LEARNING TO RECOVER: ITERATIVELY INTERNALIZING AGENT CORRECTIONS INTO VLAS FOR ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

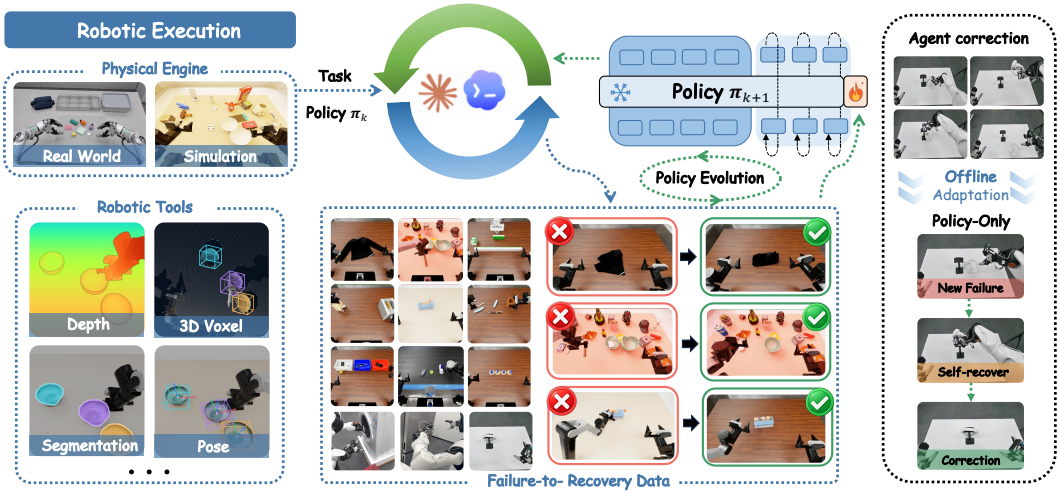

Figure 1: **The proposed L2R framework.** Left: verified agent corrections support lightweight offline adaptation; remaining policy failures drive further rounds. Right: after adaptation, the policy recovers from a new failure without online agent assistance.

ABSTRACT

Vision-language-action (VLA) models acquire manipulation skills through large-scale training, while reasoning agents can use tools and policies to handle situations beyond those skills. Yet successful assistance does not itself improve the underlying policy, leaving subsequent executions dependent on renewed online reasoning. We present **Learning to Recover (L2R)**, an agent-to-policy learning framework that iteratively internalizes agent corrections into reusable recovery capabilities. At natural failures, the agent constructs and verifies corrections; lightweight offline updates then teach the policy to reuse them without online agent assistance. Rather than imitating entire intervention records, L2R selectively learns valid corrective actions and when to invoke them through conditional low-rank adaptation of a frozen VLA and a lightweight local controller. Further rounds target failures of the updated policy, allowing recovery learning to follow the policy's changing behavior. On ten VLABench task families across five tracks, a single round trained only on in-distribution corrections achieves 52.0% conditional recovery, compared with 31.8% for standard LoRA. Three rounds improve policy-only task success from 12.6% to 27.4% on ten RoboDojo tasks and from 33.3% to 68.3% on three real-world tasks. These results show how limited agent corrections can be progressively internalized into autonomous recovery capabilities. Videos are available on our project page.

1 INTRODUCTION

Vision-language-action (VLA) models provide a foundation for general-purpose robot manipulation (Brohan et al., 2023; O'Neill et al., 2024; Kim et al., 2024; Black et al., 2024). Large-scale

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.

pretraining and robot post-training endow these models with broad manipulation capabilities, yet deployment inevitably exposes them to states that are poorly covered by demonstrations: objects shift, contacts fail, and long-horizon execution deviates from the intended trajectory (Liu et al., 2026). Robust deployment therefore requires not only broad pretrained capabilities, but also the ability to improve from failures encountered during execution (Sun et al., 2025; Zheng et al., 2025).

Reasoning agents provide a complementary mechanism for handling such execution failures. Multimodal foundation models connect visual observations with semantic knowledge and reasoning (Driess et al., 2023), enabling agents to interpret execution feedback and coordinate policies, perception tools, and executable skills (Ahn et al., 2022; Huang et al., 2022; Liang et al., 2023; Xu et al., 2026). Harness VLA, for example, treats a frozen VLA as a retryable manipulation primitive within a memory-guided agent (Zhang et al., 2026b). By reasoning over execution feedback and selectively invoking these components, an agent can diagnose failed executions and construct corrective behaviors beyond the nominal policy's immediate capabilities.

Yet a more capable agent does not necessarily yield a more capable action policy. When the action policy is left unchanged, corrective behaviors discovered during agent intervention are not retained, so similar failures may still require renewed orchestration, incurring additional latency and costs. Internalizing these corrections through lightweight policy-side learning is therefore desirable but nontrivial. A successful intervention may include failed attempts, diagnostic pauses, effective corrections, and nominal continuation, so terminal success does not make every action a valid imitation target. Recovery data also concentrate on atypical states and may not improve execution from standard task starts (Liu et al., 2026). Learning corrective actions alone further does not teach the policy when to invoke them. We therefore distinguish action supervision from recovery-invocation supervision, learning both which corrective behaviors to reproduce and when to enter, route, and exit recovery. This leads to our central question: *Can a small number of verified agent corrections be progressively internalized into reusable recovery capabilities through successive policy updates?*

We introduce **Learning to Recover (L2R)**, a framework that alternates agent-assisted correction with lightweight offline recovery learning (Figure 1). At natural failures, the agent invokes the VLA, spatial tools, and atomic skills, and verifies its corrections through execution feedback. Rather than imitating entire intervention records, L2R extracts selective action targets together with supervision for when and how recovery should be invoked. A conditional low-rank adapter learns corrective behaviors, while a compact local controller learns when to enter recovery, which recovery intent to use, and when to return to nominal execution; the base VLA parameters remain frozen. At deployment, the resulting policy performs recovery autonomously without online agent assistance.

L2R further improves the policy through successive rounds of correction and learning. Because each update changes the states visited and failures encountered by the policy (Ross et al., 2011), corrections collected from the original policy may no longer reflect its remaining weaknesses. Each subsequent round therefore targets failures of the current policy and combines newly verified corrections with earlier experience for another offline update. In this way, agent assistance is progressively internalized into the policy while supervision tracks its evolving failure distribution.

Our contributions are:

- **Learning to Recover from agent corrections.** We introduce L2R, an agent-to-policy framework that converts verified interventions into selective corrective-action and recovery-invocation supervision, enabling autonomous recovery without imitating failed actions.
- **Lightweight and iterative recovery learning.** L2R combines conditional low-rank adaptation with a local recovery controller over a frozen VLA, and iteratively targets failures of the updated policy to internalize new corrections as its behavior evolves.
- **Recovery and autonomous improvement.** Evaluations across VLABench, RoboDojo, and real-world tasks show recovery adaptation and multi-round gains in policy-only success. Controlled studies examine selective supervision, adaptation choices, and current-policy failure acquisition.

2 Related Work

Pretrained policies and efficient adaptation. Robot datasets and generalist policies support broad manipulation (Brohan et al., 2022; Walke et al., 2023; Khazatsky et al., 2024; Octo Model

Team et al., 2024), with chunked, diffusion-based, and flow-based action generation (Zhao et al., 2023; Chi et al., 2025; Physical Intelligence et al., 2025; Zhang et al., 2026a; Team et al., 2026). Data quality and feature preservation affect adaptation (Lin et al., 2025a; Mandlekar et al., 2021; Kumar et al., 2022), while adapters, low-rank updates, and feature-wise conditioning enable compact, conditional specialization (Houlsby et al., 2019; Hu et al., 2021; Liu et al., 2024c; Perez et al., 2018). We use these principles for task-level recovery after nominal post-training.

