ARE VISION-LANGUAGE-ACTION MODELS ROBUST TO ONE-STEP OBSERVATION PERTURBATIONS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding the safety risks of vision-language-action (VLA) models is essential for their deployment in the physical world. Existing safety research has mainly considered persistent perturbations that are applied continuously to observations throughout an episode. However, momentary observation corruption, in which observations are severely perturbed only briefly within an episode, remains an underexplored safety threat. To address this gap, this work investigates robustness to one-step perturbations applied at a single time step per episode. Our experiments reveal that these perturbations substantially degrade VLA performance and that their impact depends on the action chunk execution length. Based on them, we propose CARE, which dynamically selects the execution length based on consistency with the previously predicted action chunk. CARE improves robustness with low computational overhead while preserving clean performance by selecting shorter execution lengths only under perturbations.

1 INTRODUCTION

Vision-Language-Action (VLA) models provide a foundation for robot control by extending visual and linguistic understanding to action generation (Kim et al., 2024; Black et al., 2024). Since VLA outputs are executed as physical actions, inappropriate actions can not only cause task failure, but also damage nearby objects or harm people. Safe deployment therefore requires VLAs to remain reliable under observation degradation that arises in real-world environments.

One important safety concern is momentary observation corruption, where observations are severely degraded for only a brief period. For example, sudden illumination changes can temporarily overexpose a camera, and brief communication failures can lead to abnormal sensor values. While seemingly innocuous, these corruptions can cause irreversible damage when they occur at critical moments, such as dropping and breaking an object or leaving it out of the robot's reach. As a result, the robot may be unable to resume the task even after reliable observations are restored. However, robustness to such corruptions has received limited attention in prior work (Guo et al., 2026; Wang et al., 2025b; Hancock et al., 2025; Zhang et al., 2025b). Existing evaluations typically consider *persistent perturbations* which are applied continuously throughout an episode, leaving robustness to momentary observation corruption poorly understood.

In this work, we study the robustness of VLA models to *one-step perturbations*, each applied at exactly one step in an episode (Figure 1 (a)). We design eight types of perturbations that emulate momentary observation corruptions and systematically evaluate their effects across five VLA models. For each perturbation, we evaluate every inference step as a candidate intervention time in separate rollouts, with each episode receiving exactly one perturbed observation. Our experiments reveal two important findings about the robustness of current VLAs:

Finding 1: One-step perturbations at critical moments cause substantial performance degradation across evaluated VLAs. The impact of a one-step perturbation varies substantially depending on when the perturbation occurs. To characterize performance at particularly vulnerable intervention steps, we report the average of the lowest 10% of success rates across evaluated steps. Under this metric, every evaluated model exhibits a drop of more than 50 percentage points relative to its clean success rate under at least one perturbation. These results suggest that even a single corrupted observation can cause substantial performance degradation when it occurs at a critical moment.

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.

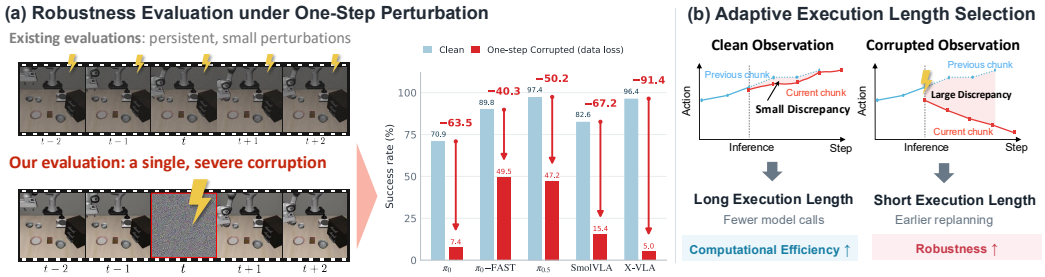

Figure 1: **Overview.** (a) We evaluate VLA robustness to one-step perturbations and observe substantial success rate drops across 40 LIBERO tasks. Corrupted scores average the lowest 10% of success rates across evaluated intervention steps. (b) Adaptive execution length selection based on prediction consistency shortens execution under perturbations to improve robustness.

Finding 2: Vulnerability to one-step perturbations depends on the action execution length. Action chunking (Zhao et al., 2023) is widely used in VLAs to predict several future actions at once. The robot then executes a fixed number of predicted actions before the next prediction, which we call the execution length. If a perturbation causes the model to predict an incorrect action chunk, the robot continues executing actions from that chunk until the next prediction. We find that shortening the execution length reduces this vulnerability. However, a shorter execution length requires more frequent inference and may reduce action consistency, creating a trade-off between robustness and task performance (Chi et al., 2025; Black et al., 2026).

