the right plate. Thus,

the figure covers three different long-horizon tasks rather than three trajectories from a single instruction.

LIBERO-PRO examples. Figure 6 uses the same task set under the Object perturbation evaluated in our 30-update LIBERO-PRO experiment. Its rows are ordered as the drawer-and-bowl task, the two-mug placement task, and the stove-and-moka-pot task. Within each row, the instruction and perturbation family are matched across methods, while the displayed trajectories retain their observed outcomes.

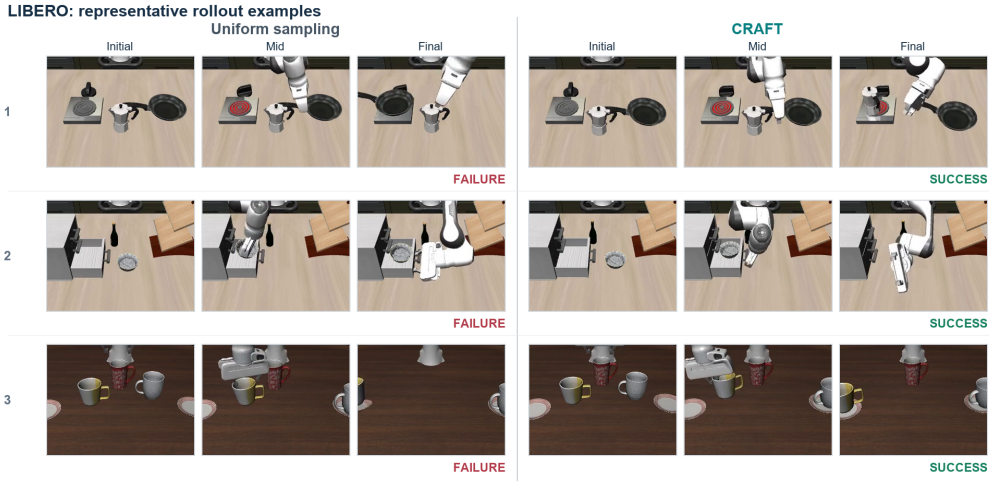

Figure 5: Representative LIBERO rollout comparisons. Uniform sampling fails in the shown trajectories, whereas CRAFT succeeds on the corresponding tasks.

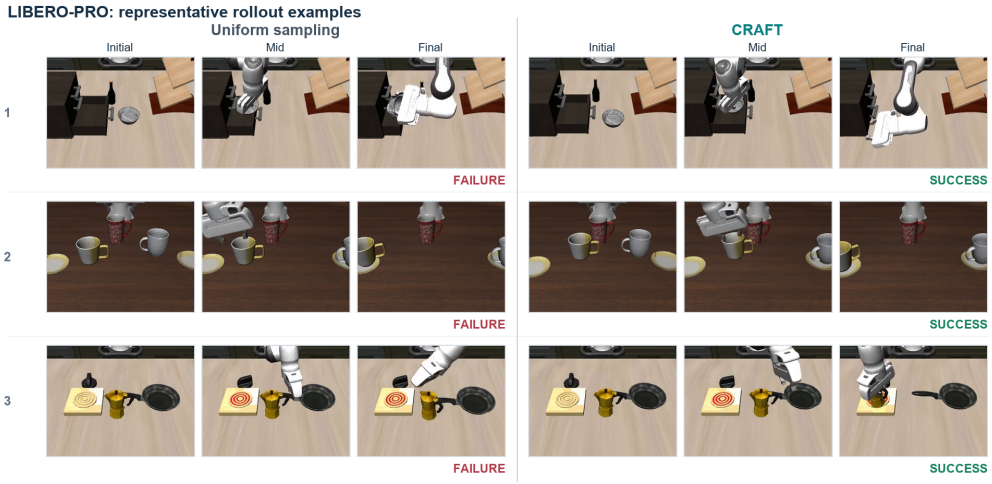

Figure 6: Representative LIBERO-PRO Object-perturbation rollout comparisons. Uniform sampling fails in the shown trajectories, whereas CRAFT succeeds on the corresponding tasks.