Embodied agents and corrective learning. Agents combine grounded skills, multimodal reasoning, feedback, and tools (Ahn et al., 2022; Driess et al., 2023; Huang et al., 2022; Yao et al., 2022; Liang et al., 2023), including spatial reasoning, planning, and pose tracking (Huang et al., 2023; Rana et al., 2023; Yuan et al., 2025; Wen et al., 2024). Harness VLA orchestrates frozen primitives through agent memory (Zhang et al., 2026b). Corrective learning instead updates policies from learner-visited states, perturbed demonstrations, or human interventions (Ross et al., 2011; Laskey et al., 2017; Kelly et al., 2019; Liu et al., 2025); monitoring, fleet learning, and policy-generated data support continued improvement (Liu et al., 2024a;b; Bousmalis et al., 2023). VLAW uses real rollouts to improve a world model that generates VLA training data (Guo et al., 2026). SAFE detects failures from internal features (Gu et al., 2025), and FPC-VLA adds failure prediction and correction (Yang et al., 2026). Our agent supplies verified corrections for offline learning; the updated policy manages recovery without online agent assistance.

Recovery and distillation. Recovery RL addresses constraint avoidance (Thananjeyan et al., 2021), whereas LIBERO-RECOVER evaluates manipulation recovery beyond nominal success (Liu et al., 2023; 2026). FailSafe pairs failures with executable recoveries (Lin et al., 2025b); RePO-VLA uses recovery-aware initialization and value-conditioned refinement (Liufu et al., 2026). Residual RL modifies control outputs (Johannink et al., 2019). PLD combines residual RL and distribution-aware collection to produce trajectories for VLA distillation (Xiao et al., 2026). PPS steers a frozen sampler through proxy velocity fields (Ning et al., 2026). We instead learn conditional low-rank increments inside a frozen action DiT from verified agent corrections. This transfer of training-time assistance connects to knowledge, policy, and cross-modal distillation (Hinton et al., 2015; Rusu et al., 2015; Yuan et al., 2022); our supervision covers corrective actions and their invocation, with further rounds targeting the updated policy's failures.

3 METHOD

L2R alternates agent-assisted recovery collection with lightweight offline policy learning (Figure 2). We first formalize the nominal policy, recovery records, and round-wise student in Section 3.1. At natural failures, a collection-time agent diagnoses the failure, orchestrates policies, tools, and skills to construct a verified correction, and converts the resulting intervention record into selective supervision (Section 3.2). We then learn corrective action generation through conditional low-rank adaptation (Section 3.3) and autonomous recovery invocation through a local controller (Section 3.4). Finally, subsequent rounds collect corrections from failures of the updated policy, enabling recovery learning to follow its evolving behavior (Section 3.5).

3.1 PRELIMINARIES

We consider a nominal VLA policy π_{θ_0} that, at each policy query t, maps camera observations o_t, proprioception q_t, and task instruction ℓ to an action chunk $A_t = (a_{t+1}, \ldots, a_{t+H}) \in \mathbb{R}^{H \times d_a}$, where H is the prediction horizon and d_a is the action dimension. The robot executes a prefix of the predicted chunk before querying the policy again. Throughout recovery learning, the base parameters θ_0 remain frozen.

During execution, a *failure source* is a naturally generated rollout that fails the task criterion. A *recovery boundary* marks the point at which corrective intervention begins. Starting from this boundary, an *intervention record* preserves the failure context, corrective attempts, execution outcomes, and verification feedback used for subsequent learning.

After round k, the student policy is $\Pi_k = (\pi_{\theta_0,\phi_k}, G_{\psi_k})$, with $\Pi_0 = \pi_{\theta_0}$. Here ϕ_k parameterizes the recovery adaptation of the action policy, while G_{ψ_k} controls when recovery is invoked and which recovery intent is used. *Policy-only execution* deploys Π_k without online agent assistance.

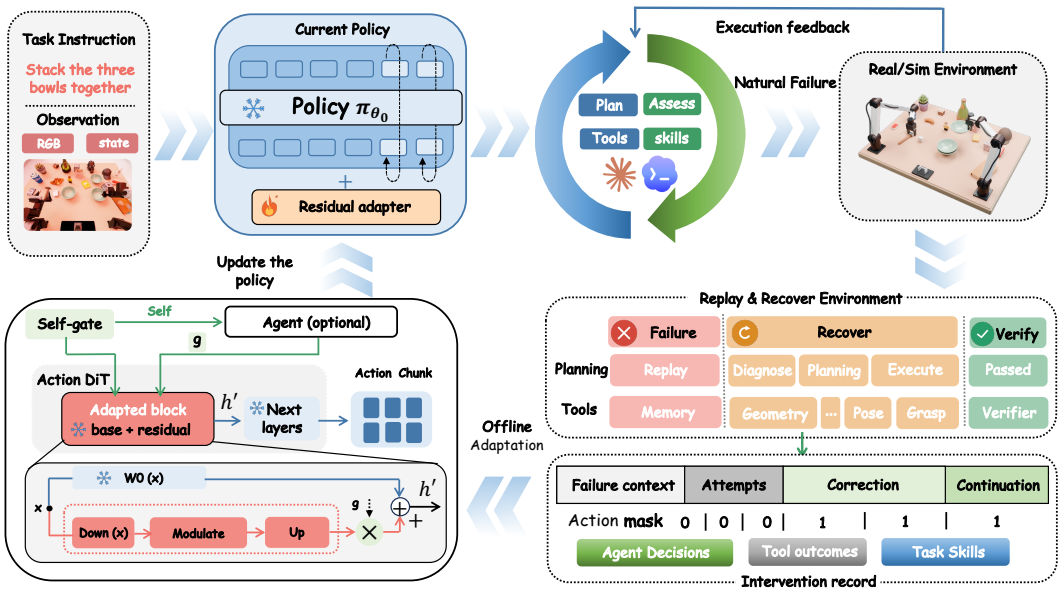

Figure 2: **L2R: from intervention to reusable recovery.** Top: the collection-time agent orchestrates the current policy, tools, and skills using execution feedback. Bottom right: recovery and verification produce intervention records for selective action and recovery-control supervision. Bottom left: offline training updates the conditional low-rank adapter and local controller. The optional agent branch is used only during collection; policy-only execution uses local recovery control. Remaining failures initiate another collection round.

3.2 Agent-Assisted Recovery and Supervision

At a recovery boundary, the collection-time agent receives the task instruction ℓ, current observations o_t, robot state q_t, and execution feedback, and reasons about how to recover from the encountered failure. Rather than directly replacing the action policy, the agent orchestrates a set of callable components. It can query the VLA for action generation, use spatial and perception tools to inspect the scene, and invoke atomic manipulation skills such as grasp, move, and place. In simulation, read-only object poses and task-verifier feedback provide additional collection-time information, while real-world recovery relies on camera observations and robot-state feedback.

Recovery is performed as a closed-loop process. The agent first diagnoses the failure and selects a corrective operation, target, and execution arm. It then invokes the appropriate policy or skill, observes the resulting state and execution feedback, and decides whether to continue, retry, or revise the correction. For example, a failed grasp can trigger regrasping before transport resumes. This assess-plan-execute loop continues until the correction is accepted by the task verifier or the intervention terminates. The resulting intervention record retains the failure context, intermediate attempts, agent decisions, tool outcomes, corrective execution, and subsequent continuation.

A verified intervention does not make every recorded action a valid imitation target. We therefore derive action and recovery-invocation supervision separately. An action mask $M_t \in \{0,1\}^{H \times d_a}$ selects executed commands in approved recovery or continuation intervals while preserving their native temporal alignment. Entry, route, and verified handoff annotations supervise when to activate recovery, which recovery intent to use, and when to return to nominal execution. Successful nominal traces additionally provide negative entry examples. Consequently, failure observations can remain useful for learning recovery invocation even when the associated actions are excluded from imitation. Appendix A describes the record representation and target construction.

3.3 Conditional Recovery Adaptation

We learn corrective actions by modulating a low-rank increment inside the frozen action DiT (Peebles & Xie, 2023). The adapter reuses the base visual-language and motor representations while adjusting its action computation for the recovery intent. It changes intermediate projections that

generate actions, rather than adding an offset to decoded robot commands. A separate local controller determines when this recovery branch is active, as described in Section 3.4.

