FROM EVALUATION TO RECOVERY: VALUE-GUIDED VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models learn to generate actions from demonstrations, but the imitation objective does not teach them to compare those actions. Executing a poor sample can therefore leave the robot in an unfamiliar state from which success is difficult. We introduce VaSeR, which equips an existing VLA with an action-conditioned value for selection and recovery. A value token in the policy's own backbone scores each candidate through the vocabulary head in one forward pass. It learns from demonstrations alone, without failure data: each demonstrated action is paired with a constructed negative whose pseudo-label decreases with its deviation from the demonstration. At inference, VaSeR executes the highest-valued of several sampled candidates and records its value before execution. When these values drop or stagnate, the robot physically retreats through its recent poses and resumes. On ten RLBench tasks, VaSeR raises success from 60.2% to 76.5% on $\pi_{0.5}$ and from 52.3% to 65.3% on HybridVLA; on two real-robot tasks, the corresponding improvements are from 58.3% to 73.3% and from 53.3% to 78.3%. Held-out evaluations show that constructed negatives improve both candidate ranking and episode-level alarm accuracy.

1 INTRODUCTION

Vision-language-action (VLA) models map images and language instructions to robot actions, using autoregressive, diffusion, or flow-based policies built on pretrained vision-language backbones (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025a; Liu et al., 2026). Yet imitation teaches a policy to reproduce demonstrated actions without comparing their quality: the policy generates an action and executes it (Figure 1(a), top). Diffusion and flow matching start from random noise, so samples from the same observation can lead to different outcomes. A single poor action may leave the robot in an unfamiliar state from which the policy struggles to complete the task.

A policy that evaluates its own actions can select among candidates to avoid a poor sample. When all candidates are poor and execution stops progressing, it can also recognize the need to back up. Existing evaluators include separate critics and verifiers (Nakamoto et al., 2025; Kwok et al., 2025; Xie et al., 2026) and values produced by the policy itself (Zhang et al., 2025a; Kim et al., 2026). Several train their value estimates from reward-labeled data or policy rollouts (Nakamoto et al., 2025; Physical Intelligence et al., 2025b; Kim et al., 2026); some score imagined futures rather than actions, and their main use is selection rather than recognizing when execution is failing.

We introduce VaSeR (Value-guided Selection and Recovery), which adds an action-conditioned *value token* `[V]` to the VLA backbone (Figure 2(a)). Inspired by next-token reward modeling (Zhang et al., 2025b), we decode the token through the vocabulary head over value bins after it attends to the observation and a candidate encoded by the actor's action pathway. Scoring takes one deterministic forward pass. Demonstrations supply successful actions but no comparison with inferior ones. We therefore pair each demonstrated action with a constructed negative, reduce its target value according to its deviation in action space, and train with classification and ranking losses. Both observations and targets come from demonstrations, so training requires no failure data. The shared backbone also exposes the actor to this supervision.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

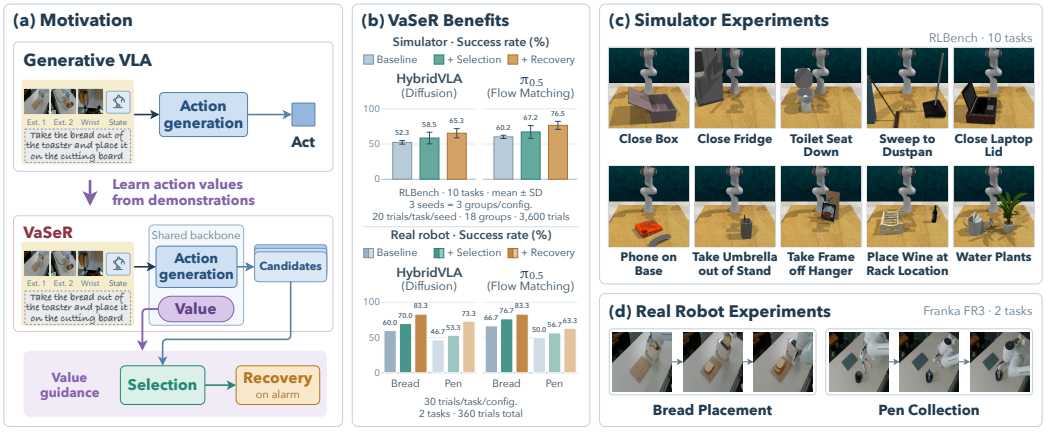


Figure 1: **Overview of VaSeR.** The proposed framework equips a generative VLA with an action-conditioned value, learned from demonstrations within the policy's own backbone, and uses this value through two mechanisms: value-guided selection and trend-aware backtracking. The former samples several candidate actions and executes the one with the highest value. The latter tracks the values of the selected actions and, when they drop or stagnate, physically retreats the robot through its recorded poses before resuming. Under this design, VaSeR improves a diffusion-based and a flow-based VLA in simulation and on a real robot. In simulation and on the real robot, +Selection includes value training; +Recovery adds backtracking to the same configuration.

VaSeR uses this value twice (Figure 1(a), bottom). Before acting, it executes the highest-valued candidate, using the comparison learned from negatives. During execution, it records each selected action's value before acting and monitors its trend. The demonstration targets rise toward completion; a drop or stagnation can therefore signal trouble. With thresholds calibrated on successful runs, an alarm withholds the selected action and triggers a physical retreat through one to three recorded poses. Retreat restores the robot's configuration, not the scene. Its premise is that a familiar pose may permit a new continuation even after an object moves, whereas an unfamiliar robot configuration may leave the policy unable to escape. This premise has limits, which we discuss in Section 5.

On ten RLBench tasks (James et al., 2020), VaSeR raises success from 60.2% to 76.5% on the flow-based $\pi_{0.5}$ (Physical Intelligence et al., 2025a) and from 52.3% to 65.3% on our diffusion-only HybridVLA adaptation (Liu et al., 2026), improving every task on both backbones (Table 1). On two real-robot tasks, the corresponding improvements are from 58.3% to 73.3% and from 53.3% to 78.3%. On held-out candidate pairs with different outcomes, the evaluators rank correctly in 91.7% and 88.3% of cases, compared with 48.3% and 46.7% without negatives. Episode-level alarm accuracy likewise falls from 93.3% and 81.7% to 51.7% and 45.0% when negatives are removed. Our contributions are:

- We train a discriminative value token inside a generative VLA using constructed negative actions and deviation-based pseudo-labels, without failure data.
- We use this value to select among sampled actions and directly measure the two factors governing selection: candidate outcome diversity and ranking accuracy.
- We use value trends to trigger physical backtracking through recorded poses, improving aggregate success on both backbones in the ten-task comparison.

2 Related Work

Vision-language-action models. VLA models adapt pretrained vision-language models into robot policies. RT-2 and OpenVLA (Brohan et al., 2023; Kim et al., 2024) predict discretized action tokens autoregressively, and FAST (Pertsch et al., 2025) compresses action chunks into shorter token sequences. Other methods keep actions continuous by denoising (Chi et al., 2023; Octo Model Team et al., 2024; Liu et al., 2024) or flow matching (Lipman et al., 2022): π_0 and $\pi_{0.5}$ (Black et al., 2024;

Physical Intelligence et al., 2025a) add a flow-matching action expert to a VLM, CogACT (Li et al., 2024) adds a diffusion action module, and HybridVLA (Liu et al., 2026) folds diffusion denoising into the language model's token sequence. These methods improve how actions are represented, generated, and fused; VaSeR adds an explicit action value to such a policy.

Value-guided robot policies. At inference, learned scores can rank candidate actions: V-GPS (Nakamoto et al., 2025) uses an offline-RL value, RoboMonkey (Kwok et al., 2025) trains a separate vision-language verifier on synthetic preferences, and other critics, scoring heads, and probes rank candidates or partially denoised actions (Xie et al., 2026; Song et al., 2026; Xu et al., 2026; Dong et al., 2026; Zhang et al., 2026). FabriMAE (Aniri et al., 2026) instead derives action reliability from internal attention entropy. During training, ReinboT (Zhang et al., 2025a) conditions actions on a predicted return-to-go, and $\pi^*_{0.6}$ (Physical Intelligence et al., 2025b) conditions the policy on advantages from a separately learned value model. Cosmos Policy (Kim et al., 2026) predicts actions, future frames, and values in one diffusion framework; its model-based planning variant scores imagined futures after fine-tuning on 648 outcome-labeled rollouts. Learned evaluators use supervision from offline demonstrations, synthetic preferences, rollout outcomes, or search returns. Among scores that need no failure data, the RoboMonkey verifier is a separate model, Implicit BC (Florence et al., 2022) trains an energy over actions from demonstrations with sampled counter-examples and obtains actions by optimizing this energy instead of sampling an explicit policy, and progress estimators learned from videos or obtained in context from vision-language models (Ma et al., 2023b;a; Yang et al., 2024; Ma et al., 2024) evaluate observations or visual trajectories rather than unexecuted candidate actions. In contrast, VaSeR keeps the generative actor and learns, from demonstrations alone, a discriminative action value that is decoded from the actor's own backbone as a classification over value bins (Zhang et al., 2025b; Farebrother et al., 2024).