Motivated by these findings, we propose Consistency-Aware Robust Execution length selection (CARE), which adjusts the execution length based on the consistency of consecutive action predictions (Figure 1 (b)). At each inference step, our method compares the newly predicted action sequence with the unexecuted portion of the previous prediction. A large difference suggests that the current observation or action prediction may be unstable. In this case, the method selects a short execution length to prevent an incorrect action sequence from being executed for an extended period. This simple design requires no additional model calls or training and is broadly applicable to VLA models that predict action chunks. Experiments show that CARE substantially improves robustness to one-step input perturbations with low computational overhead, while largely preserving clean task performance for most evaluated models.

Our contributions are summarized as follows:

- We present the first systematic study of VLA robustness to one-step perturbations and reveal severe task-performance degradation across several VLA models.
- We analyze this vulnerability and identify longer action-chunk execution lengths as a key factor that amplifies the effect of a one-step perturbation.
- We propose CARE, a training-free and broadly applicable method for adaptive execution length selection, and demonstrate its effectiveness across multiple models and benchmarks.

2 Related Work

This section provides an overview of related work, with further details in Appendix A.

Robustness to Input Perturbations. A growing body of work has investigated VLA safety to support safe deployment in real-world environments (Zhang et al., 2025a; Robey et al., 2025; Jones et al., 2025; Li et al., 2026). One focus of this research is robustness to input perturbations during inference. Studies in this area have evaluated VLA robustness to perturbations affecting image inputs, state inputs, or both, reporting substantial drops in task success rates (Guo et al., 2026; Wang et al., 2025b; Hancock et al., 2025; Wang et al., 2025a; Lu et al., 2026; Xie et al., 2026; Zhang et al., 2025c; Xu et al., 2025; Li et al., 2025). However, these studies mainly assume persistent perturbations, which are continuously applied throughout an episode. This work evaluates the robustness against one-step perturbations that intervene at only a single step.

Action Chunking. Action chunking (Zhao et al., 2023; Chi et al., 2025) predicts multiple future actions in a single inference pass and sequentially executes a fixed number of them. While this mech-

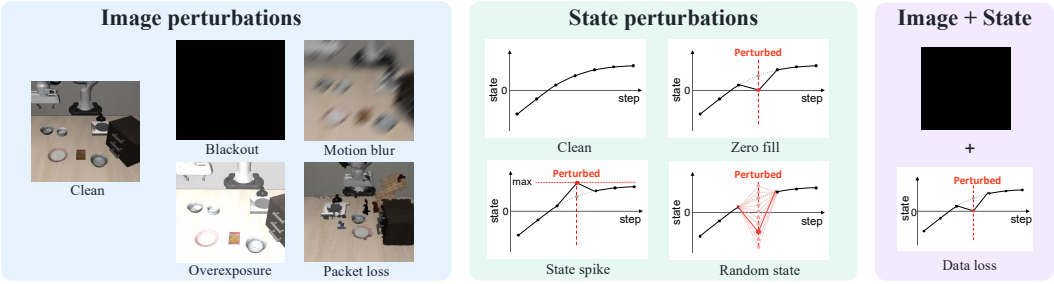

Figure 2: **Eight types of one-step perturbations.** Each perturbation corrupts the image input, the state input, or both at a single inference step.

anism enables high-frequency robot control, it reduces responsiveness to rapid changes in motion. To address this trade-off, several methods have been proposed to dynamically adjust the execution length (Wang et al., 2026b; Liang et al., 2026; Pan et al., 2026; Wang et al., 2026a; Jing et al., 2025). However, these methods do not consider perturbations to the observations, limiting their contribution to robustness. In this work, we propose CARE, an adaptive execution length selection method that uses the consistency with the predicted action chunk to improve robustness. While prior work has used temporal consistency across action predictions for other purposes (Agia et al., 2025; Liu et al., 2025b; So et al., 2025), CARE first uses it for adaptive execution length selection.

3 Preliminaries

We consider a robotic control setting in which a VLA policy makes sequential decisions to control a robot to accomplish a goal specified by a language instruction. At step t, the policy π_θ takes three inputs as observation: an image input v_t consisting of RGB images; a robot state s_t, such as the end-effector pose and gripper state; and a language instruction l. The policy then outputs an action chunk consisting of K future actions:

$$\mathbf{A}_t = (a_t, a_{t+1}, \ldots, a_{t+K-1}) \sim \pi_\theta\left(\cdot \mid v_t, s_t, l\right), \tag{1}$$

where a_t denotes the action for step t. The robot executes only the first $H \leq K$ actions $(a_t, \ldots, a_{t+H-1})$ of the predicted chunk before the next policy inference. We refer to H as the execution length. After these H actions have been executed, the policy generates the next action chunk $\mathbf{A}_{t+H}$ from v_{t+H} and s_{t+H}.