Condition-modulated increments. The condition encoder E_ϕ maps the recovery condition z_t to a recovery embedding. We combine it with the controller's local-history representation s_t through $c_t = E_\phi(z_t) + P_{\text{ctx}}(\text{sg}(s_t))$, where sg stops gradients and P_{ctx} is a zero-initialized affine projection included in ϕ. The discrete condition identifies the recovery intent, while the history representation supplies context from recent observations.

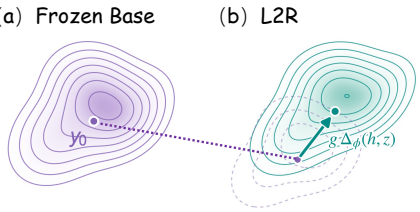

Figure 3: L2R conditional adapter.

For input h at frozen projection j, rank r, and scale α, the adapted computation is

$$m_j(c_t) = 1 + \tanh(P_j c_t + d_j), \tag{1}$$

$$\Delta_{\phi,j}(h, c_t) = \frac{\alpha}{r} U_j \big[(D_j h) \odot m_j(c_t)\big], \tag{2}$$

$$h' = W_j^0 h + b_j^0 + g_t \Delta_{\phi,j}(h, c_t). \tag{3}$$

Here $D_j \in \mathbb{R}^{r \times d_j^{\text{in}}}$ and $U_j \in \mathbb{R}^{d_j^{\text{out}} \times r}$ are the low-rank factors, $P_j \in \mathbb{R}^{r \times d_c}$ and $d_j \in \mathbb{R}^r$ parameterize modulation, and d_c is the condition dimension. The condition encoder is shared across projections; the low-rank factors and modulation maps are projection-specific.

The vector $m_j(c_t)$ reweights the low-rank channels according to recovery context, whereas the binary indicator g_t enables or disables the branch. For fixed c_t, the induced weight increment has rank at most r. Setting $m_j \equiv 1$ gives an ordinary LoRA increment (Hu et al., 2021). Ordinary LoRA already depends on observations through h; our additional condition pathway explicitly controls how the low-rank channels contribute. Zero-initialized up and modulation maps give $\Delta_{\phi,j} = 0$ and $m_j = 1$ initially. When $g_t = 0$, the projection recovers the base computation for identical inputs and execution settings.

Masked action learning. The adapter is trained on the selective targets from Section 3.2, retaining the base flow-matching objective (Lipman et al., 2022). For action chunk A, Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, and noise time u drawn from the base schedule, we construct

$$X_u = (1 - u)\epsilon + uA, \qquad V^\star = A - \epsilon. \tag{4}$$

We sample eligible windows ξ according to $p_{\mathcal{D}}(\xi) \propto \omega(\xi)$, using the hierarchical inverse-count weights. For a batch $\mathcal{B} \sim p_{\mathcal{D}}$, action learning enables the adapter and uses annotated conditions: $v_b = v_{\theta_0,\phi}(X_{u_b}, u_b, o_b, q_b, \ell_b, c_b, 1)$. Let $N_{\mathcal{B}} = \sum_{b \in \mathcal{B}} \sum_{i=1}^{H} \sum_{d=1}^{d_a} M_{bid}$ count valid action coordinates. For $N_{\mathcal{B}} > 0$, the loss is

$$\mathcal{L}_{\text{vel}}(\phi) = \mathbb{E}_{\mathcal{B},\epsilon,u}\left[\frac{1}{N_{\mathcal{B}}} \sum_{b \in \mathcal{B}} \sum_{i=1}^{H} \sum_{d=1}^{d_a} M_{bid} w_{bid} (v_{bid} - V^\star_{bid})^2\right]. \tag{5}$$

Normalization counts supervised coordinates rather than the full padded chunk. The inherited loss weights w weight action errors, independently of the sampling weights ω that select training windows. Excluded coordinates do not contribute to the action loss.

3.4 Policy-only recovery execution

The action objective teaches a correction under a supplied recovery condition. Autonomous execution also requires the policy to infer that condition and decide when the correction is needed. We train a local controller on the entry, route, and exit labels from Section 3.2 so that these decisions no longer require the collection-time agent.

The controller receives a causal history $\mathcal{H}_t$ of frozen visual features, the global task instruction, proprioception, and timing/mode indicators. Visual features are taken before recovery-prompt conditioning; teacher answers and privileged simulator state are excluded. A GRU encodes up to the

most recent eight policy queries as s_t, and three heads predict entry, exit, and route. These predictions govern the local execution mode b_t and route ρ_t, which determine branch activation and conditioning:

$$(p_t^{\text{on}}, p_t^{\text{off}}, p_t^{\text{route}}) = G_\psi(\mathcal{H}_t), \qquad g_t = \mathbf{1}[b_t = \text{RECOVER}], \quad z_t = \mathsf{C}(\rho_t, b_t). \tag{6}$$

Entry and exit use sigmoid heads; the route head is categorical. The encoding C maps a legal mode-route combination to the same condition representation used in action training. The original task instruction is retained in both nominal and recovery execution.

During nominal execution, the controller evaluates whether to enter recovery. On activation, it selects a route and enables the adapter. The route persists while the action policy responds to fresh observations; a confirmed exit disables the branch and returns control to the base path. Thresholds, consecutive confirmations, and holding rules stabilize transitions at policy-query boundaries. Safety constraints can interrupt execution, and unresolved recovery follows the fixed abstention rule. Task termination is assessed separately by the task verifier, not by the local exit prediction. Thus, entering recovery, choosing its intent, and returning to nominal execution are all local decisions; policy-only evaluation includes no assisted fallback.

Joint training with distinct supervision. For a source-sampled control batch $\mathcal{B}_g$, let μ_t^h indicate whether head h has a valid target y_t^h. The controller loss is

$$\mathcal{L}_{\text{gate}}(\psi) = \sum_{h \in \{\text{on},\text{off},\text{route}\}} \lambda_h \frac{\sum_{t \in \mathcal{B}_g} \mu_t^h \, \ell_h(p_t^h, y_t^h)}{\sum_{t \in \mathcal{B}_g} \mu_t^h}, \tag{7}$$

where λ_h weights each head, ℓ_h is binary cross-entropy for entry/exit and categorical cross-entropy for routes, and terms with no valid targets are omitted. The control masks μ are distinct from the action mask M: excluding a failed action from imitation does not discard its valid boundary or route supervision. We train the two components within the same offline update procedure:

$$\min_{\phi,\psi} \; \mathcal{L}_{\text{train}}(\phi, \psi) = \mathcal{L}_{\text{vel}}(\phi) + \mathcal{L}_{\text{gate}}(\psi), \qquad \theta_0 \text{ fixed}. \tag{8}$$

The action loss updates the low-rank modules, condition encoder, and P_{ctx}, but not the GRU; entry, route, and exit supervision train the controller. Gradients stop at the frozen feature extractor and do not cross the hard execution switch. This separates learning corrective actions from learning their invocation while producing a policy that uses both at deployment. Controller architecture and action-adaptation settings are given in Appendix A.

3.5 Iterative policy adaptation

Algorithm 1 L2R: iterative policy adaptation from verified corrections

Require: Frozen base θ_0; initialization (ϕ_0, ψ_0); number of rounds K; per-round update budgets S_k.
1: Set $\Pi_0 = \pi_{\theta_0}$ and replay $\mathcal{R}_0 = \varnothing$.
2: **for** $k = 0, \ldots, K-1$ **do**
3: Execute Π_k; retain nominal traces and obtain verified corrections for natural failures.
4: Compile action and control views $\mathcal{P}_{k+1}^{\text{act}}, \mathcal{P}_{k+1}^{\text{gate}}$, including eligible replay from $\mathcal{R}_k$.
5: Jointly update $(\phi_k, \psi_k) \to (\phi_{k+1}, \psi_{k+1})$ with $\mathcal{L}_{\text{train}}$ for S_k steps; keep θ_0 fixed.
6: Set $\Pi_{k+1} = (\pi_{\theta_0, \phi_{k+1}}, G_{\psi_{k+1}})$.
7: Select $\mathcal{R}_{k+1}$ from accumulated training sources under the fixed replay rule.
8: **end for**

Recovery learning changes the policy's execution distribution: once earlier corrections are internalized, the updated policy reaches different states and exhibits different remaining failures. L2R therefore collects supervision from the current student rather than repeatedly correcting failures of the original Base. Specifically, R1 learns from Base failures, while round $k+1$ executes the complete student Π_k and obtains verified corrections for its remaining failures.