Failure detection and recovery. Inner Monologue, DoReMi, and REFLECT (Huang et al., 2022; Guo et al., 2023; Liu et al., 2023) use execution feedback with language models to explain failures or revise plans, while ReKep (Huang et al., 2024) re-solves keypoint constraints and backtracks across stages. SAFE and related detectors (Gu et al., 2025; Xu et al., 2025) read failure from the policy's features or uncertainty, and Sentinel (Agia et al., 2024) monitors generative policies through the temporal consistency of their action predictions and a vision-language progress check. Recovery RL (Thananjeyan et al., 2021) switches to a recovery policy when a safety critic predicts danger, and B2FF (Shin et al., 2026) steers a VLA toward a pre-imagined milestone image, using controlled recovery-entry signals or proprioceptive triggers calibrated on successful trajectories. These approaches use language feedback, constraint checks, dedicated detectors, or a separate safety critic. VaSeR reuses the value computed for action selection to trigger physical retreat through recently recorded poses.

3 VaSeR: From Evaluation to Recovery

3.1 Overview

A VLA receives $x_t = (o_t, p_t, \ell)$, comprising an image, robot state, and instruction, and produces $a_t \in \mathbb{R}^7$: an end-effector keyframe target (translation and rotation) and gripper command, normalized to $[-1, 1]$. VaSeR uses one backbone with two outputs (Figure 2). The actor $\pi_\theta(a \mid x)$ is trained by imitation with loss $\mathcal{L}_{\text{act}}$: diffusion denoising in our HybridVLA adaptation, with no autoregressive action objective or gated fusion, and flow matching in $\pi_{0.5}$ (Appendix A). The value token $V_\theta(x, a)$ shares the backbone and action pathway. It deterministically scores each clean candidate in one forward pass, without generating an imagined future or a sampled value latent. The real-robot experiments use 10-dimensional relative actions instead of keyframes (Section 4.3).

3.2 Integrating evaluation into the generative backbone

To score a clean candidate a, the input sequence contains the observation prefix, the candidate, and an end-of-action marker, the `[V]` token in Figure 2(a). The candidate enters through the actor's action projection and pathway at the clean end of the generative process. The marker's hidden state is decoded through the vocabulary head restricted to M value bins spanning $[-1, 1]$. The scalar value is

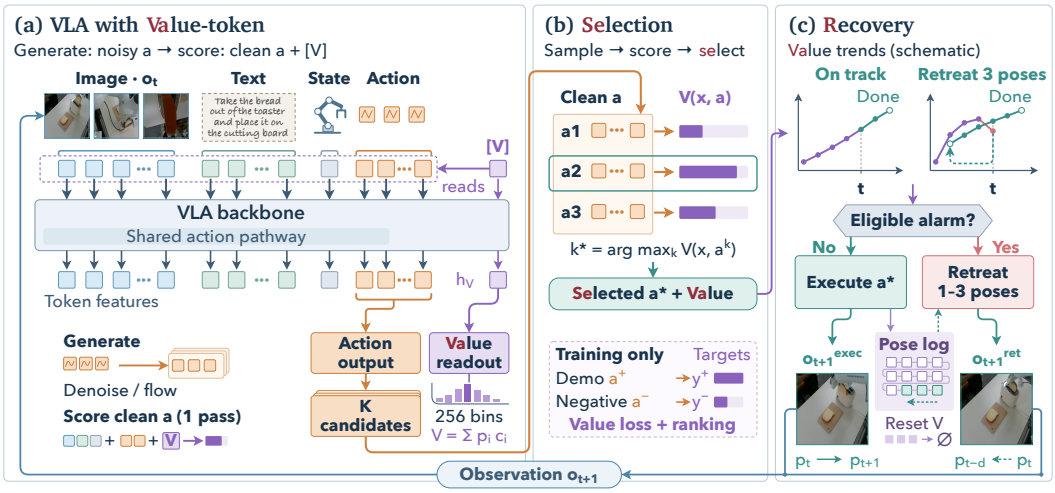


Figure 2: **VaSeR framework.** (a) The VLA samples K candidate actions with its native diffusion or flow-matching generator. The value token `[V]` then scores each clean candidate in one forward pass through the shared backbone and action pathway. (b) The highest-valued candidate a^* is selected. During training (inset), the value token learns from demonstrated actions a^+ and constructed negatives a^-. (c) When the trend of recorded values raises an alarm, the robot withholds a^*, retreats through 1–3 recorded poses, and resets the value history; otherwise, it executes a^*. Both branches continue from a new observation o_{t+1}.

their expectation,

$$V_\theta(x, a) = \sum_{j=0}^{M-1} c_j \frac{\exp z_j(x, a)}{\sum_k \exp z_k(x, a)}, \tag{1}$$

where c_j is bin j's center and $z(x, a)$ are the logits. Both generation and evaluation gradients reach the shared backbone. Causal attention lets the marker attend to the observation and candidate while masking the subsequent training-time teacher token, so the target cannot leak into the score. Scoring requires one deterministic forward pass and no second model. Appendix B gives the discretization and hyperparameters.

3.3 Collaborative Training Recipe

Value targets. Demonstrations carry no rewards. We assign a unit reward to task completion and use its discounted return, mapped to the value interval, as the target for action i:

$$y_i^+ = 2\gamma^{\tau_i} - 1, \tag{2}$$

where $\gamma \in (0, 1)$ and τ_i counts remaining steps on a fixed keyframe clock (Appendix B). The target rises toward 1 as completion approaches. Unlike a linear target based on i/N, it depends on remaining steps rather than total demonstration length. It also leaves room above the lower bound for negative labels early in a trajectory, where a linear target starting at -1 would clip them to the demonstrated target. Appendix B compares exponential and linear targets. These targets alone describe successful progress but offer no examples of inferior actions.

Negative actions. We pair each demonstrated action a^+ with a constructed negative a^- from the same observation, using one of two constructions,

$$a^- = \begin{cases} \mathrm{clip}\big(a^+ + \sigma\epsilon,\ -1,\ 1\big), & \epsilon \sim \mathcal{N}(0, I), \quad \text{perturbation negative,} \\ \mathrm{sg}\big[\pi_\theta(x; \xi)\big], & \text{policy negative,} \end{cases} \tag{3}$$

where σ is the perturbation scale. A policy negative uses a few denoising steps of the current actor, with stopped gradients and no environment execution. Both constructions receive a pseudo-label

reduced by their deviation from the demonstration:

$$y^- = \mathrm{clip}\big(y^+ - \delta - d(a^-, a^+),\, -1,\, 1\big), \qquad d(a^-, a^+) = \tfrac{1}{7}\textstyle\sum_{j=1}^{7} |a_j^- - a_j^+|. \tag{4}$$

Here $\delta > 0$ adds a fixed offset before clipping. Larger action-space deviations receive lower targets. This is a proxy: it does not guarantee failure and can penalize valid alternatives in multimodal tasks. The construction also affects shared training: perturbation negatives are not sampled from the actor, whereas policy negatives explicitly rank the actor's samples below demonstrations (Section 4.2).

Collaborative objective. Let $Q(y)$ map a target to its bin. The value token is trained with a classification loss on both actions and a ranking loss between them,

$$\mathcal{L}_V(x, a, y) = \mathrm{CE}\big(z(x, a),\, Q(y)\big), \qquad \mathcal{L}_{\mathrm{rank}} = \mathrm{softplus}\big(m + V_\theta(x, a^-) - V_\theta(x, a^+)\big), \tag{5}$$

$$\mathcal{L} = \mathcal{L}_{\mathrm{act}} + \lambda_V\, \tfrac{1}{2}\big[\mathcal{L}_V(x, a^+, y^+) + \mathcal{L}_V(x, a^-, y^-)\big] + \lambda_R\, \mathcal{L}_{\mathrm{rank}}, \tag{6}$$

where m is the ranking margin and λ_V, λ_R weight the losses. Negative generation and labeling receive no gradients, while both value losses update the actor's backbone. Section 4.2 examines this coupling by changing the negatives and detaching value gradients.

Two parts of the target. Before clipping, Eq. (4) subtracts a deviation penalty from the demonstration value. For candidates at the same observation, the demonstration term is shared, so their target differences come from the penalty; along successful trajectories, the demonstration term supplies the rising reference trend. Selection and alarms thus use complementary structure in the target. Both, however, evaluate the actor's own samples, and the ablations show that learning only from demonstrated actions is insufficient for either use.

3.4 Value-guided candidate selection

A generative actor normally executes one random sample, although different samples at the same observation may have different outcomes. VaSeR instead draws K candidates and executes the one with the highest predicted value (Figure 2(b)):

$$a_t^{(k)} \sim \pi_\theta(\cdot \mid x_t), \quad k = 1, \ldots, K, \qquad a_t = a_t^{(k^*)}, \quad k^* = \arg\max_k V_\theta\big(x_t, a_t^{(k)}\big). \tag{7}$$

No action is regenerated after selection; $K = 1$ without scoring recovers the actor alone. Generation and scoring cost scale with K. For $K = 2$, let ρ be the probability that candidate outcomes differ and p_{rank} the probability of ranking the successful candidate higher on such pairs. Relative to randomly selecting one candidate, the single-decision success gain is $\rho(p_{\mathrm{rank}} - \frac{1}{2})$ under a common continuation policy (Appendix C). Selection therefore needs both outcome diversity and better-than-random ranking; we measure both in Section 4.2.

3.5 Trend-aware backtracking

Selection can avoid a poor sample but does not change the actor's distribution at the current observation. It cannot help when all candidates are poor or undo an earlier deviation. VaSeR therefore records the selected action's value before execution, $v_t = V_\theta(x_t, a_t)$, and treats a deteriorating trend as a signal to stop and retreat (Figure 2(c)).