4 Robustness Evaluation to One-Step Perturbations

We first ask whether momentary observation corruption can cause task failure in current VLAs. Unlike persistent perturbations used in prior evaluations, real-world observation corruption caused by overexposure, motion blur, or sensor input loss may last only briefly. To study this threat in a controlled setting, we consider one-step perturbations, which corrupt the observation at a single step while leaving observations at all other steps unchanged. We evaluate robustness against these perturbations and investigate why its effects may persist beyond the perturbed step.

4.1 Experimental Setup

Models and Benchmarks. We evaluate five VLA models: π_0 (Black et al., 2024), π_0-FAST (Pertsch et al., 2025), $\pi_{0.5}$ (Black et al., 2025), SmolVLA (Shukor et al., 2025), and X-VLA (Zheng et al., 2026). For each model, we select the execution length from $H \in \{1, 5, 10, 15, 20, 25, 30\}$ that achieves the highest task success rate without perturbations. Specifically, we use $H = 15$ for π_0, $\pi_{0.5}$, and X-VLA, and $H = 10$ for π_0-FAST and SmolVLA. As the simulation environments, we use four task suites from the LIBERO benchmark (Liu et al., 2023). Further details are provided in Appendix C.

Perturbations. We consider eight perturbations that emulate momentary observation corruptions that may occur in real-world environments (Figure 2). We group them into three categories by input modality. **(i) Image perturbations:** These simulate disruptions to camera capture or image transmission. We apply Blackout, Overexposure, Motion blur, and Packet loss. **(ii) State perturbations:**

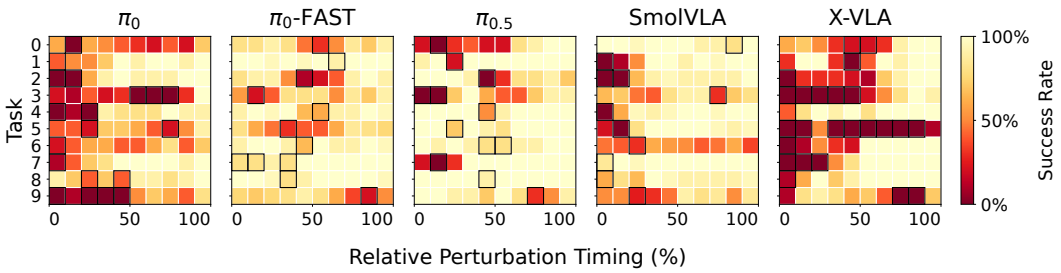

Figure 3: **Perturbation timing strongly affects task success.** Heatmaps show success rates under data loss on LIBERO-Goal, with black boxes marking task-wise minima.

Table 1: **Existing VLAs are vulnerable to one-step perturbations.** Clean success rate and Bottom10%-SR under each perturbation (%), averaged over 40 tasks on LIBERO.

Model	Clean	Image input				State input			Image + state
		Overexp.	Blackout	Blur	Packet loss	Zero fill	Spike	Random	Data loss
π_0	70.9	53.9 (↓ 17.0)	48.1 (↓ 22.8)	51.4 (↓ 19.5)	50.2 (↓ 20.7)	5.9 (↓ 65.0)	13.8 (↓ 57.1)	19.6 (↓ 51.3)	7.4 (↓ 63.5)
π_0-FAST	89.8	44.4 (↓ 45.4)	45.0 (↓ 44.8)	36.7 (↓ 53.1)	75.6 (↓ 14.2)	79.7 (↓ 10.1)	72.9 (↓ 16.9)	76.1 (↓ 13.7)	49.5 (↓ 40.3)
$\pi_{0.5}$	97.4	88.0 (↓ 9.4)	45.9 (↓ 51.5)	64.9 (↓ 32.5)	91.1 (↓ 6.3)	90.1 (↓ 7.3)	91.6 (↓ 5.8)	91.0 (↓ 6.4)	47.2 (↓ 50.2)
SmolVLA	82.6	33.3 (↓ 49.3)	34.8 (↓ 47.8)	44.1 (↓ 38.5)	69.5 (↓ 13.1)	53.3 (↓ 29.3)	52.3 (↓ 30.3)	58.5 (↓ 24.1)	15.4 (↓ 67.2)
X-VLA	96.4	90.5 (↓ 5.9)	28.6 (↓ 67.8)	38.5 (↓ 57.9)	41.5 (↓ 54.9)	15.0 (↓ 81.4)	2.8 (↓ 93.6)	10.4 (↓ 86.0)	5.0 (↓ 91.4)

These simulate sensor malfunctions or communication failures. We apply Zero fill, State spike, and Random state, which replace the affected state components with zero, their upper bounds, or random outliers, respectively. **(iii) Image-and-state perturbations:** This simulates a complete interruption of observation inputs. We apply Data loss, which replaces the input image with a black image and all state components with zero. Implementation details are provided in Appendix C.2.