Each round inherits the adapter and controller from the preceding student, combines newly collected supervision with replay from earlier rounds, and performs another offline update while keeping θ_0 frozen. The resulting student then executes policy-only before the next collection round. Algorithm 1 summarizes this iterative correction-learning loop.

4 EXPERIMENTS

We separate recovery competence from autonomous use: VLABench evaluates adaptation at supplied failure boundaries (Section 4.2), while RoboDojo and real-world trials evaluate policy-only execution and successive updates (Sections 4.3 and 4.4). Controlled acquisition and ablation studies examine where further corrections come from and how they are learned (Sections 4.3 and 4.5).

4.1 EXPERIMENTAL SETUP

Implementation and learning rounds. Base is XR-1 (Xiaomi Robotics Team et al., 2026) after domain post-training. Using starVLA (StarVLA Community, 2026), we freeze Base and train a rank-eight conditional adapter and a local controller per task. R1 uses Base-failure corrections; subsequent rounds target failures of the updated policy and replay earlier corrections. Each collection round adds 30 independent failure sources with three complete verified recoveries each, yielding 30/60/90 sources and 90/180/270 trajectories per task across the three-round studies.

Benchmarks and evaluation. VLABench (Zhang et al., 2025) evaluates R1 on ten task families across five tracks, using 50 held-out natural Base-failure sources per task-track pair. Three-round policy-only execution starts from task initialization on ten RoboDojo tasks (Chen et al., 2026) and three real-world tasks, with 50 and 40 trials per task and method, respectively. Training and test sources are disjoint; methods share test initializations and execution budgets within each evaluation. We report terminal task success and average rates equally across tasks and, for VLABench, tracks. Appendix A gives implementation and evaluation settings.

4.2 RECOVERY ADAPTATION FROM LIMITED CORRECTIONS

Base-Matched, Full-FT (BC/FM), LoRA-BC, and L2R-R1 resume from the same VLABench failure boundaries. Learned methods share ID-only correction sources, action targets, masks, and sampling; each task's checkpoint is fixed across all five tracks. Base-Matched samples fresh continuations with the same supplied recovery information and execution budget.

Table 1: **Recovery from natural failures on VLABench.** Conditional RecoverySR (%) on shared Base-failure boundaries. Training corrections are ID-only, with fixed R1 weights across tracks.

	ID	Distribution shifts				Overall
Method	In-dist.	Cross category	Common sense	Semantic instruction	Unseen texture	Macro ↑
Base-Matched	33.0	18.0	16.0	20.0	24.0	22.2
Full-FT (BC/FM)	42.0	25.0	19.0	23.0	34.0	28.6
LoRA-BC	43.0	28.0	23.0	29.0	36.0	31.8
L2R-R1	**69.0**	**53.0**	**33.0**	**44.0**	**61.0**	**52.0**
Δ vs. LoRA-BC (pp)	+26.0	+25.0	+10.0	+15.0	+25.0	+20.2

Each track averages ten task families; Macro weights all five tracks equally. BC/FM denotes the native action-imitation objective. Δ reports differences in percentage points; all entries are point estimates.

L2R-R1 reaches 52.0% macro RecoverySR, versus 31.8% for LoRA-BC: a 20.2-point gain from the same limited corrections (Table 1). The improvement extends to every track without track-specific retraining. Across the four shifted tracks, recovery averages 47.8% versus 29.0% for LoRA-BC. The lower commonsense score (33.0%, compared with 69.0% in ID) nevertheless shows that transfer remains sensitive to the evaluation condition.

4.3 POLICY-ONLY EXECUTION AND ITERATIVE IMPROVEMENT

We next evaluate the complete controller-adapter policy from task initialization, with recovery activation, conditions, and exit predicted locally rather than supplied at a failure boundary.

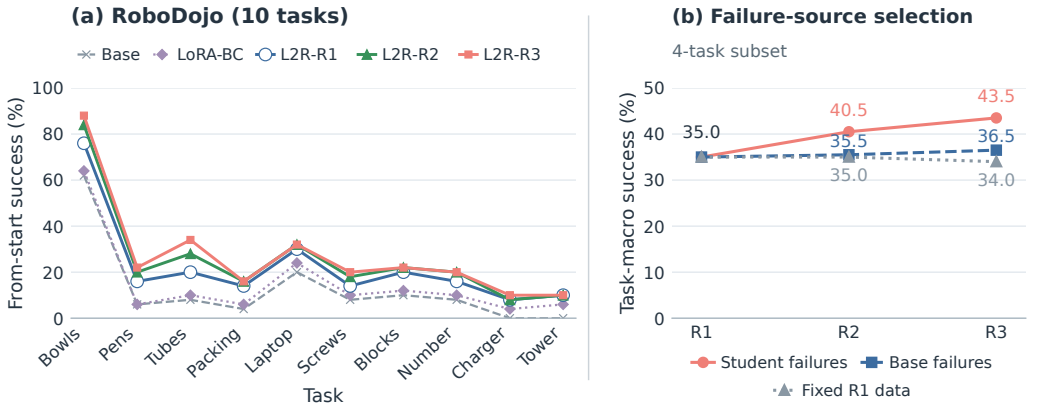

Figure 4: **L2R execution and failure-source selection.** Left: from-start success on ten RoboDojo tasks; lines connect categorical results. Right: three update strategies on a four-task subset. Appendix B gives task-wise intervals; Appendix C details the acquisition comparison.

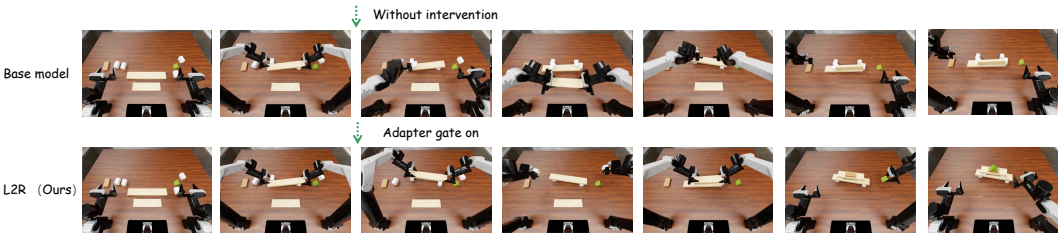

Figure 5: **Autonomous recovery in block stacking.** Top: Base continues unsuccessful stacking. Bottom: L2R-R3 activates recovery and completes the task. Frames progress from left to right.

On ten RoboDojo tasks (Figure 4, left), R1 reaches 22.4% macro success, versus 12.6% for Base and 15.2% for LoRA-BC. R2 and R3 reach 25.8% and 27.4%, adding 5.0 points beyond R1. Gains taper in later rounds, with five tasks unchanged from R2 to R3.

Acquiring failures of the updated policy. To distinguish targeted acquisition from additional optimization, we compare three branches from each task's common R1 checkpoint on a four-task subset (Figure 4, right). All continue training with matched update budgets. Two add equal numbers of accepted corrections from either the preceding student's or the fixed Base's failures; the third retains R1 data only. R3 success is 43.5%, 36.5%, and 34.0%, respectively. The 7.0- and 9.5-point advantages support collecting corrections for the updated policy's failures, beyond more optimization or equal-sized new Base-failure collections (Appendix C).

Figure 5 illustrates recovery during policy-only execution. After a stacking failure, R3 activates its recovery adapter, places the block at an angle, and completes the remaining placements in sequence. Base continues unsuccessful stacking attempts.

4.4 Real-world validation

The same correction-learning procedure improves policy-only execution on three real-world tasks (Figure 6). Macro success rises from 33.3% for Base to 51.7%, 63.3%, and 68.3% for R1-R3: 35.0 points over Base and 16.7 beyond R1. LoRA-BC shows no gain over Base in this evaluation (31.7% versus 33.3%). Every task improves beyond R1, so the later-round gain is not confined to a single task. These results extend iterative policy adaptation to physical execution. The accompanying videos illustrate failures and autonomous recovery without online agent calls or human takeover.