Trend features and alarms. We summarize the history $v_{1:t}$ with two features:

$$D_t = \left[\max_{s<t} v_s - v_t\right]_+, \qquad S_t = \max_{w \in \{2,3\},\, w \le t} \left[\mu_{w,t} - (v_t - v_{t-w})\right]_+. \tag{8}$$

Drawdown D_t measures the drop from the running maximum; stagnation S_t measures the shortfall from $\mu_{w,t}$, the median w-step gain of successful trajectories at the same stage. Per-task thresholds η_D, η_S are calibrated so at most a fraction $\alpha/2$ of successful development runs ever exceed each, where α is the successful-run alarm budget. Either feature strictly exceeding its threshold triggers an alarm. Calibration requires no failure data (Appendix D).

Physical retreat. If an eligible alarm fires, the selected forward action is withheld and the robot retreats instead. The alarm's severity, $u_t = \max_{f \in \{D,S\}:\, f_t > \eta_f} (f_t - \eta_f)/s_f$ with a calibrated scale s_f, determines the depth

$$d_t = \min\big(h_t,\ B_t - 1,\ 1 + \mathbf{1}[u_t \geq 1] + \mathbf{1}[u_t \geq 2]\big) \in \{1, 2, 3\}, \tag{9}$$

subject to recorded history h_t and remaining command budget B_t. Thus the requested depth is one pose for $u_t < 1$, two for $1 \leq u_t < 2$, and three for $u_t \geq 2$. This is the rule used in the reported adaptive-recovery experiments. The controller records the measured pose before each completed forward command and retreats through the last d_t poses in reverse, holding the current gripper command. It then takes a new observation, clears the value history, and returns control to the actor. Forward and retreat commands share budget B; no environment state is restored. Algorithm 1 in Appendix D.1 gives the complete loop.

What retreat restores. Retreat moves the robot, not the world: a displaced object stays where it is. Our hypothesis is that a falling value can reflect an unfamiliar robot configuration from which the policy cannot find a useful continuation, as in compounding errors in imitation learning (Ross et al., 2011). Demonstrations vary object placements but visit robot configurations along their own paths. Retreat may therefore restore a familiar pose while leaving moderate scene changes to policy generalization. It can fail when scene changes exceed the demonstrated variation or the retreat overshoots a recoverable configuration; the present experiments do not isolate these cases.

4 Experiments

We evaluate VaSeR on ten simulated RLBench tasks (Section 4.1), analyze its components (Section 4.2), and evaluate two real-robot tasks (Section 4.3).

Setup. The simulation benchmark comprises ten RLBench tasks (James et al., 2020) (Figure 1(c)), with 100 successful demonstrations per task. The policy observes a front RGB image, robot state, and instruction, and outputs one keyframe action per decision. Each main-comparison configuration is evaluated over 600 episodes: three seed groups, with 20 episodes per task and group. Task-success ablations use the first four tasks, with 60 episodes per task and 240 per row. Evaluator quality uses separate pair and alarm metrics, whose denominators are stated in Table 2. Earlier studies of models trained only on four tasks are reported separately in the appendix and compared only within their own protocols.

We use our 7B HybridVLA adaptation and $\pi_{0.5}$ (Appendix A). HybridVLA uses only its diffusion branch throughout: neither autoregressive action training nor gated fusion is used. Its baseline is trained with the diffusion objective alone, and value supervision is added to the same branch. $\pi_{0.5}$ uses its native flow-matching action expert. Training variants within each backbone share pretrained parameters, data, training seeds, and update counts. Further details are in Appendices B, C, and F.

4.1 Simulation experiments

Table 1 compares three cumulative configurations: a baseline without value training, value training with selection among two candidates ($K = 2$), and the addition of trend-aware backtracking (complete VaSeR). Forward and retreat commands consume the same command budget.

For $\pi_{0.5}$, value training with selection raises success from 361/600 (60.2%) to 403/600 (67.2%), and recovery raises it to 459/600 (76.5%). HybridVLA improves from 314/600 (52.3%) to 351/600 (58.5%) and then 392/600 (65.3%). Complete VaSeR exceeds the baselines by 16.3 and 13.0 percentage points and improves all ten tasks on both backbones. The intermediate configuration is less uniform: it falls below baseline on Phone on Base and Frame off Hanger for both backbones, and on Wine at Rack for $\pi_{0.5}$. Adding recovery improves nine tasks in each backbone block and leaves one unchanged. These comparisons describe aggregate counts. Table 13 separates training, selection, and recovery on the first four tasks.

Table 1: Overall success on ten RLBench tasks. Rows add value training and selection ($K = 2$), then recovery. Cells are success rates (%): three seed groups, 20 episodes per task and group (60 per task, 600 per row).

Method	Base	Close box	Close fridge	Toilet seat down	Sweep to dustpan	Close laptop lid	Phone on base	Umbrella out	Frame off hanger	Wine at rack	Water plants	Mean ± SD
$\pi_{0.5}$	Flow matching	6.7	85.0	98.3	36.7	85.0	66.7	73.3	55.0	68.3	26.7	60.2 ± 2.3
+ Value & selection		46.7	96.7	**100.0**	45.0	86.7	63.3	76.7	43.3	63.3	50.0	67.2 ± 9.0
+ Recovery		**66.7**	**98.3**	**100.0**	**60.0**	**93.3**	**68.3**	**90.0**	**58.3**	**70.0**	**60.0**	$\mathbf{76.5 \pm 5.7}$
HybridVLA	Diffusion	70.0	70.0	86.7	20.0	46.7	53.3	43.3	66.7	53.3	13.3	52.3 ± 2.5
+ Value & selection		83.3	86.7	88.3	23.3	50.0	50.0	63.3	63.3	**56.7**	20.0	58.5 ± 8.2
+ Recovery		**90.0**	**90.0**	**95.0**	**28.3**	**58.3**	**60.0**	**68.3**	**78.3**	**56.7**	**28.3**	$\mathbf{65.3 \pm 6.5}$

Mean ± SD summarizes the three seed-group ten-task rates (sample SD). Bold marks each backbone's column maximum, including ties. Both baselines omit value training; HybridVLA uses diffusion only, without an autoregressive objective or gating. The last two rows share their value-trained actor and selection settings; recovery uses the depth rule actually evaluated in the main experiment (Appendix D).

Table 2: Evaluator quality on the first four tasks. Pair accuracy uses 60 outcome-discordant pairs per backbone (chance: 50%); alarm accuracy uses 60 held-out episodes per backbone. Cells give percentages (correct/total). The three evaluators share the corresponding samples; sampling and calibration are specified in Appendix F.

	$\pi_{0.5}$		HybridVLA	
Evaluator	Pair acc.	Alarm acc.	Pair acc.	Alarm acc.
No negatives	48.3 (29/60)	51.7 (31/60)	46.7 (28/60)	45.0 (27/60)
Detached	63.3 (38/60)	71.7 (43/60)	76.7 (46/60)	65.0 (39/60)
VaSeR	91.7 (55/60)	93.3 (56/60)	88.3 (53/60)	81.7 (49/60)
Scored / sampled (ρ)	45.5 (60/132)	–	48.4 (60/124)	–

An alarm decision is correct for an alarmed failure or an unalarmed success. ρ is the observed scored-pair fraction in the sampled pool, not an additional accuracy metric.

4.2 Ablation studies

The quantitative ablations use the first four tasks; qualitative executions are shown separately. Rows matching the baseline, selection, or complete-method configurations reuse their corresponding counts from Table 1; the remaining rows are evaluated separately.

Value learning. We measure the evaluator in two ways. *Pair accuracy* compares two candidates sampled at the same decision and executed from the same recorded simulator state, with single-candidate continuation. Only pairs with one successful and one failed outcome are scored; ranking the successful candidate higher is correct, and chance is 50%. *Alarm accuracy* counts an alarmed failed episode or an unalarmed successful episode as correct. Each backbone has 60 scored pairs and 60 held-out alarm-evaluation episodes. The three evaluators score the same candidate pairs and the same alarm episodes, collected from the VaSeR actor. Sampling and calibration are specified in Appendix F.

Table 2 compares the complete value objective with two training variants. *No negatives* retains demonstration-value targets but removes negative-action supervision and the ranking loss. *Detached* uses the full value objective while stopping its gradients at the backbone. VaSeR ranks 55/60 pairs correctly on $\pi_{0.5}$ and 53/60 on HybridVLA, versus 29/60 and 28/60 without negatives. The detached evaluator ranks 38/60 and 46/60 correctly. The same ordering holds for alarm accuracy: VaSeR reaches 93.3% and 81.7%, detached reaches 71.7% and 65.0%, and no negatives reaches 51.7% and 45.0%. The scored pairs come from sampled pools of 132 and 124 pairs, with observed outcome-discordant fractions of 45.5% and 48.4%. In these samples, negative supervision improves both ranking and alarm decisions, and the shared-backbone evaluator outperforms the detached evaluator. These are descriptive comparisons; joint pair-correctness records were not retained for a paired significance test.

Table 3: Actor training on the first four tasks. One candidate, no value readout or recovery; 60 episodes per task and 240 per row. Entries are successes/episodes, with overall success rate (%) in parentheses. Baselines are the corresponding rows of Table 1.