Evaluation Protocol. We evaluate how task performance depends on when a perturbation is applied. Since the policy receives observations only at inference steps, we evaluate each intervention step in separate rollouts, applying a perturbation at exactly one inference step per episode. For example, when $H = 10$, the perturbation is applied at a single inference step selected from $t = 1, 11, 21, \ldots$. For each model, task, and perturbation, we run 10 episodes per intervention step. We also report ***Bottom10%-SR*** to assess performance at particularly vulnerable steps. This metric is the mean of the lowest 10% of success rates across evaluated intervention steps.

4.2 Existing VLA Models are Vulnerable to One-Step Perturbations

Perturbation timing strongly affects task success. Figure 3 shows success rates under data loss across perturbation steps for five VLA models on LIBERO-Goal. Relative perturbation timing is defined as the normalized position of the perturbed inference step within an episode. Even for the same model and task, success rates differ substantially depending on when the perturbation is applied. Although the most vulnerable steps differ across models and tasks, earlier perturbations tend to cause larger reductions in task success. This pattern may arise from compounding errors, a well-known problem in imitation learning (Ross et al., 2011; Xu et al., 2024). These findings suggest that robustness evaluations should account for particularly vulnerable steps. Additional results and discussion are provided in Appendix D.1.

Robustness at vulnerable steps. We next compare model robustness at vulnerable perturbation steps using Bottom10%-SR. Table 1 shows that every evaluated model has a Bottom10%-SR more than 50 percentage points below its clean success rate under at least one perturbation. These results reveal substantial vulnerability to one-step perturbations. Furthermore, the perturbations that expose these vulnerabilities differ across models. π_0-FAST and SmolVLA are particularly sensitive to image perturbations, whereas π_0 and X-VLA are especially vulnerable to state perturbations.

4.3 Execution Length Amplifies Perturbation Effects

We hypothesize that action chunking contributes to the observed vulnerability to one-step perturbations. A single perturbed observation can cause incorrect actions over multiple consecutive steps,

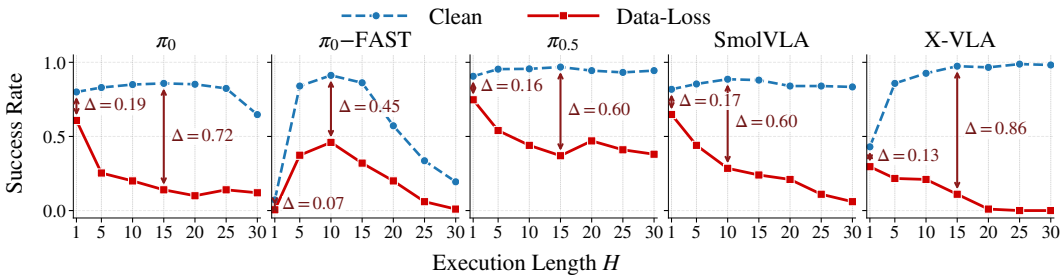

Figure 4: **Execution Length Amplifies Perturbation Effects**. Clean success rate and Bottom10%-SR under Data-loss perturbation across execution lengths H.

making recovery more difficult. To test this hypothesis, we compare robustness across execution lengths H. Because clean performance also depends on H, we report both the clean success rate and the Bottom10%-SR under perturbation. A smaller gap between these curves indicates lower sensitivity to the perturbation. As shown in Figure 4, the gap between the clean success rate and Bottom10%-SR widens as H increases for most models. In contrast, the gap is small at $H = 1$, where the policy can replan from a new observation at every step, limiting the persistence of erroneous actions. An exception is π_0-FAST, whose clean performance declines at longer execution lengths, leaving less room for further degradation. These results support our hypothesis that action chunking amplifies vulnerability to one-step perturbations by prolonging their effects.

5 Method

The preceding analysis shows that shorter execution lengths improve robustness to one-step perturbations. However, always using a short execution length does not necessarily balance robustness and task performance. Short execution lengths can degrade performance by reducing consistency across action chunks (Chi et al., 2025; Black et al., 2026) as shown in Figure 4, and increase computational cost by requiring more frequent model inference. To address this trade-off, we propose CARE, an adaptive execution length selection method. CARE aims to balance robustness, task performance, and computational efficiency by selecting short execution lengths only under perturbations while retaining long execution lengths for clean observations.