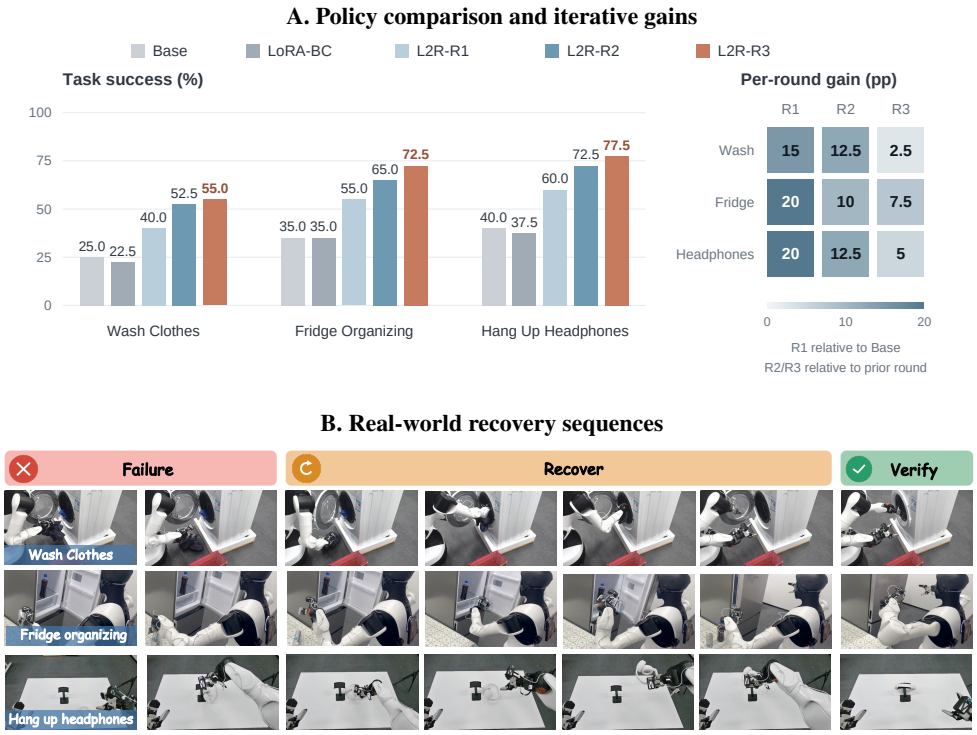

Figure 6: **L2R on real-world manipulation tasks.** A: Task success (%, left) and round-wise gains (pp, right). Gains compare R1 with Base and each subsequent round with its predecessor. B: Failures encountered and autonomously recovered from during policy-only execution.

4.5 Adaptation and Supervision Analyses

On four RoboDojo tasks, we compare adaptation and target selection using matched R1 correction sources and recovery information (Appendix D). Comparison adapters are trained with the corresponding R1 controller frozen. L2R achieves 35.0% macro from-start success, versus 25.0% for condition-aware LoRA and 9.0% when all valid actions are imitation targets. The former compares adaptation configurations under a shared recovery interface; the latter supports selecting corrective actions rather than imitating every action in a successful intervention. Failure observations can still supervise recovery invocation when their actions are excluded from imitation.

Recovery-trajectory budget. A separate three-task study fixes ten failure mechanisms and increases recovery trajectories from 20 to 80 per task (Appendix B.1). Macro RecoverySR rises from 35.6% to 40.0%, characterizing modest returns from more recordings within the same mechanism set.

5 Conclusion

L2R turns verified agent corrections into reusable recovery through lightweight, task-level adaptation of a frozen VLA. It learns corrective actions and their local invocation from distinct supervision, so deployment does not depend on online agent assistance. Beyond initial adaptation, the controlled comparisons favor collecting corrections for the updated policy's failures over additional optimization or equal-sized Base-failure collections. The resulting loop uses external problem solving to improve subsequent policy-only execution. Cross-task transfer and robustness across independent training runs remain future directions.

Statement on the Use of Generative AI

Generative AI assisted with drafting and editing, literature retrieval and citation checks, methodological and experimental feedback, and preparation of supplementary software. The authors reviewed and revised the assisted work and take responsibility for the final content.

Ethics Statement

This work studies task-level recovery learning for robotic manipulation. Physical deployment requires application-specific safety assessment and safeguards against unintended actions. Collection APIs and released artifacts should be used with appropriate permissions, privacy protection, and compliance with applicable licenses.

Reproducibility Statement

Appendix A summarizes implementation and evaluation settings; Appendix B reports task-level RoboDojo results. Appendices C and D give controlled acquisition and adaptation comparisons. The supplement provides rollout-recording code, natural-failure boundary selection, and an optional collection-time API interface, together with source-identified Stack Bowls examples and software checks. These components connect to an externally installed simulator and policy runtime. Recovery videos are included; training code, evaluation infrastructure, and model weights are not distributed.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauzá, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. RoboCat: A self-improving generalist agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

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

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. PaLM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. SAFE: Multitask failure detection for vision-language-action models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=XPyAukgsFf.

Yanjiang Guo, Tony Lee, Lucy Xiaoyang Shi, Jianyu Chen, Percy Liang, and Chelsea Finn. VLAW: Iterative co-improvement of vision-language-action policy and world model. *arXiv preprint arXiv:2602.12063*, 2026. URL https://arxiv.org/abs/2602.12063.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pp. 6023–6029. IEEE, 2019.

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

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

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

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=pISLZG7ktL.

Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. FailSafe: Reasoning and recovery from failures in vision-language-action models. *arXiv preprint arXiv:2510.01642*, 2025b. URL https://arxiv.org/abs/2510.01642.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Huihan Liu, Shivin Dass, Roberto Martín-Martín, and Yuke Zhu. Model-based runtime monitoring with interactive imitation learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4154–4161. IEEE, 2024a.

Huihan Liu, Yu Zhang, Vaarij Betala, Evan Zhang, James Liu, Crystal Ding, and Yuke Zhu. Multi-task interactive robot fleet learning with visual world models. *arXiv preprint arXiv:2410.22689*, 2024b.

Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. *The International Journal of Robotics Research*, 44(10-11):1727–1742, 2025.

Lin Liu, Zhicheng Bao, Lu Zhang, Ziying Song, Wu Yang, Shuai Tao, Wulong Liu, and Huchuan Lu. LIBERO-RECOVER: Beyond task success towards failure recovery in robotic manipulation models. *arXiv preprint arXiv:2609.05178*, 2026.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024c.

Weijia Liufu, Xiaoyu Guo, Ruiyi Chen, Jingzhi Liu, Kaidong Zhang, Xiwen Liang, Jianqi Lin, Dawei Sun, Yuze Wang, Rongtao Xu, Bingqian Lin, Bowen Yang, Tongtong Cao, Bowen Peng, Dongyu Zhang, Guangrun Wang, Min Wang, Liang Lin, and Xiaodan Liang. RePO-VLA: Recovery-driven policy optimization for vision-language-action models. *arXiv preprint arXiv:2605.09410*, 2026. URL https://arxiv.org/abs/2605.09410.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

Chuanruo Ning, Tianrui Wang, Wei-Chiu Ma, and Kuan Fang. Proxy policy steering. *arXiv preprint arXiv:2609.09148*, 2026.

Octo Model Team et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

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

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Physical Intelligence et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pp. 1–16. IEEE, 2020.

Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. SayPlan: Grounding large language models using 3D scene graphs for scalable robot task planning. *arXiv preprint arXiv:2307.06135*, 2023.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

StarVLA Community. StarVLA: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

Fuchun Sun, Wenbing Huang, Yu Luo, Tianying Ji, Huaping Liu, He Liu, and Jianwei Zhang. Robot cognitive learning by considering physical properties. *Engineering*, 47:168–179, 2025. doi: 10.1016/j.eng.2024.10.013. URL https://doi.org/10.1016/j.eng.2024.10.013.