Training variant	Box	Fridge	Toilet	Sweep	Overall
$\pi_{0.5}$					
Baseline (no value training)	4/60	51/60	59/60	22/60	136/240 (56.7)
Value training ($K = 1$)	6/60	53/60	55/60	21/60	135/240 (56.3)
No negatives	6/60	48/60	56/60	17/60	127/240 (52.9)
Detached	2/60	39/60	43/60	15/60	99/240 (41.3)
HybridVLA					
Baseline (no value training)	42/60	42/60	52/60	12/60	148/240 (61.7)
Value training ($K = 1$)	40/60	39/60	53/60	15/60	147/240 (61.3)
No negatives	33/60	35/60	54/60	13/60	135/240 (56.3)
Detached	28/60	36/60	47/60	14/60	125/240 (52.1)

Table 3 evaluates actors with one candidate and no value readout or recovery, isolating training effects. Value training gives 135/240 successes on $\pi_{0.5}$ and 147/240 on HybridVLA, one fewer than each baseline (136/240 and 148/240). Removing negatives gives 127/240 and 135/240; detaching value gradients gives 99/240 and 125/240. These results do not establish an increase in single-candidate success from value training. Earlier four-task-model studies also show that its effect depends on negative construction and training seed (Appendix B); we freeze the trained checkpoints for the inference comparisons below.

The demonstration-value target also affects the final policy. With the same selection configuration ($K = 2$, no recovery), replacing the exponential target with a linear target rising from -1 to 1 gives 121/240 successes on $\pi_{0.5}$ and 155/240 on HybridVLA, compared with 173/240 and 169/240 for the exponential target (Table 5). The aggregate differences are 21.7 and 5.8 percentage points; HybridVLA's Sweep result is one success lower with the exponential target, so the advantage is not uniform across tasks.

Candidate selection. Using the same value-trained actors, selection among two candidates changes four-task success from 135/240 to 173/240 for $\pi_{0.5}$ and from 147/240 to 169/240 for HybridVLA (Figure 3). HybridVLA loses one success on Sweep and is unchanged on Toilet; all other task counts increase. Table 9 varies K and reports measured decision latency. Among the tested settings, $\pi_{0.5}$ has the most successes at $K = 2$ (173/240), and HybridVLA at $K = 4$ (175/240). At $K = 8$, success falls to 149/240 and 145/240 while latency increases to 782.33 and 959.21 ms. More candidates therefore do not monotonically improve these measured outcomes. The opportunity for improvement depends on candidate-outcome disagreement and ranking accuracy (Section 3.4). These results do not establish superiority over unmeasured alternatives for spending the same compute budget.

Trend-aware backtracking. Adding recovery to selection raises four-task success from 173/240 to 195/240 for $\pi_{0.5}$ and from 169/240 to 182/240 for HybridVLA; Figure 4 shows the corresponding ten-task changes. Table 11 compares fixed retreat depths with the adaptive rule evaluated in the main experiment. Fixed depths of one, two, and three yield 170, 167, and 163 successes for $\pi_{0.5}$, and 169, 180, and 173 for HybridVLA, each out of 240. The largest fixed-depth count occurs at different depths for the two backbones. The adaptive counts are 195/240 and 182/240. Adaptive retreat follows Eq. (9), with severity boundaries at one and two calibration scales (Appendix D).

The component comparison further separates recovery from candidate selection (Table 13). With a single candidate, value training plus recovery gives 162/240 successes on $\pi_{0.5}$ and 167/240 on HybridVLA, compared with 135/240 and 147/240 without recovery. The complete method reaches 195/240 and 182/240, exceeding either inference component alone in these pooled evaluations. Thus the learned value remains useful for monitoring when there is no candidate choice, while combining selection and recovery gives the highest aggregate count for each backbone.

Figure 5 illustrates three successful fixed-depth executions: stagnation followed by a one-step retreat, a small drawdown followed by two steps, and a larger drawdown followed by three steps. They show

Table 4: Real-robot success. Both backbones use the same three cumulative configurations as in simulation: baseline without value training, added value training and selection, then recovery. HybridVLA uses diffusion only. Each task has 30 trials; overall rates (%) pool 60 trials.

Configuration	Bread placement	Pen collection	Overall
$\pi_{0.5}$			
Baseline	20/30	15/30	35/60 (58.3)
+ Value & selection	23/30	17/30	40/60 (66.7)
+ Recovery	25/30	19/30	44/60 (73.3)
HybridVLA			
Baseline	18/30	14/30	32/60 (53.3)
+ Value & selection	21/30	16/30	37/60 (61.7)
+ Recovery	25/30	22/30	47/60 (78.3)

recorded physical retreat and subsequent completion, rather than an aggregate evaluation of online depth selection. Earlier depth and repeated-execution studies are reported separately in Appendices D and E.

4.3 Real-world experiments

We evaluate both backbones on two tasks with a Franka Research 3 arm and parallel-jaw gripper, observed by two external RealSense cameras (D435i and D435) and a wrist-mounted D405 (Figure 1(d)). In Bread Placement, the robot moves two slices from a toaster to a cutting board; in Pen Collection, it places two tabletop pens in a holder. Both objects must be placed for a trial to succeed. Training uses 300 and 500 demonstrations, respectively, downsampled to 10 Hz, with relative end-effector actions instead of keyframes (Appendix G).

Each task, backbone, and configuration has 30 trials (Table 4). The baseline omits value training, the second configuration adds value training and selection, and the third adds recovery, as in simulation. Success counts increase at each stage for both tasks and backbones: value training with selection adds two or three successes per task, and recovery adds two to six. The largest recovery increase is HybridVLA on Pen Collection, from 16/30 to 22/30. Across both tasks, complete VaSeR raises success from 58.3% to 73.3% on $\pi_{0.5}$ and from 53.3% to 78.3% on HybridVLA. These are descriptive comparisons of aggregate counts.

5 Conclusion and Limitations

VaSeR trains a discriminative value token inside a generative VLA from demonstrations and constructed negatives, then uses it for candidate selection and physical backtracking. It improves both backbones on all ten RLBench tasks and on two real-robot tasks. Ablations show that negatives matter for both ranking and alarms, that the effect of value training on the actor depends on the construction and training seed, and that selection depends on candidate outcome diversity and ranking accuracy.

The value is a learned proxy, not a calibrated success probability. Its deviation penalty weights action dimensions equally and can penalize valid alternatives; it also does not distinguish robot displacement from scene displacement. The ten-task comparison reports aggregate counts over three seed groups, while component controls cover four tasks. Earlier training-recipe studies use four-task models, with updates rather than compute matched across variants. The evaluator comparisons are descriptive, without paired significance tests, and real-robot evidence covers only two tasks with 30 trials per condition. We have not isolated recovery outcomes by scene displacement. Future work could weight action deviations by the variability of each dimension in demonstrations and gate retreat on whether the current scene remains within the policy's demonstrated variation.

Reproducibility statement

Section 3 and Appendices A–D describe the architecture, training settings, selection controls, and recovery procedure; Appendices F–G describe evaluation protocols. Supplementary CSV files provide the recorded counts: `benchmark_seeds.csv` contains the three evaluation-group counts underlying Table 1; `simulation_successes.csv`, `real_robot_successes.csv`, and `evaluator_summary.csv` contain current task and evaluator aggregates. `table_index.csv` maps every result table to its data files, including the separate historical studies. The accompanying `README.txt` defines units, denominators, reused results, and missing fields. These files support recomputing the reported aggregates; unrecorded per-episode outcomes and paired evaluator significance tests are not reconstructed.

Statement on the use of AI

AI tools were used to aid or polish writing during manuscript preparation, including translation, drafting and language editing, literature lookup, feedback on methodological descriptions and result interpretations, consistency checks against author-provided experimental records, and assistance with table-generation scripts and figure layout. Reported measurements come from recorded experiments or author-provided counts. The authors remain responsible for the final text, methods, results, and interpretations.

References

Christopher Agia, Rohan Sinha, Jingyun Yang, Zi-ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *Conference on Robot Learning (CoRL)*, 2024. URL https://arxiv.org/abs/2410.04640.

Aniri, Chen Yilin, Jinhe Bi, Junfei Guo, Donglai Ran, Xu Bian, Zengjie Jin, Yujun Wang, Yijun Tian, Volker Tresp, Fei Shen, Tat-Seng Chua, and Yunpu Ma. FabriMAE I trust myself? self-evaluating VLA action generation with Markov attention entropy. *arXiv preprint arXiv:2608.16697*, 2026. URL https://arxiv.org/abs/2608.16697.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

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. *arXiv preprint arXiv:2303.04137*, 2023. URL https://arxiv.org/abs/2303.04137.

Perry Dong, Alexander Swerdlow, Dorsa Sadigh, and Chelsea Finn. FASTER: Value-guided sampling for fast RL. *arXiv preprint arXiv:2604.19730*, 2026. URL https://arxiv.org/abs/2604.19730.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep RL. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2403.03950.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 158–168. PMLR, 2022. URL https://proceedings.mlr.press/v164/florence22a.html.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. SAFE: Multitask failure detection for vision-language-action models. *arXiv preprint arXiv:2506.09937*, 2025. URL https://arxiv.org/abs/2506.09937.

Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu Chen. DoReMi: Grounding language model by detecting and recovering from plan-execution misalignment. *arXiv preprint arXiv:2307.00329*, 2023. URL https://arxiv.org/abs/2307.00329.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. URL https://arxiv.org/abs/2207.05608.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024. URL https://arxiv.org/abs/2409.01652.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020. URL https://arxiv.org/abs/1909.12271.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2601.16163.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. RoboMonkey: Scaling test-time sampling and verification for vision-language-action models. *arXiv preprint arXiv:2506.17811*, 2025. URL https://arxiv.org/abs/2506.17811.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024. URL https://arxiv.org/abs/2411.19650.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. URL https://arxiv.org/abs/2210.02747.