5.1 Formulation

We begin by formulating execution length selection based on consistency between action chunks. Let $\mathbf{A}_t^* \sim \pi_\theta(\cdot \mid v_t, s_t, l)$ and $\mathbf{A}_t \sim \pi_\theta(\cdot \mid v_t', s_t', l)$ be action chunks generated from clean and perturbed observations at step t, respectively. If $\mathbf{A}_t$ differs substantially from $\mathbf{A}_t^*$, executing more actions may move the robot farther from the desired trajectory. We therefore define the ideal execution length H^* as the longest prefix whose discrepancy from the clean action chunk does not exceed a threshold τ:

$$H^* = \max\left\{ H \in \{1, \dots, K\} \;\middle|\; D\left(\mathbf{A}_t^{1:H}, \mathbf{A}_t^{*,1:H}\right) \le \tau \right\}, \tag{2}$$

where $\mathbf{A}_t^{i:j}$ denotes the subsequence of actions indexed from i to j within $\mathbf{A}_t$, and $D(\cdot,\cdot)$ measures the discrepancy between two action chunks. A small discrepancy from the oracle actions yields a long execution length to reduce inference cost, whereas a large discrepancy yields a short execution length to limit the effect of the perturbation.

Unfortunately, the clean observation (v_t, s_t) is unavailable at inference time, so $\mathbf{A}_t^*$ cannot be computed directly. To address this issue, we use the unexecuted portion of the previous action chunk $\mathbf{A}_{t-H_{\text{prev}}}$ as a reference for the current prediction, where H_{prev} is the previous execution length. If the environment has not changed substantially since the previous inference step, the unexecuted actions $\mathbf{A}_{t-H_{\text{prev}}}^{H_{\text{prev}}+1:K}$ should approximate the corresponding current clean actions. For a candidate execution length H, we therefore compare $\mathbf{A}_t^{1:H}$ with $\mathbf{A}_{t-H_{\text{prev}}}^{H_{\text{prev}}+1:H_{\text{prev}}+H}$, which covers the same future steps.

This comparison requires at least H unexecuted actions in the previous action chunk. To satisfy this condition for every candidate execution length, CARE limits execution to $H_{\max} = \lfloor K/2 \rfloor$ actions

per chunk. We therefore reformulate execution length selection as

$$H_t = \max \left\{ H \in \{1, \dots, H_{\text{max}}\} \;\middle|\; D\Big(\mathbf{A}_t^{1:H}, \mathbf{A}_{t-H_{\text{prev}}}^{H_{\text{prev}}+1:H_{\text{prev}}+H}\Big) \leq \tau \right\}. \tag{3}$$

This execution-length cap is consistent with existing VLA deployment settings, where execution lengths are often less than half the action chunk length K (Shukor et al., 2025). CARE therefore requires no extension of the model's native action chunk in many cases.

Our consistency-based design offers practical advantages for deployment. Methods that select or update action chunks using multiple policy predictions incur additional inference costs (Liang et al., 2026; Liu et al., 2025b; So et al., 2025), while those relying on learned monitors or verifiers require additional training (Pan et al., 2026; Wang et al., 2026b). By reusing the unexecuted portion of the previous action chunk, CARE requires neither additional training nor extra policy inference for execution-length selection, and leaves the model architecture unchanged.

5.2 Design of the Discrepancy Function

Our framework does not depend on a particular discrepancy function and can incorporate any function that quantifies consistency between action chunks. This section provides one concrete instantiation of this general framework. For each relative position $i \in \{0, \dots, H_{\text{max}} - 1\}$, we compute the discrepancy r_i between the current action $\mathbf{a}_{t+i}$ and the corresponding unexecuted action $\hat{\mathbf{a}}_{t+i}$ using the normalized Euclidean distance to reduce sensitivity to action magnitude:

$$r_i = \frac{\|\mathbf{a}_{t+i} - \hat{\mathbf{a}}_{t+i}\|_2}{\|\mathbf{a}_{t+i}\|_2 + \|\hat{\mathbf{a}}_{t+i}\|_2 + \epsilon}, \tag{4}$$

where $\epsilon > 0$ is a small constant added to avoid division by zero. Relying solely on per-step discrepancies has two limitations. First, the method may overreact to a large discrepancy at a single step, even without perturbations. For example, a one-step flip in the binary gripper action between open and closed may produce a large discrepancy. Second, because we use unexecuted actions from the preceding prediction as a surrogate for the current clean action chunk, discrepancies due to natural prediction errors are more likely at more distant future steps. We therefore define the discrepancy between action chunks as the discounted cumulative discrepancy across steps:

$$D\left(\mathbf{A}_t^{1:H}, \hat{\mathbf{A}}^{1:H}\right) = \sum_{i=0}^{H-1} \gamma^i r_i, \tag{5}$$

where $\gamma \in (0, 1)$ is a discount factor. This accumulation emphasizes discrepancies that persist across several steps. In contrast, isolated discrepancies at a single step have less influence. Discounting discrepancies in the distant future also reduces the influence of surrogate prediction errors. Together, these choices prevent overreaction to discrepancies unrelated to perturbations and stabilize execution length selection under clean observations. We provide the pseudocode in Algorithm 1.