RA Team et al. Joyai-ra 0.5: Scaling robot manipulation learning via dual action alignment. *arXiv preprint arXiv:2608.05674*, 2026.

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery RL: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6 (3):4915–4922, 2021.

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. BridgeData V2: A dataset for robot learning at scale. In *Conference on robot learning*, pp. 1723–1736. PMLR, 2023.

Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. FoundationPose: Unified 6D pose estimation and tracking of novel objects. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17868–17879. IEEE, 2024.

Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eUGoqrZ6Ea.

Xiaomi Robotics Team et al. Xiaomi-Robotics-1: Scaling vision-language-action models with over 100K hours of real-world trajectories. *arXiv preprint arXiv:2607.15330*, 2026.

Sheng Xu, Ruixing Jin, Huayi Zhou, Bo Yue, Guanren Qiao, Yunxin Tai, Yueci Deng, Kui Jia, and Guiliang Liu. From reaction to anticipation: Proactive failure recovery through agentic task graph for robotic manipulation. *arXiv preprint arXiv:2605.11951*, 2026.

Yifan Yang, Zhixiang Duan, Tianshi Xie, Fuyu Cao, Pinxi Shen, Peili Song, Chenyang Zhao, Piaopiao Jin, Guokang Sun, Shaoqing Xu, Yangwei You, and Jingtai Liu. FPC-VLA: A vision-language-action framework with a supervisor for failure prediction and correction. *Expert Systems with Applications*, 316:131742, 2026. doi: 10.1016/j.eswa.2026.131742. URL https://doi.org/10.1016/j.eswa.2026.131742.

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

Zhihao Yuan, Xu Yan, Yinghong Liao, Yao Guo, Guanbin Li, Shuguang Cui, and Zhen Li. X-Trans2Cap: Cross-modal knowledge transfer using transformer for 3D dense captioning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8553–8563. IEEE, 2022.

Zhihao Yuan, Shuyi Jiang, Chun-Mei Feng, Yaolun Zhang, Shuguang Cui, Zhen Li, and Na Zhao. Scene-R1: Video-grounded large language models for 3D scene reasoning without 3D annotations. *arXiv preprint arXiv:2506.17545*, 2025.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. VLABench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11142–11152. IEEE, 2025.

Tianle Zhang, Zhihao Yuan, Dafeng Chi, Peidong Liu, Dongwei Li, Kejun Hu, Likui Zhang, Junnan Nie, Ziming Wei, Zengjue Chen, et al. Joyai-ra 0.1: A foundation model for robotic autonomy. *arXiv preprint arXiv:2604.20100*, 2026a.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness VLA: Steering frozen VLAs into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026b.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. URL https://arxiv.org/abs/2304.13705.

Qinghua Zheng, Huan Liu, Xiaoqing Zhang, Caixia Yan, Xiangyong Cao, Tieliang Gong, Yong-Jin Liu, Bin Shi, Zhen Peng, Xiaocen Fan, Ying Cai, and Jun Liu. Machine memory intelligence: Inspired by human memory mechanisms. *Engineering*, 55:24–35, 2025. doi: 10.1016/j.eng.2025.01.012. URL https://doi.org/10.1016/j.eng.2025.01.012.

A IMPLEMENTATION AND EVALUATION PROTOCOL

Collection interfaces and supervision records. Collection records identify the decision provider and its tool permissions. API agents and deterministic structured routers are distinct providers; the released Stack Bowls examples use the latter. Those examples record read-only object poses and task progress, with grasp-lift, held-object transport, handover, regrasp, placement, and return-to-home skills. These collection-time signals support annotation and verification, not privileged inputs to the deployed controller.

At anchor t, the training-window representation is

$$\xi_t = (\mathcal{H}_t, o_t, q_t, \ell_t, z_t, A_t, M_t), \qquad A_t = (a_{t+1}, \ldots, a_{t+H}). \tag{9}$$

The local interface retains $\ell_t = \ell$; richer recovery prompts belong to separate externally routed controls. Validity, execution-role, phase, and horizon annotations define

$$M_t = M_t^{\text{valid}} \odot M_t^{\text{role}} \odot M_t^{\text{phase}} \odot M_t^{\text{horizon}}. \tag{10}$$

The role predicate selects approved recovery or continuation intervals; phase and horizon predicates stop targets before incompatible or excluded actions. Native temporal alignment, including meaningful holds, is preserved. Excluded actions receive no positive imitation loss, but their observations can still carry valid control labels. An ended recording alone does not establish a safe handoff. Sampling weights balance failure classes, sources, and annotated recovery events/phases; the mask determines target eligibility.

Initialization and round inheritance. R1 uses random controller initialization and a zero adapter residual; R2-R3 inherit the selected controller-adapter pair and normalization settings, restarting the optimizer. Route IDs persist; new routes are appended.

Architecture and interfaces. The discrete condition encoder uses seven 16-dimensional embeddings and a $112 \to 64 \to 64$ SiLU MLP. Rank-eight updates ($\alpha = 8$) modify QKV, attention output, and MLP down projections in DiT layers 18-35 (zero-based; 54 sites). The low-rank modules and discrete encoder contain 1,663,920 parameters; P_{ctx} adds $64(256 + 1) = 16,448$, excluding the controller. RoboDojo uses 25 Hz action alignment, 30-step prediction, and an at-most-five-step supervision prefix. RoboDojo and real-world nominal execution use 30-step blocks; recovery executes five steps per query. VLABench uses seven-dimensional actions, ten-step prediction, and five-step replanning.

Local controller. Mean-pooled frozen visual tokens precede recovery-prompt conditioning. A shared 128-dimensional projection per view gives 384 dimensions; projected frozen task text (64) and proprioceptive/timing features (64) complete the input. LayerNorm, Linear$(512, 256)$, and SiLU feed a single-layer, unidirectional GRU (input/hidden width 256). Independent $256 \to 128 \to \text{output}$ heads predict entry, exit, and route; dropout is 0.1 after fusion and within heads. Inputs include joint, end-effector and gripper states and changes, initial-safe-pose error, Δt, remaining budget, adapter status and duration. These observations provide progress cues without a separate progress head. Teacher route/phase, privileged state, and future outcomes are excluded.

Causal query history. Each decision re-encodes up to eight actual queries, including the current one, from zero hidden state. Length masks prevent padding updates; valid events carry actual Δt. No hidden state crosses windows and no extra VLA queries are made. The eight-query limit excludes explicit route/dwell state. Labels and local predictions share encoding C; routes persist until local exit, independently of task verification.

Compute and collection-time models. API-assisted collection uses GPT-5.6 Sol (`gpt-5.6-sol`) for VLABench and RoboDojo, and GPT-6 Astra (`gpt-6-astra`) for real-world recovery without privileged simulator state. These models provide collection-time decisions, distinct from the student action policy and executable tools. Decisions, actions, and tool outputs supply supervision; collected recoveries receive no manual edits.

Recovery-expert optimization. VLABench and RoboDojo share the settings in Table A1. Each run samples 163,840 windows, including repeats, over 5,120 updates. Full-FT and LoRA-BC share the sources and supervision of L2R, with learning rates tuned independently on ID development data under a common budget. Iterative training updates both learned modules; the adapter controls in Appendix D freeze the corresponding R1 controller.

Table A1: **Recovery-expert training configuration.** Action-supervised modules are Conditional LoRA, the condition encoder, and context projection.

Setting	Value
Rank / scale; condition width	$r = 8$, $\alpha = 8$; 64
Optimizer	DeepSpeed FusedAdam in AdamW mode (Loshchilov & Hutter, 2017)
Learning rate	Warmup $5 \times 10^{-7} \rightarrow 8 \times 10^{-6}$ over 256 updates; then cosine decay to 8×10^{-7}
Adam coefficients	$\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$
Weight decay	Weight matrices: 0.01; biases and embeddings: 0
Gradient clipping	1.0
Updates / global batch	5,120 per expert / 32 windows per update
Device layout	8 NVIDIA H200 GPUs; microbatch 4 per GPU, accumulation 1
Precision / sharding	BF16 / ZeRO-2 (Rajbhandari et al., 2020)
Checkpoint saving	Every 1,024 optimizer updates

VLABench acquisition and evaluation. Tracks 1, 2, 3, 4, and 6 test in-distribution, cross-category, commonsense, semantic instruction, and unseen texture conditions. R1 collects 30 independent ID failure sources per task, with three complete verified recoveries per source. Base failures arise under original instructions and native execution, without injected perturbations. Corrections and model selection use only ID training and development pools, respectively; these and all test pools are source-disjoint.