Jiaming Liu, Hao Chen, Zhuoyang Liu, Pengju An, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, Chengkai Hou, Mengdi Zhao, Kaichen Zhou, Pheng-Ann Heng, and Shanghang Zhang. HybridVLA: Collaborative diffusion and autoregression in a unified vision-language-action model. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=H1KDMNOKQn.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: A diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024. URL https://arxiv.org/abs/2410.07864.

Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing robot experiences for failure explanation and correction. In *Conference on Robot Learning*, 2023. URL https://arxiv.org/abs/2306.15724.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. LIV: Language-image representations and rewards for robotic control. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23301–23320. PMLR, 2023a. URL https://proceedings.mlr.press/v202/ma23b.html.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations (ICLR)*, 2023b. URL https://arxiv.org/abs/2210.00030.

Yecheng Jason Ma, Joey Hejna, Ayzaan Wahid, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Jonathan Tompson, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. *arXiv preprint arXiv:2411.04549*, 2024. URL https://arxiv.org/abs/2411.04549.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4996–5013. PMLR, 2025. URL https://proceedings.mlr.press/v270/nakamoto25a.html.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090. URL https://www.roboticsproceedings.org/rss20/p090.pdf.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. URL https://arxiv.org/abs/2501.09747.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025a. URL https://arxiv.org/abs/2504.16054.

Physical Intelligence et al. $\pi^*_{0.6}$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025b. URL https://arxiv.org/abs/2511.14759.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. URL https://arxiv.org/abs/1011.0686.

Suyeon Shin, Juwon Kim, Hyeonbin Park, Hyunseo Kim, Hyundo Lee, Hyung-Sin Kim, and Byoung-Tak Zhang. Back to the familiar future: Failure recovery for VLA policies via pre-imagined milestone selection. *arXiv preprint arXiv:2606.09258*, 2026. URL https://arxiv.org/abs/2606.09258.

Haoming Song, Delin Qu, Yuanqi Yao, Qizhi Chen, Jiarui Li, Qi Lv, Yiwen Tang, Li Kang, Heng Zhou, Xianqiang Gao, Yuhang Tang, Xiaofan Li, Modi Shi, Guanghui Ren, Maoqing Yao, Bin Zhao, Dong Wang, and Xuelong Li. FM-Steer: Enhance generalist policies with value-guided cascaded denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13407–13418, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Song_FM-Steer_Enhance_Generalist_Policies_with_Value-Guided_Cascaded_Denoising_CVPR_2026_paper.html.

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*, 2021. URL https://arxiv.org/abs/2010.15920.

Xinyi Xie, Zican Hu, Zhanyu Liu, Yicheng Dong, Wenhao Wu, Zhenhong Sun, Haoran Li, Chunlin Chen, Zhi Wang, and Pichao Wang. Look before you leap: Distilling tree search into action evaluation for frozen VLA models. *arXiv preprint arXiv:2607.03751*, 2026. URL https://arxiv.org/abs/2607.03751.

Changhua Xu, En Yu, Junyu Xuan, and Jie Lu. VGAS: Value-guided action-chunk selection for few-shot vision-language-action adaptation. *arXiv preprint arXiv:2602.07399*, 2026. URL https://arxiv.org/abs/2602.07399.

Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Ambrus, Haruki Nishimura, and Masha Itkina. Can we detect failures without failure data? Uncertainty-aware runtime failure detection for imitation learning policies. *arXiv preprint arXiv:2503.08558*, 2025. URL https://arxiv.org/abs/2503.08558.

Daniel Yang, Davin Tjia, Jacob Berg, Dima Damen, Pulkit Agrawal, and Abhishek Gupta. Rank2Reward: Learning shaped reward functions from passive video. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. URL https://arxiv.org/abs/2404.14735.

Hongyin Zhang, Zifeng Zhuang, Han Zhao, Pengxiang Ding, Hongchao Lu, and Donglin Wang. ReinboT: Amplifying robot visual-language manipulation with reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 77254–77271. PMLR, 2025a. URL https://proceedings.mlr.press/v267/zhang25dr.html.

Jiachen Zhang, Junnan Nie, Junyi Lao, Wei Cheng, Chenghao Liu, Jiaxin Jiang, and Songfang Huang. What frozen VLAs already know about success: A probing study of value-like structure in foundation robot policies. *arXiv preprint arXiv:2605.28527*, 2026. URL https://arxiv.org/abs/2605.28527.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations (ICLR)*, 2025b. URL https://arxiv.org/abs/2408.15240.

A BACKBONE INTEGRATION

$\pi_{0.5}$. The backbone contains a frozen SigLIP image encoder, a language expert for the observation prefix, and a smaller action expert that exchanges information with it through attention. The native flow-matching actor receives a noisy action $a_s = s\epsilon + (1-s)a^+$ and predicts the velocity $\epsilon - a^+$, with flow time entering through adaptive normalization. The value token reuses the action input projection and action expert at $s = 0$. A clean simulation candidate, with seven real dimensions padded to the native 32-dimensional representation, is followed by an end-of-action marker. A width adapter maps its hidden state to the tied vocabulary head. The last $M = 256$ vocabulary entries encode value bins; the training classification uses the full vocabulary, while the scalar readout normalizes over the value bins. A teacher token carrying the target follows the marker during training and predicts the end-of-sequence token. The causal mask hides this teacher from the marker, and inference omits it. The vision encoder is frozen; the remaining language and action pathways, adapters, and embeddings are trained. Policy negatives are generated with four Euler steps under a stopped gradient.

HybridVLA. Frozen DINOv2 and SigLIP encoders process the front image; their concatenated patch features are projected into a LLaMA-2 7B model. The robot state, diffusion timestep, and noisy action enter as additional tokens, and the action-position head predicts diffusion noise. Our experiments use the diffusion branch alone, without autoregressive action training or gated fusion. The value-trained variant uses the language head for value and end-of-sequence targets. Each example has a noisy diffusion pass and a clean pass scoring the demonstrated action and a constructed negative through the shared transformer. Policy negatives use four DDIM steps. The action and value losses have separate output heads but share the backbone.

B TRAINING AND STABILITY

Targets and discretization. The simulation progress target uses $\gamma = 0.955$ and the keyframe clock $\tau_i = 4(N - i) - 1$ for demonstration action $i \in \{0, \dots, N-1\}$ in a trajectory of length N; the terminal post-action target is 1. The value representation uses $M = 256$ bins. Ordered edges are $e_j = -1 + 2j/255$, $j = 0, \dots, 255$, and the target bin is $Q(y) = \mathrm{clip}(\mathrm{searchsorted}(e, y) - 1, 0, 255)$ with right-continuous search. Readout centers are $c_j = (e_j + e_{j+1})/2$ for $j < 255$ and $c_{255} = c_{254}$. Negative construction uses $\sigma = 0.2$, $\delta = 0.1$, and ranking margin $m = 0.05$; the loss weights are $\lambda_V = 1$ and $\lambda_R = 0.1$.

Shape of the value target. Table 5 compares the exponential target with a linear target increasing uniformly from -1 at the first demonstrated action to 1 at the last. The negative-target construction is unchanged. Both variants use the training setting of the main comparison and $K = 2$ selection without retreat on the first four tasks. The exponential rows are the corresponding entries of Table 1; the linear rows are separate evaluations.

Detached evaluator. The detached control uses the same value objective but stops its gradient at the backbone. Only value-specific parameters, including the marker embedding, width adapter, and value-bin vocabulary rows, receive the value losses; the backbone receives the action loss alone. Table 2 compares evaluator quality on the same sampled candidates and held-out executions. This comparison does not by itself measure the actor performance of the detached model.

Separate four-task training studies. The remaining training, seed, and sampler studies in Appendices B–D use earlier models trained only on Close Box, Close Fridge, Toilet Seat Down, and Sweep to Dustpan. They are distinct from the ten-task checkpoints of Table 1, including when both studies evaluate the same four tasks. Table 6 documents these earlier recipes; their performance should be compared within the indicated cohort.

Policy negatives help HybridVLA in this cohort but reduce the single-candidate success of $\pi_{0.5}$ (Table 7). Removing their ranking term or reducing the auxiliary loss weights recovers much of the decrease. This sensitivity motivates checking both the actor and evaluator when adding shared value supervision.

Table 5: Exponential versus linear value targets with selection ($K = 2$), without recovery. Training follows the main comparison except for the demonstration-value target; negative targets retain Eq. (4). Cells are successes/60; overall rates (%) use 240 episodes.

Backbone	Target	Box	Fridge	Toilet	Sweep	Overall
$\pi_{0.5}$	Exponential	28/60	58/60	60/60	27/60	173/240 (72.1)
$\pi_{0.5}$	Linear	12/60	32/60	53/60	24/60	121/240 (50.4)
HybridVLA	Exponential	50/60	52/60	53/60	14/60	169/240 (70.4)
HybridVLA	Linear	48/60	45/60	47/60	15/60	155/240 (64.6)

The exponential target is $2\gamma^{\tau} - 1$ and reuses Table 1's selection row. The linear target interpolates uniformly from -1 at the first demonstration action to 1 at the last action.