6 Experiments

We empirically evaluate our proposed method on the LIBERO (Liu et al., 2023) and Meta-World (Yu et al., 2020) benchmarks and on a real robot. Due to space limitations, this section reports only the LIBERO and real-world results. Implementation details and additional experiments, including the Meta-World results, are provided in Appendices C and D.

6.1 Setup

For the proposed method, we set the discount factor to $\gamma = 0.85$, the maximum execution length to 15. We set the threshold to $\tau = 1.5$ for π_0, π_0-FAST, and SmolVLA, and to $\tau = 1.0$ for $\pi_{0.5}$ and X-VLA. These parameters were selected based on LIBERO-goal results without tuning on other suites. We use the same models, perturbations, and evaluation metrics as in Section 4.

Baseline Methods. To the best of our knowledge, no existing method is specifically designed to improve robustness to momentary observation corruptions. Nevertheless, we compare our method with the three methods from the perspectives of robustness and adaptive execution length selection.

Table 2: **CARE improves robustness to one-step observation perturbations across diverse VLA models.** Clean success rate and Bottom10%-SR (%, ↑) across 40 tasks on LIBERO.

Model	Method	Clean	Image input				State input			Image + state
			Overexp.	Blackout	Blur	Packet loss	Zero fill	Spike	Random	Data loss
π_0	Vanilla	70.9	53.9	48.1	51.4	50.2	5.9	13.8	19.6	7.4
	RobustVLA	72.2	53.4	52.1	52.8	54.1	10.1	17.2	21.9	6.9
	VLA-Corrector	69.8	54.2	**53.0**	53.6	54.1	9.8	21.5	24.2	11.5
	AAC	70.1	50.6	49.4	51.1	52.0	10.2	10.6	21.0	9.6
	CARE (ours)	72.5	**54.8**	52.9	**54.0**	**56.2**	**51.2**	**47.4**	**50.5**	**53.5**
π_0-FAST	Vanilla	89.8	44.4	45.0	36.7	75.6	79.7	72.9	76.1	49.5
	RobustVLA	91.5	44.1	49.9	56.2	**77.1**	**79.2**	**74.6**	**77.6**	55.5
	VLA-Corrector	–	–	–	–	–	–	–	–	–
	AAC	83.0	42.2	41.6	33.8	64.2	67.5	64.6	65.0	42.0
	CARE (ours)	78.2	**62.5**	**63.4**	**64.5**	63.5	64.4	65.4	63.4	**64.2**
$\pi_{0.5}$	Vanilla	97.4	88.0	45.9	64.9	91.1	90.1	91.6	91.0	47.2
	RobustVLA	97.8	58.1	53.4	61.3	88.5	87.9	87.0	89.4	53.2
	VLA-Corrector	99.0	**93.6**	51.6	70.1	**95.6**	**96.6**	**95.9**	**97.5**	51.4
	AAC	96.2	88.5	64.0	67.5	89.1	89.8	89.4	88.9	63.6
	CARE (ours)	97.2	92.1	**92.0**	**92.2**	93.5	93.1	93.5	92.1	**92.4**
SmolVLA	Vanilla	82.6	33.3	34.8	44.1	69.5	53.3	52.3	58.5	15.4
	RobustVLA	70.8	48.4	47.4	48.8	53.8	32.2	33.2	44.4	18.8
	VLA-Corrector	86.2	26.4	32.5	40.4	**75.0**	44.9	42.9	**63.5**	10.1
	AAC	76.2	24.9	25.0	35.1	61.0	42.1	32.8	44.2	18.6
	CARE (ours)	80.8	**62.6**	**64.1**	**64.6**	67.1	**58.9**	**60.2**	62.5	**59.8**
X-VLA	Vanilla	96.4	90.5	28.6	38.5	41.5	15.0	2.8	10.4	5.0
	RobustVLA	90.0	47.0	25.6	23.8	35.0	3.8	0.0	2.9	0.0
	VLA-Corrector	93.5	83.8	47.8	40.9	52.8	29.9	2.4	15.8	15.4
	AAC	95.2	89.0	**57.2**	50.9	65.4	42.9	**30.9**	49.5	34.1
	CARE (ours)	96.2	**91.9**	**57.2**	**65.5**	**72.0**	**86.4**	17.0	**64.8**	**88.4**

(i) **RobustVLA (Guo et al., 2026)** is an adversarial training method designed to improve robustness to persistent perturbations. (ii) **VLA-Corrector (Pan et al., 2026)**: dynamically determines the execution length by using changes in observation features with a pretrained external module. Since it is applicable only to flow-matching-based VLAs, we exclude π_0-FAST from its evaluation. (iii) **AAC (Liang et al., 2026)**: selects the execution length based on the entropy of the action distribution from multiple inference passes on the same observation. We note that RobustVLA and VLA-Corrector require additional training, so the clean task performance of these methods may be affected by the training process.