F Task Allocation and Task-Level Performance

F.1 Task Allocation Dynamics

Figures 7 and 8 visualize rollout-group allocation by π_{RL} + CRAFT during training on LIBERO-Long and the Object perturbation of LIBERO-PRO, respectively. Each figure shows how task-sampling probabilities evolve during collection and the fraction of all allocated groups assigned to each task. Each run completes 30 policy updates and allocates 240 groups. The horizontal axis counts allocated groups, not policy updates. The gray region marks the initial coverage warm-up, and the dashed line indicates the expected 10% allocation share per task under the π_{RL} sampling scheme.

On LIBERO-Long, sampling increasingly concentrates on T8 later in training, while the probability assigned to T3 decreases. On LIBERO-PRO, allocation remains more distributed, with T7 receiving the largest cumulative share. These examples illustrate how CRAFT adjusts task allocation during training rather than maintaining a fixed task-sampling distribution.

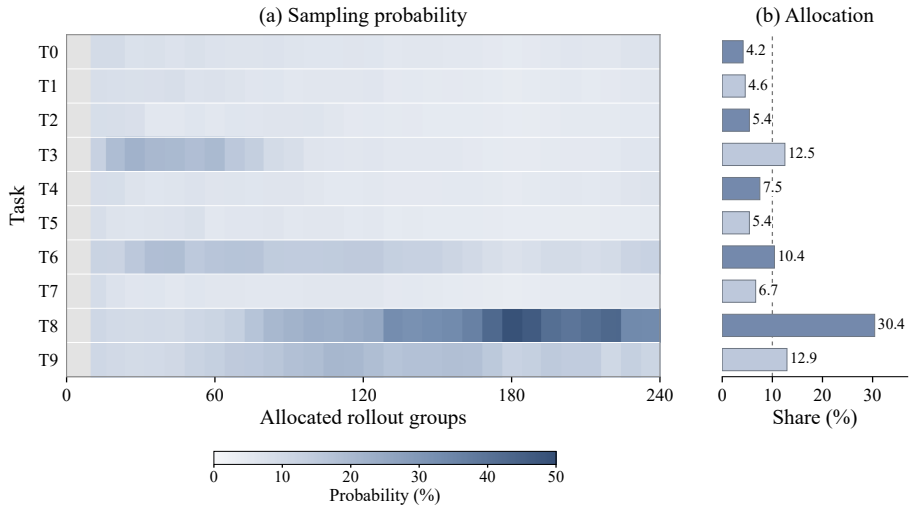

Figure 7: Task-sampling probabilities and cumulative allocation shares of π_{RL} + CRAFT on LIBERO-Long.

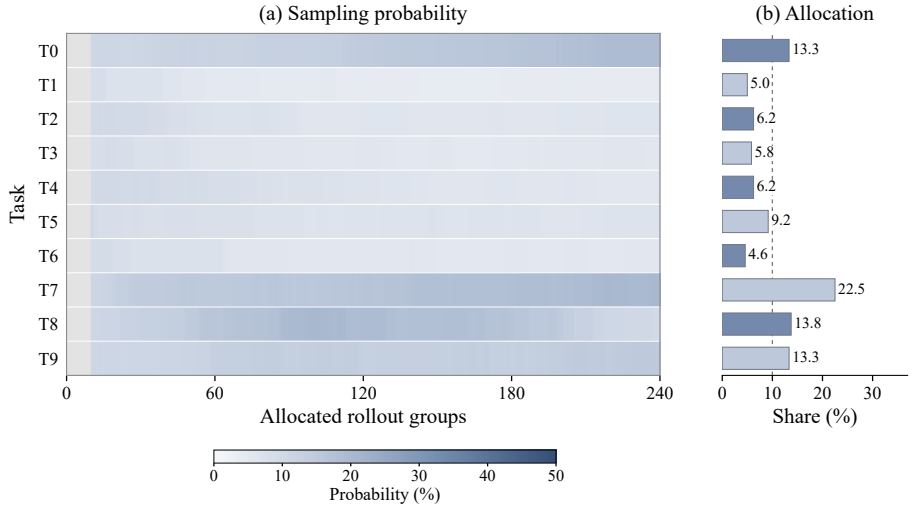

Figure 8: Task-sampling probabilities and cumulative allocation shares of π_{RL} + CRAFT under the Object perturbation of LIBERO-PRO.

F.2 Task-Level Performance

Figures 9 and 10 compare the per-task success rates of π_{RL} and $\pi_{\mathrm{RL}} + \mathrm{CRAFT}$ after 30 policy updates, using the same task indices as the allocation plots. We evaluate each task over 50 episodes on LIBERO-Long and 10 episodes on LIBERO-PRO Object. The lower panels report differences between the two methods in percentage points, rather than changes from the initialization checkpoint.