Table 6: Settings of the earlier four-task studies and training-seed replication. These settings do not supply missing metadata for the ten-task checkpoints. A dash retains the backbone's original recipe.

Setting	HybridVLA	$\pi_{0.5}$
Initialization	HybridVLA pretrained	$\pi_{0.5}$ base
Training seed	42	42; 43, 44 for replication
Global batch size / updates	64 / 7,200	64 / 7,200
Negatives / noise repeats per example	1 / 4	1 / 4
Policy-negative sampler	DDIM 4	Euler 4
Optimizer	AdamW	AdamW
Learning rate / weight decay	– / –	2×10^{-5} / 0
AdamW β_2	–	0.95; 0.999 for replication
Frozen modules	Vision encoders	Vision encoder
Training-ablation inference	DDIM 8	Euler 8
Candidate-sweep inference	DDIM 4	Euler 8
Candidate-sweep decision budget	10	10

Optimizer cohort and training seeds. With $\beta_2 = 0.999$ and the other four-task settings unchanged, $\pi_{0.5}$ obtains 182/240 with flow only, 184/240 with demonstration progress targets, 150/240 with policy negatives and ranking, 200/240 without ranking, 205/240 with auxiliary losses scaled by one tenth, and 206/240 with perturbation negatives. Table 8 evaluates the flow-only and perturbation recipes on held-out groups across three training seeds. The gains change sign, and the pooled totals are close (602 versus 599 successes out of 720). Shared value training therefore does not establish an average single-candidate improvement across these seeds.

Table 7: Earlier four-task actor-training study. One candidate is executed without value readout; each row has 240 episodes. These models were trained only on the four-task subset.

Training objective	Successes	SR (%)
HybridVLA, DDIM 8		
Diffusion only	176/240	73.3
Progress targets on demonstrations	196/240	81.7
Policy negatives + ranking	207/240	86.3
$\pi_{0.5}$, *Euler 8*, $\beta_2 = 0.95$		
Flow only	203/240	84.6
Perturbation negatives + ranking	225/240	93.8
Policy negatives + ranking	133/240	55.4
without ranking	191/240	79.6
auxiliary losses $\times 0.1$	200/240	83.3

Table 8: Actor performance across training seeds. $\pi_{0.5}$ on the four-task subset, with $\beta_2 = 0.999$ and held-out evaluation groups; 240 episodes per seed and model.

Training seed	Flow only (n/N)	Perturbation negatives (n/N)	ΔSR (pp)
42	186/240	205/240	+7.9
43	198/240	193/240	-2.1
44	218/240	201/240	-7.1
Pooled	602/720	599/720	-0.4

ΔSR is the success-rate difference from flow only. Seed 42 evaluates the development checkpoints on held-out groups; seeds 43 and 44 are independently trained.

C SELECTION AND COMPUTATION

Main-checkpoint comparisons. Table 9 varies candidate count with the value-trained checkpoints of Table 1, without retreat. Figure 3 shows the task-level changes from $K = 1$ to $K = 2$. A single candidate is the no-selection reference; selecting uniformly from identically distributed candidates has the same marginal action distribution, although it need not produce identical finite-sample counts. These results compare value selection with one candidate and do not establish an advantage over averaging candidates or spending the same computation on additional denoising steps.

A two-candidate decomposition. Fix a decision state and a continuation policy, and let q be a candidate's eventual success probability. If the two candidate outcomes are conditionally independent and identically distributed, both succeed with probability q^2, and their outcomes differ with probability $\rho = 2q(1 - q)$. If the evaluator selects the successful candidate with probability p_{rank} conditional on different outcomes, value selection succeeds with probability $q^2 + \rho p_{\text{rank}}$. Since $q = q^2 + \rho/2$, its improvement over a uniformly chosen candidate is

$$\Delta P = \rho\big(p_{\text{rank}} - \tfrac{1}{2}\big). \tag{10}$$

The same difference identity holds for a fixed distribution of candidate pairs without outcome independence: concordant pairs contribute zero gain, and on discordant pairs uniform selection succeeds with probability $1/2$. Averaging uses the accuracy conditional on discordant pairs. The quantities measured in Table 2 describe its sampled pair distribution; they do not directly predict the full-episode improvement of a policy that changes its state distribution at every decision.

Earlier four-task candidate sweep. Table 10 uses frozen checkpoints trained only on the four-task subset. HybridVLA improves up to $K = 4$ and decreases at $K = 8$; the stronger $\pi_{0.5}$ actor gains little and the weaker actor does not improve. Each row uses three fixed simulator-seed groups with 20 episodes per task and group. Latency is the mean request-to-action wall-clock time over decisions, excluding environment stepping. The matched-denoising panel is a separate evaluation cohort: two DDIM-4 candidates have the same denoiser evaluation count as one DDIM-8 candidate, but still incur value-scoring computation. Equal evaluation seeds do not ensure bit-identical simulator trajectories across these arms. None of these older counts or latencies are substituted for the main-checkpoint results.

Table 9: Candidate count and decision latency with the value-trained actors from Table 1. No recovery is used; $K = 1$ omits value readout. Counts are successes out of 60 per task; overall rates (%) use 240 episodes. Latency includes candidate generation and scoring, excludes environment stepping, and uses the same GPU model and batching across rows.

K	Box	Fridge	Toilet	Sweep	Overall	Latency (ms)
$\pi_{0.5}$						
1	6/60	53/60	55/60	21/60	135/240 (56.3)	94.54
2	28/60	58/60	60/60	27/60	173/240 (72.1)	209.82
4	31/60	56/60	58/60	25/60	170/240 (70.8)	397.74
8	23/60	51/60	56/60	19/60	149/240 (62.1)	782.33
HybridVLA						
1	40/60	39/60	53/60	15/60	147/240 (61.3)	112.34
2	50/60	52/60	53/60	14/60	169/240 (70.4)	233.28
4	55/60	53/60	57/60	10/60	175/240 (72.9)	475.79
8	37/60	41/60	54/60	13/60	145/240 (60.4)	959.21

The $K = 1$ rows share the evaluation in Table 3; $K = 2$ shares Table 1. Denoising cost scales as K times the per-candidate cost. Counts and latency are descriptive measurements; no per-group uncertainty is reconstructed from aggregate counts.

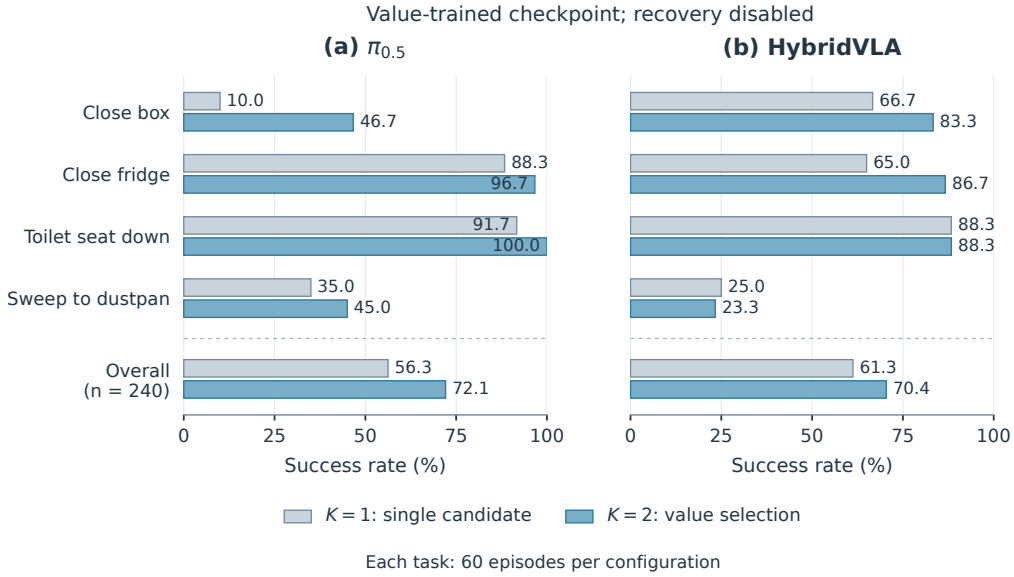


Figure 3: Candidate selection with the value-trained checkpoints used in Table 1. The first four RLBench tasks compare a single candidate ($K = 1$, no value readout) with the highest-value of two candidates ($K = 2$); recovery is disabled in both configurations. Each task has 60 episodes per configuration (three seed groups of 20), and Overall pools 240 episodes. Values are success rates computed from the aggregate success counts. No error bars are shown; these are descriptive comparisons, not significance tests.

Table 10: Candidate selection: success and inference cost. Frozen checkpoints on the four-task subset; sampler entries give the method and steps per candidate.

K	Sampler	Denoiser NFE	Successes (n/N)	SR (%)	Latency (ms)
HybridVLA (policy negatives): K sweep					
1	DDIM 4	4	184/240	76.7	–
2	DDIM 4	8	207/240	86.3	–
4	DDIM 4	16	216/240	90.0	–
8	DDIM 4	32	201/240	83.8	–
HybridVLA: matched-budget cohort (separate evaluation)					
2	DDIM 4	8	205/240	85.4	–
1	DDIM 8	8	207/240	86.3	–
$\pi_{0.5}$ *strong (perturbation neg.): K sweep*					
1	Euler 8	8	225/240	93.8	95
2	Euler 8	16	228/240	95.0	210
4	Euler 8	32	229/240	95.4	398
$\pi_{0.5}$ *weak (policy neg.): K sweep*					
1	Euler 8	8	133/240	55.4	94
2	Euler 8	16	132/240	55.0	209
4	Euler 8	32	125/240	52.1	394

NFE counts denoiser evaluations per decision; value scoring adds K forward passes for $K > 1$. Latency is the mean time from request to returned action; – means not reported. $K = 1$ rows never read the value. The matched-budget panel is a separate cohort compared within itself.