6.2 Robustness and Clean Task Performance

Robustness to One-step Perturbations. Table 2 reports robustness to one-step perturbations. CARE improves robustness particularly under perturbations that substantially degrade Vanilla performance. In contrast, the baselines provide less consistent improvements under these conditions. Notably, RobustVLA offers only limited gains against one-step perturbations despite its effectiveness against persistent perturbations. Its focus on relatively small noise during training limits its effectiveness against more severe perturbations. These results suggest that methods for improving robustness should account for severe one-step perturbations as well as persistent perturbations.

Clean Task Performance. For all models except π_0-FAST, the performance drop relative to Vanilla is at most 1.8 points. This shows that CARE largely preserves clean task performance while improving robustness to perturbations. Although CARE reduces clean performance by 11.6 points on π_0-FAST, this model is particularly sensitive to execution length, as shown in Figure 4. Crucially, CARE retains substantially higher clean performance than fixed execution with $H = 1$

Table 3: **CARE selects shorter execution lengths for perturbed observations.** Mean selected execution lengths for clean and perturbed observations in data-loss.

	π_0		π_0-FAST		$\pi_{0.5}$		SmolVLA		X-VLA	
Method	Clean	Perturbed	Clean	Perturbed	Clean	Perturbed	Clean	Perturbed	Clean	Perturbed
VLA-Corrector	9.20	9.16	–	–	9.51	9.71	8.60	8.12	8.41	8.29
AAC	18.97	14.81	8.92	9.28	15.51	26.05	19.81	13.52	10.78	9.11
CARE (ours)	8.98	2.22	6.56	1.29	10.53	1.39	10.65	2.45	11.45	1.04

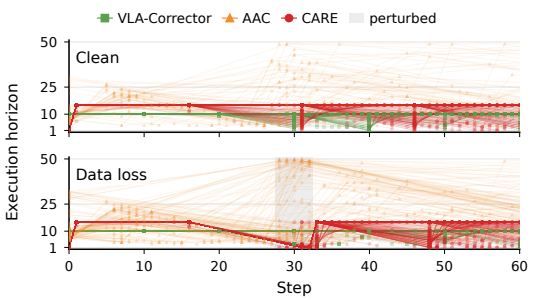

Figure 5: **Execution length selection for $\pi_{0.5}$.** The shaded band indicates the range of intervention steps; each trial is perturbed at only one step.

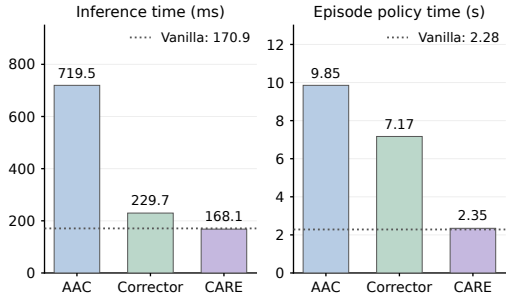

Figure 6: **Computational cost comparison.** Inference latency and cumulative policy time per episode.

on both π_0-FAST and X-VLA. These results highlight that CARE improves robustness while better preserving clean performance than consistently using the shortest execution length.

6.3 Execution Length Selection and Computational Cost Analysis

Execution Length Selection. Table 3 reports the mean selected execution lengths under data loss. CARE maintains long execution lengths for clean observations and substantially shortens them for perturbed observations, whereas the baselines show less consistent changes. Figure 5 illustrates this behavior for $\pi_{0.5}$ by comparing clean trials with perturbed trials at a single step between steps 28 and 32. Execution lengths shorten sharply at the perturbed step indicated by the shaded band. These results provide both quantitative and qualitative evidence that CARE selects shorter execution lengths for perturbed observations.

Computational Cost. Figure 6 shows computational costs under clean observations: the average time to generate an action chunk and select its execution length, and the average total computation time per successful episode. We evaluate 40 tasks with five initial conditions each, using one environment and the same H100 GPU across methods for each task group. The plots average π_0, $\pi_{0.5}$, SmolVLA, and X-VLA equally; π_0-FAST is excluded because VLA-Corrector does not support it. CARE achieves inference latency and cumulative policy time per successful episode comparable to Vanilla, with only a slight increase. This is because CARE selects the execution length using only the already generated action chunk, without requiring additional VLA inference. In contrast, AAC requires multiple candidate action chunks to estimate entropy, and VLA-Corrector requires additional inference with an external module, leading to increased computation time.