On LIBERO-Long, CRAFT improves five tasks, matches four, and slightly reduces performance on one; T3 and T9 improve by 16 and 14 percentage points, respectively. On LIBERO-PRO Object, T1 and T9 each improve by 30 percentage points, while T3 and T4 each improve by 10 percentage points; the remaining tasks are unchanged. These results show that gains vary across tasks and do not correspond directly to cumulative allocation shares.

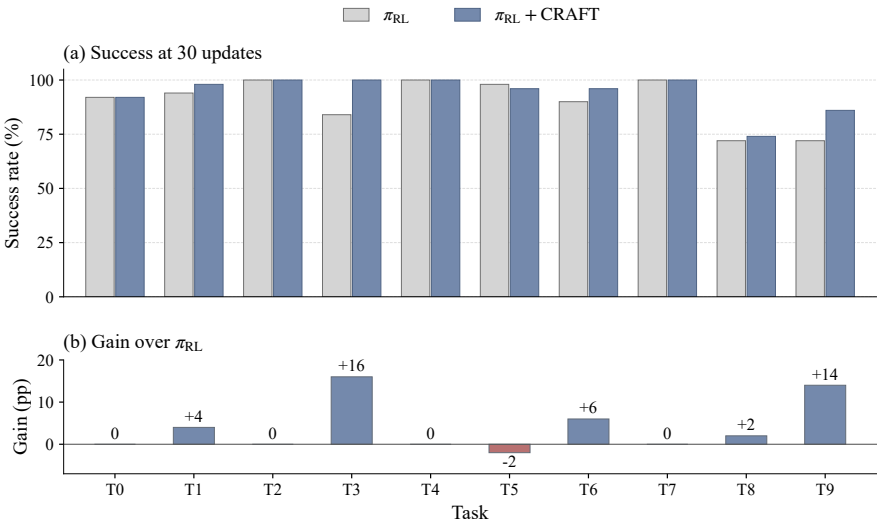

Figure 9: Per-task success rates and gains over π_{RL} on LIBERO-Long after 30 policy updates.

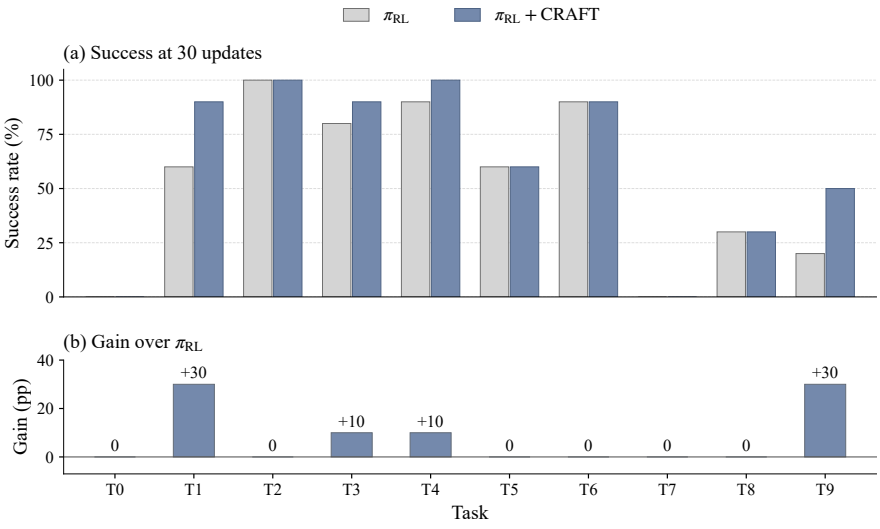

Figure 10: Per-task success rates and gains over π_{RL} under the Object perturbation of LIBERO-PRO after 30 policy updates.

Overall, these analyses provide complementary views of CRAFT: the allocation plots show how collection priorities evolve. At the same time, the task-level comparisons reveal where performance gains emerge under the same update budget. The results indicate that improved aggregate performance does not require equal gains across tasks, nor does a larger allocation share necessarily translate into a larger improvement. Instead, CRAFT uses group feedback to continually adjust collection priorities while preserving opportunities to revisit all tasks. This supports viewing adaptive allocation as a mechanism for managing limited training opportunities, rather than a guarantee of improvement on every task.