D RECOVERY AND SENSITIVITY

Alarm calibration. For each feature $f \in \{D, S\}$, let M_e^f be the maximum defined feature value in successful development episode e, and sort the n maxima. Each feature receives half the alarm budget, $\alpha_f = \alpha/2 = 0.05$ with $\alpha = 0.1$. Its threshold is $\eta_f = M^f_{(n-\lfloor \alpha_f n \rfloor - 1)}$ with zero-based indexing, so at most $\lfloor \alpha_f n \rfloor$ successful development episodes strictly exceed it. A numerical floor of $2/255$ is applied to thresholds. The stage-specific reference $\mu_{w,t}$ is the median w-step gain in successful development episodes when at least five observations are available, and the pooled median otherwise. Alarms require at least ten successful calibration episodes. The severity scale is

$$s_f = \max\Big(0.02, \operatorname{median}_e \left| M_e^f - \operatorname{median}_{e'} M_{e'}^f \right|, \eta_f\Big).$$

The evaluated adaptive rule in Eq. (9) uses severity thresholds at one and two calibration scales to select a depth from one to three, subject to history and remaining budget.

Physical retreat and scope. Retreat targets are measured end-effector poses recorded before completed forward commands. The controller revisits the last d targets in reverse while holding the current gripper command, then takes a new observation and resumes policy execution. Forward and retreat commands count against the same budget. This operation does not reset objects or the simulator. In the earlier controlled four-task study, each target allows at most two planning attempts and counts as reached within 1 cm and 5°; partial retreats remain in their assigned arm. That study requires three completed commands, three recorded poses, and at least four remaining commands before intervention, and allows one intervention per episode. These study-specific eligibility settings are not assumed to fill missing main-experiment metadata.

D.1 INFERENCE ALGORITHM

Algorithm 1 VaSeR inference with value selection and trend-aware backtracking

Require: actor π_θ, value token V_θ, calibrated thresholds and scales, candidates K, physical budget B

pose history $H \leftarrow \emptyset$; value history $\mathcal{V} \leftarrow \emptyset$
while task not done **and** commands issued $< B$ **do**
  observe x_t; sample $a_t^{(1:K)} \sim \pi_\theta(\cdot \mid x_t)$ and score each candidate
  choose $a_t = a_t^{(k^*)}$, where $k^* = \arg\max_k V_\theta(x_t, a_t^{(k)})$
  append $v_t = V_\theta(x_t, a_t)$ to $\mathcal{V}$; compute D_t, S_t (Eq. 8)
  if a trend exceeds its threshold **and** retreat is eligible **then**
    set d_t by Eq. 9; revisit the last d_t recorded poses in reverse
    count each retreat command against B; reset $\mathcal{V}$; **continue**
  end if
  execute a_t and count it against B; on completion record its pre-action pose
end while

D.2 DEPTH COMPARISONS

Table 11 varies only retreat depth for the main checkpoints on four tasks. Candidate count, alarm thresholds, command budget, and evaluation seeds are shared across its arms. Fixed depth uses $d_t = \min(h_t, B_t - 1, d)$. The no-retreat and adaptive rows are reused from Table 1. Figure 4 separately shows the effect of adding adaptive recovery over all ten tasks. These are descriptive component and depth comparisons; they are not tests of intervention timing.

Earlier fixed-depth sensitivity study. Table 12 uses the strong and weak four-task $\pi_{0.5}$ actors, two command budgets, and four depths (including no retreat). Four tasks, three evaluation seeds, and ten episodes per cell give 1,920 formal episodes; 24 engineering checks are excluded. Calibration uses disjoint development groups with 60 episodes per actor, task, and budget (120 for Sweep to Dustpan); the $B = 20$ arms reuse the $B = 10$ calibration. The twelve prespecified depth comparisons are paired by task, seed, and episode index, use exact two-sided tests, and receive a Holm correction as one family. All assigned episodes remain in the analysis, including those with no eligible alarm or incomplete retreat. No depth comparison is significant after correction (minimum adjusted $p = 0.375$). This earlier result concerns different checkpoints and should not be pooled with Table 11.

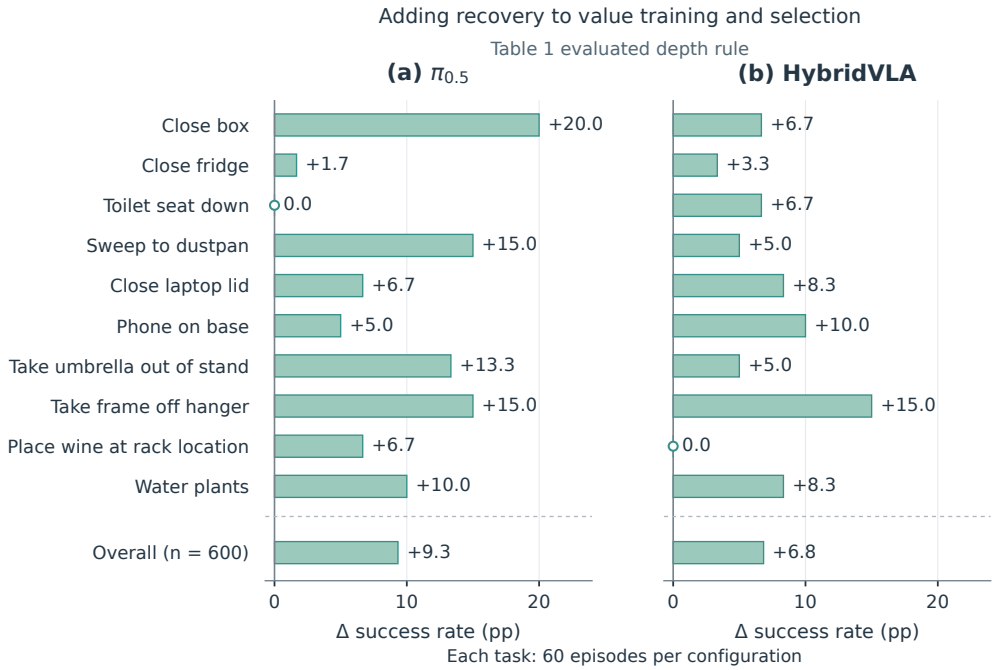


Figure 4: Increment from adding recovery to value training and selection in Table 1, across all ten RLBench tasks. Each bar is the success-rate difference between +Recovery and +Value & selection, computed from integer success counts: 60 episodes per task and configuration (three groups of 20), or 600 for Overall. Both configurations use $K = 2$. Recovery uses the severity-based depth rule described in the method. No error bars are shown; all differences, including zeros, are descriptive.

Table 11: Retreat-depth comparison on the first four tasks with the main-comparison checkpoints. Rows use $K = 2$, the same alarm thresholds, command budget, and evaluation seeds. Fixed depth d uses $d_t = \min(h_t, B_t - 1, d)$. Cells are successes/60; overall rates (%) use 240 episodes.

Retreat rule	Box	Fridge	Toilet	Sweep	Overall
$\pi_{0.5}$					
No retreat	28/60	58/60	60/60	27/60	173/240 (72.1)
Fixed 1 step	31/60	57/60	59/60	23/60	170/240 (70.8)
Fixed 2 steps	32/60	55/60	55/60	25/60	167/240 (69.6)
Fixed 3 steps	22/60	60/60	57/60	24/60	163/240 (67.9)
Adaptive 1–3 steps	40/60	59/60	60/60	36/60	195/240 (81.3)
HybridVLA					
No retreat	50/60	52/60	53/60	14/60	169/240 (70.4)
Fixed 1 step	51/60	53/60	50/60	15/60	169/240 (70.4)
Fixed 2 steps	55/60	56/60	52/60	17/60	180/240 (75.0)
Fixed 3 steps	54/60	51/60	49/60	19/60	173/240 (72.1)
Adaptive 1–3 steps	54/60	54/60	57/60	17/60	182/240 (75.8)

No retreat and adaptive retreat reuse the selection and recovery rows of Table 1. Adaptive retreat follows Eq. (9), with severity boundaries at one and two calibration scales.

Table 12: Effect of backtracking depth. Four-task subset with the same alarms and eligibility in every arm; cells show successes out of 120 (SR in %).

Budget B	No retreat	$d = 1$	$d = 2$	$d = 3$
Strong (perturbation neg.)				
10	111 (92.5)	114 (95.0)	111 (92.5)	108 (90.0)
20	111 (92.5)	112 (93.3)	112 (93.3)	113 (94.2)
Weak (policy neg.)				
10	61 (50.8)	64 (53.3)	63 (52.5)	59 (49.2)
20	61 (50.8)	63 (52.5)	64 (53.3)	59 (49.2)

Each retreat command counts toward B. No depth comparison is significant after Holm correction (smallest adjusted $p = 0.375$).