For each task-track pair, Base generates 50 independent test failures with one primary boundary per source. Admission and boundary selection do not depend on teacher success. Methods share these pools; Base-Matched samples fresh continuations rather than replaying failed actions. Original instructions, recovery information, restored states, remaining control budgets, action chunks, and replanning are matched. Track 3 variants `select_nth_largest_poker` and `select_unique_type_mahjong` use the `select_poker` and `select_mahjong` checkpoints. Recovery is evaluated with supplied boundaries and conditions; autonomous invocation is evaluated in the from-start studies.

Aggregation. Each task-track recovery rate is the success count divided by 50. Track scores average the ten task families; the overall score averages the five tracks equally. Task-wise summaries likewise average all five tracks. Percentage-point gains use unrounded rates.

VLABench task families. The task families are `select_painting`, `select_book`, `select_drink`, `select_chemistry_tube`, `select_poker`, `select_mahjong`, `select_toy`, `select_fruit`, `add_condiment`, and `insert_flower`.

RoboDojo and real-world evaluation. RoboDojo training and evaluation use seed 0. From-start evaluation uses 50 episodes per task and method; the trajectory-budget study uses 30 recovery evaluations per task and budget. Generalization tasks use 25 standard and 25 randomized episodes; other tasks use 50 standard episodes. We retain native full-task success criteria, including robot return, and the original episode limits throughout recovery. Methods and rounds share test layouts and execution budgets. Task success is the fraction of successful episodes; macro scores weight tasks equally. Real-world recovery-data collection and policy evaluation use an AGIBOT G2 robot. Each real-world task uses 40 from-start reset trials per method.

Demonstration videos. Real-world clips show policy-only failure recovery; RoboDojo clips illustrate simulation recovery. Selected excerpts have labeled clip boundaries and playback speeds.

Statistical reporting. RecoverySR includes every admitted failure, including teacher failures; from-start success includes every test initialization, with failures and timeouts retained. Infrastructure replacements follow a method-independent rule. RoboDojo whiskers are 95% Wilson intervals (Wilson, 1927), computed from task-wise success counts under an independent-trial binomial model. They describe trial-level uncertainty conditional on the trained checkpoint.

B TASK-LEVEL RESULTS ON THE TEN ROBODOJO TASKS

Figure 7 reports all five configurations on the selected ten RoboDojo tasks, evaluated under the protocol in Appendix A.

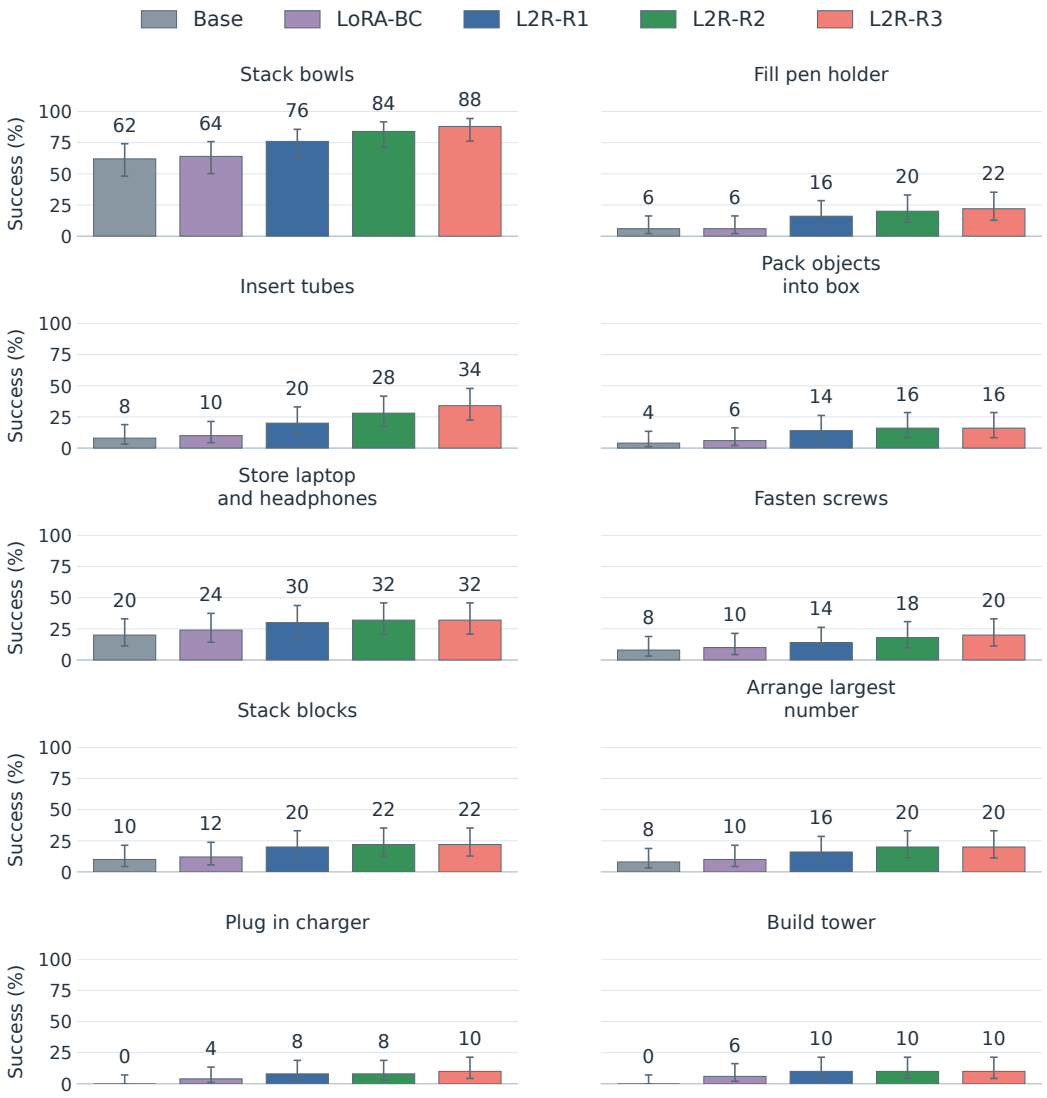

Figure 7: **Results on the ten RoboDojo tasks.** From-start autonomous success (%) on a common 0-100% scale. Colors identify methods and rounds; whiskers show 95% Wilson intervals as defined in Appendix A.

B.1 RECOVERY-TRAJECTORY BUDGET

We vary recovery trajectories while retaining ten failure mechanisms on `stack_bowls`, `insert_tubes`, and `store_laptop_and_headphones`. Training budgets contain 20, 40, 60, or 80 recovery trajectories per task. Held-out natural-failure test cases are matched across budgets, following the recovery-evaluation protocol in Appendix A.

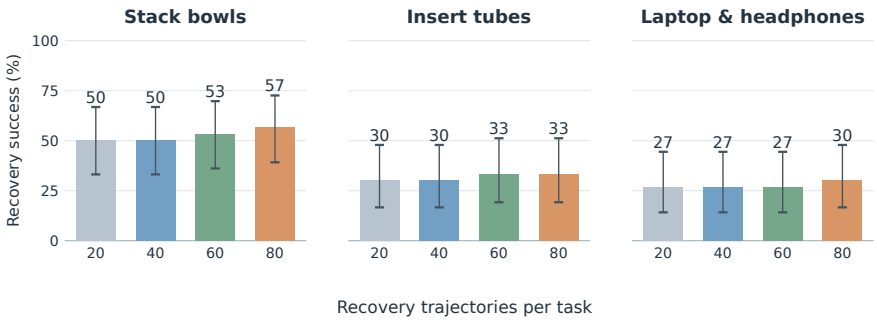

Figure 8: **L2R with additional trajectories at fixed mechanism coverage.** Conditional recovery on three tasks, with ten failure mechanisms retained across budgets. Labels round rates to whole percentages; whiskers show 95% Wilson intervals as defined in Appendix A.