E Recovery cases and interpretation

Repeated executions in an earlier four-task study illustrate both the value and the limits of retreat. Three Sweep to Dustpan cases with the strong actor at $B = 10$ were replayed five times per intervention using the recorded simulator and candidate seeds. In one case, continuing failed in all five runs and a one-pose retreat succeeded in all five. In another, the original candidate succeeded but a value-selected alternative failed; in the third, none of the tested interventions succeeded. These selected cases illustrate reproducible episode-specific behavior rather than estimating a population recovery rate. They do not establish that retreat restores displaced objects. We do not report a scene-intact versus scene-moved subgroup analysis, and the method's applicability remains limited by physical reachability, the remaining command budget, and whether the resumed state lies within the actor's useful operating range.

F Data and evaluation protocols

Main ten-task comparison. Table 1 uses three seed groups with 20 episodes per task and group: 60 episodes per task, 600 per configuration, and 3,600 across its six configurations. The seed sheet contains 180 integer success counts out of 20. Each task rate pools its three groups. For the Mean column, each group supplies a rate over 200 episodes; the standard deviation uses the sample denominator $3 - 1$. The $\pi_{0.5}$ baseline, value-and-selection, and complete-method group totals are $(115, 123, 123)$, $(114, 141, 148)$, and $(140, 161, 158)$; HybridVLA totals are $(104, 100, 110)$, $(98, 126, 127)$, and $(116, 135, 141)$. Equal group sizes make the mean identical to the pooled rate

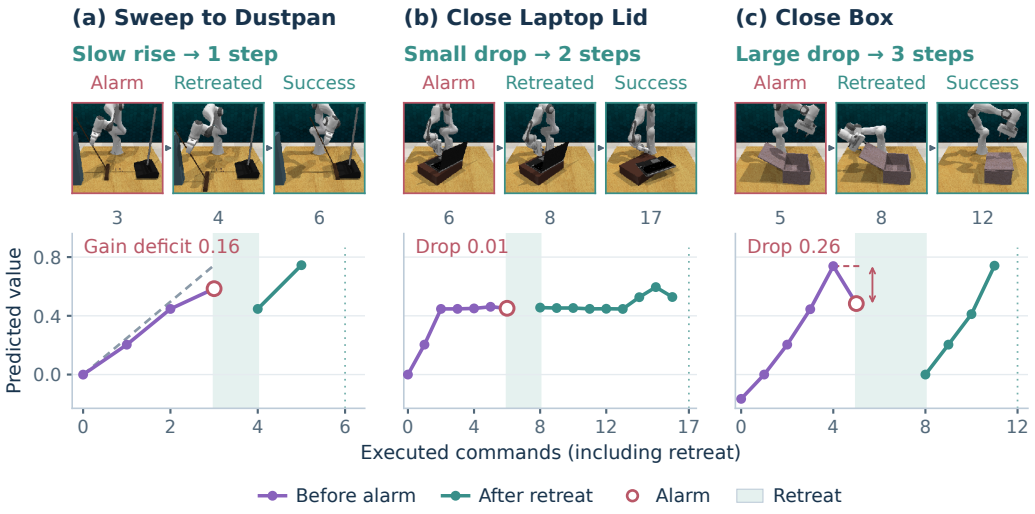


Figure 5: **Recorded recovery examples.** Three selected successful $\pi_{0.5}$ episodes with fixed retreat depths: (a) slow value growth and one-pose retreat, (b) a small drawdown and two-pose retreat, and (c) a large drawdown and three-pose retreat. Frames show the alarm, the end of retreat, and task completion. Numbers below frames count executed commands; dotted vertical lines mark success. Curves are the selected-action values logged before execution; shaded intervals contain physical retreat commands, which have no value readout. The dashed line in (a) shows the calibrated successful-run gain reference. All plots share a value scale; their command ranges differ. These examples illustrate recovery behavior, not online depth selection or a success-rate estimate.

over 600 episodes. The retained group ordering does not identify whether the randomization is training or evaluation randomness, so these standard deviations are not evidence of variation across independently trained models. Detailed checkpoint and run metadata are incomplete; the four-task training settings in Table 6 should not be interpreted as a full record of the ten-task training recipe.

Four-task component comparison. Table 13 uses the main checkpoints on Close Box, Close Fridge, Toilet Seat Down, and Sweep to Dustpan, with 60 episodes per task and 240 per row. The baseline, value-and-selection, and complete-method rows are reused from Table 1. Value training alone uses $K = 1$ without reading the value at inference. Value training with recovery alone also uses $K = 1$, but scores the candidate to monitor trends. Selection uses $K = 2$. Repeated rows across tables denote the same evaluations rather than independent replications.

Evaluator data source. Table 2 uses the main checkpoints on the first four tasks. The VaSeR actor generates both candidate pairs and alarm-evaluation episodes. The complete, no-negative, and detached evaluators score these same data, isolating the evaluator comparison from actor sampling.

Candidate pairs. For each backbone, each of four tasks uses three random seeds, with five scored candidate pairs per seed: $4 \times 3 \times 5 = 60$ pairs, or 15 per task. At a held-out decision point, the simulator state is recorded, two candidates are sampled, and each is executed from that same state. The same actor then continues with one candidate and no retreat. A pair is scored only if one branch ultimately succeeds and the other fails; it is correct when the successful candidate receives the higher value. Executing both branches to completion and finding discordant outcomes is costly, motivating this sample size. For $\pi_{0.5}$, 60 of 132 sampled pairs are scored ($\rho = 45.5\%$); for HybridVLA, 60 of 124 are scored ($\rho = 48.4\%$). These fractions describe the collected sample rather than a universal rate of candidate disagreement.

Alarm evaluation and statistical interpretation. Each backbone supplies 60 held-out single-candidate, no-retreat episodes with observations and values logged at each decision. Each evaluator calibrates its own thresholds using a separate development set containing successful episodes only.

Table 13: Component decomposition on the first four tasks, using the main-comparison checkpoints. Each cell is successes/60; overall rates (%) use 240 episodes. Baseline, selection, and complete-method rows reuse Table 1.

Configuration	Value	Select	Recover	Box	Fridge	Toilet	Sweep	Overall
$\pi_{0.5}$								
Baseline	–	–	–	4/60	51/60	59/60	22/60	136/240 (56.7)
Value training	✓	–	–	6/60	53/60	55/60	21/60	135/240 (56.3)
+ Selection	✓	✓	–	28/60	58/60	60/60	27/60	173/240 (72.1)
+ Recovery, $K = 1$	✓	–	✓	25/60	59/60	55/60	23/60	162/240 (67.5)
Complete VaSeR	✓	✓	✓	40/60	59/60	60/60	36/60	195/240 (81.3)
HybridVLA								
Baseline	–	–	–	42/60	42/60	52/60	12/60	148/240 (61.7)
Value training	✓	–	–	40/60	39/60	53/60	15/60	147/240 (61.3)
+ Selection	✓	✓	–	50/60	52/60	53/60	14/60	169/240 (70.4)
+ Recovery, $K = 1$	✓	–	✓	48/60	50/60	54/60	15/60	167/240 (69.6)
Complete VaSeR	✓	✓	✓	54/60	54/60	57/60	17/60	182/240 (75.8)

Value training alone uses $K = 1$ without value readout. Recovery without selection also uses $K = 1$ but reads its candidate's value to monitor the trend. Rows with selection use $K = 2$. Both baselines omit value training; HybridVLA uses diffusion only.

An episode alarms if any step exceeds a threshold; a failed episode with an alarm and a successful episode without one are both correct. Alarm accuracy divides the number of correct episodes by 60. The $4 \times 3 \times 5$ grouping above describes candidate pairing, not the alarm dataset. Evaluator differences are descriptive: joint per-pair correctness records were not retained, so no paired significance test is reported. The pairwise ranking sample and the episode-level alarm sample answer distinct questions and are not pooled.

G Real-robot setup

Platform and observations. The experiments use a seven-degree-of-freedom Franka Research 3 arm with a parallel-jaw gripper. An Intel RealSense D435i and a D435 provide external workspace views, and a wrist-mounted D405 provides a close view. The three image streams are stored at 224×224 pixels. The robot state is the end-effector position and 6D rotation representation (nine dimensions).

Actions and demonstrations. Each 10-dimensional action contains a translation increment in the robot base frame, the relative rotation $R_t^\top R_{t+1}$ in a 6D representation, and the next gripper opening. Training uses subsets of 300 Bread Placement and 500 Pen Collection demonstrations. The full recordings contain 310 demonstrations (126,061 steps, mean 40.7 s) and 556 demonstrations (249,480 steps, mean 44.9 s), respectively; these totals describe the full recordings rather than the training subsets. Recordings are downsampled to 10 Hz, with each step pairing the images, state, and action.

Tasks and language. In Bread Placement, two slices start in a toaster; success requires taking both out and placing them on a cutting board. In Pen Collection, two pens start on the tabletop; success requires placing both in a holder. The dataset metadata use the short descriptions "move the bread" and "pick up the pen". The full bread instruction supplied to the model is "Take the bread out of the toaster and place it on the cutting board" (Figures 1 and 2).

Configurations and evaluation. As in simulation, $\pi_{0.5}$ uses flow matching and HybridVLA uses only diffusion, without autoregressive action training or gated fusion. The baseline is trained with the action objective alone. + *Value & selection* adds value training and candidate selection, and + *Recovery* adds trend-aware backtracking. Each task, backbone, and configuration receives 30 trials, giving 360 trials. These finite-sample results are descriptive. The retained records establish the setup and aggregate outcomes but not a complete per-run protocol; the findings are limited to these two task settings.