Macro RecoverySR is 35.6%, 35.6%, 37.8%, and 40.0% across the four budgets (Figure 8). Quadrupling the number of trajectories adds 4.4 percentage points. Additional recordings within this fixed mechanism set thus yield modest gains.

C CONTROLLED ITERATIVE ACQUISITION

Protocol. We use Stack Bowls, Insert Tubes, Store Laptop and Headphones, and Fasten Screws from the ten-task study. From each task's common R1 checkpoint, the Student-failure branch corrects its preceding student's failures; the Base-failure branch corrects new failures of the fixed post-trained Base. Both add 30 independent sources with three complete verified recoveries per source per round, giving cumulative budgets of 30/60/90 sources and 90/180/270 trajectories. Fixed-data continuation retains the original 30 sources and 90 trajectories. All branches update their preceding checkpoint, with matched update and model-selection budgets; the acquisition branches share the replay rule. Collection budgets match accepted recoveries; teacher attempts and collection time may differ between acquisition branches.

Evaluation. Each checkpoint follows the ten-task study's policy-only evaluation protocol; macro success weights the four selected tasks equally. The common R1 and Student-failure checkpoints and evaluation records are reused from the corresponding subset of the main study.

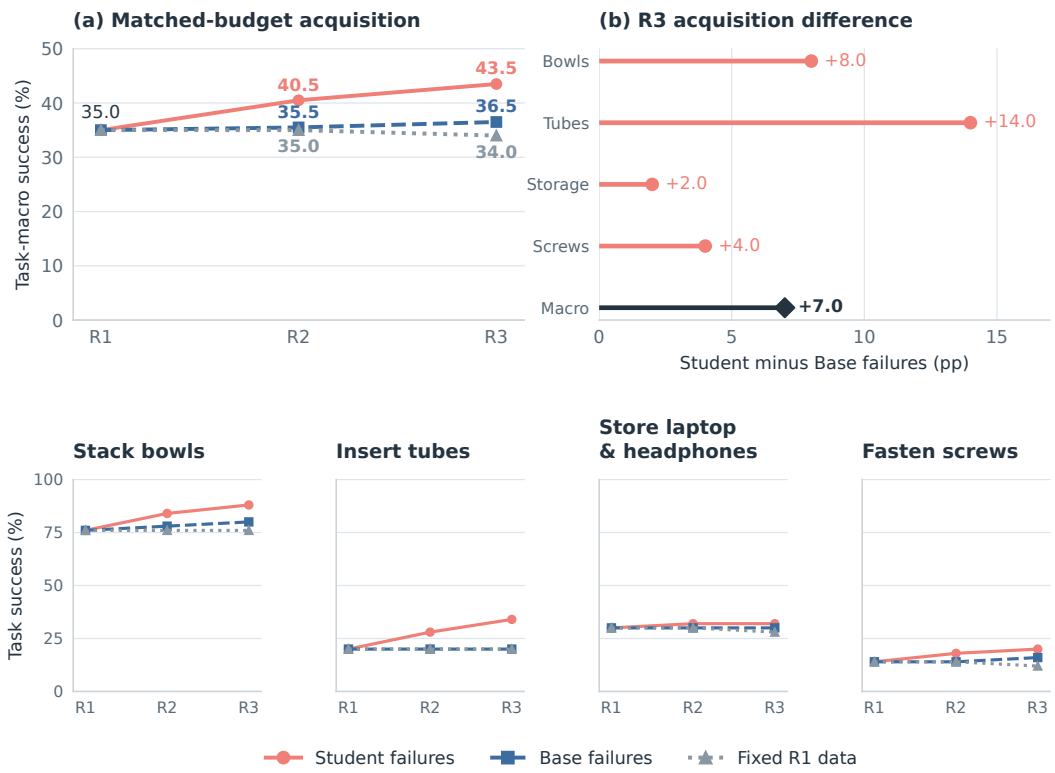

Figure 9: **Effect of the acquisition policy in L2R.** Top left: four-task macro success for branches sharing each task's R1 checkpoint. Top right: R3 differences between current-student and Base-failure acquisition. Bottom: all four task trajectories on a common 0-100% scale.

Results. From 35.0% at R1, Student-failure acquisition reaches 40.5% and 43.5% at R2 and R3, versus 35.5% and 36.5% for Base-failure acquisition and 35.0% and 34.0% for fixed-data continuation. The R3 advantages are 7.0 and 9.5 percentage points. Student-minus-Base gains are 8.0, 14.0, 2.0, and 4.0 points by task: tube insertion benefits most, while storage changes little after R1. In this configuration, current-student corrections outperform extra optimization and equal-sized new Base-failure collections.

D ADAPTATION AND INTERVENTION-SUPERVISION CONTROLS

Matched information and control. We use the four tasks and R1 correction sources from Appendix C, with matched training/tuning budgets. The L2R reference reuses the main experiment's jointly trained R1 checkpoints and evaluation records. For each task, comparison adapters are trained and evaluated with that R1 controller frozen. Both adapter architectures receive the same task instruction, mode-route conditions, and detached GRU features, using the condition-fusion rule in Section 3.3. Condition-aware standard LoRA uses a non-modulating condition input and an ordinary low-rank increment; L2R modulates its low-rank channels. Controller weights and rules are shared; switching decisions respond to observations reached by each action policy. This condition-aware control is distinct from the original LoRA-BC reference.

Selective-supervision control. *All valid actions* removes semantic target selection within the intervention records while retaining basic action-validity and padding checks; control labels are unchanged. The control retains L2R's sampling rule, recovery architecture, and evaluation interface. All checkpoints follow the ten-task study's from-start evaluation protocol.

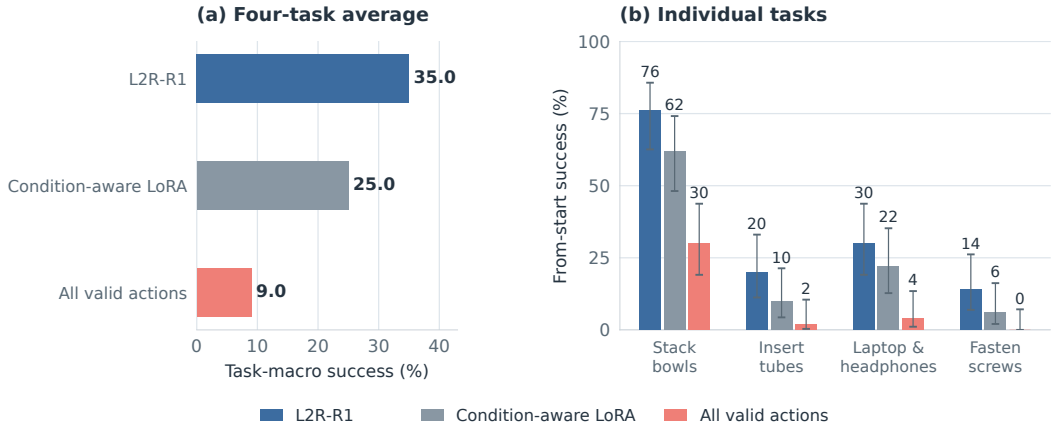

Figure 10: **Adaptation and supervision controls for L2R.** Left: equally weighted four-task macro success. Right: task-wise from-start success (%); whiskers show 95% Wilson intervals as defined in Appendix A.

Results and interpretation. L2R reaches 35.0% macro success versus 25.0% for condition-aware standard LoRA, with task-wise gains of 14.0, 10.0, 8.0, and 8.0 percentage points. The all-valid-actions control reaches 9.0%. Under the shared R1 controller, the full system outperforms the tested adapter controls. Because the full reference was jointly trained, this compares adaptation configurations rather than isolating modulation alone. The decrease without target selection is consistent with unsuccessful actions becoming imitation targets within otherwise successful intervention records; terminal success does not make every recorded action useful for imitation.