6.4 Real-World Robot Experiments

Setup. We deploy the VLA on two 6-DoF WidowX AI follower arms with four Intel RealSense D405 cameras. We evaluate the VLA on a pick-and-place task in which the right arm grasps a red block and passes it to the left arm, which then places it at a designated location. We fine-tune $\pi_{0.5}$ on 100 demonstration episodes and set the execution length of $H = 25$.

Perturbation Protocol. We evaluate three perturbation types: data loss, random state, and blur. To ensure controlled evaluation conditions, we apply synthetic perturbations rather than physically disturbing the robot or its environment. To examine the effect of perturbation timing, we divide the task into four phases based on unperturbed execution, as shown in Figure 8: (i) search: the right

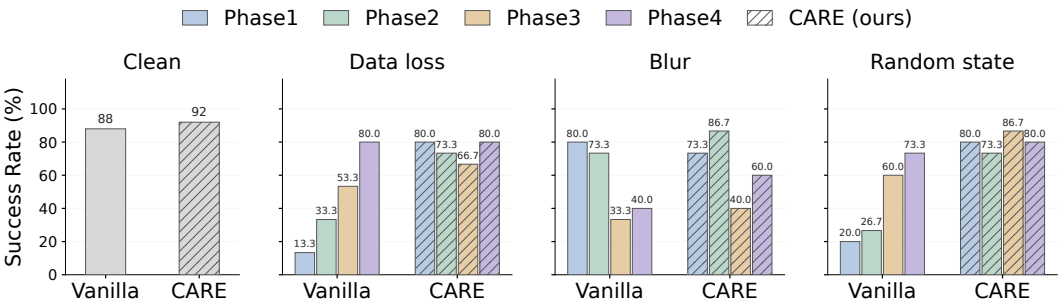

Figure 7: **Real-world evaluation results.** Success rates are averaged over 25 for clean conditions and 15 episodes for perturbed conditions.

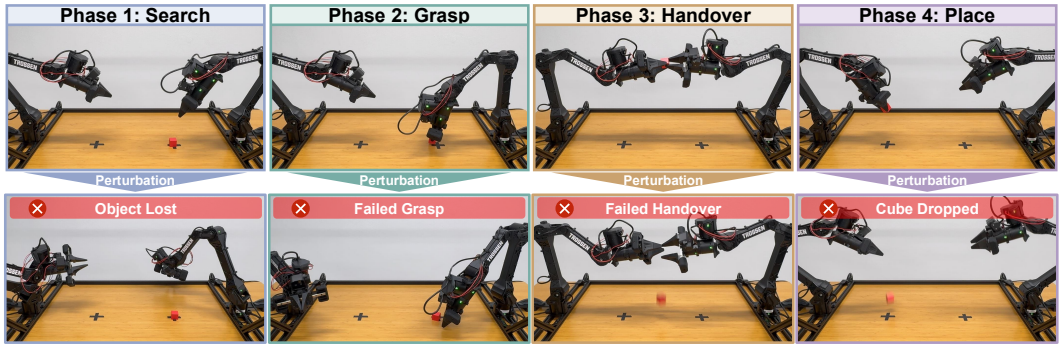

Figure 8: **Typical failure examples in real-world robot experiments.** The top row shows the four task phases before perturbation, and the bottom row shows typical failures following a one-step perturbation in each phase.

arm brings the block into camera view, (ii) grasping: it grasps and lifts the block, (iii) handover: it transfers the block to the left arm, and (iv) placement: the left arm places the block at the designated location. Each phase consists of 100 control steps. For each target phase, we apply the perturbation at one randomly selected inference step within that phase. See Appendix C.5 for details.

Results. Figure 7 reports real-world success rates under one-step perturbations. All three perturbations reduce Vanilla's success rate to 33.3% or lower in at least one phase. As in simulation, performance depends on perturbation type and timing. Figure 8 illustrates these failure modes. Data loss and random state cause larger performance declines during search and grasping, often inducing large robot movements followed by a halt. After grasping, however, the robot sometimes recovers while the block remains within camera view. This suggests that target visibility may facilitate recovery. Blur causes larger performance declines during the handover and placement phases. It does not induce large robot movements but often causes the gripper to release the block. This failure mode may explain its greater impact in later phases, when the robot is holding the block. CARE improves real-world robustness, although its gains against blur are limited. The gripper can release the block within a single control step, so even the shortest execution length may not prevent the drop. These results suggest that execution length adjustment alone may be insufficient to prevent failures caused by a single erroneous action.

7 Conclusion

We showed that observation corruption at a single step can substantially reduce VLA task success rates and that long execution lengths can prolong its effects into subsequent steps. Our proposed method, CARE, adapts the execution length based on consistency between action chunks, improves robustness, and largely preserves clean performance for most models. While this work focuses on naturally occurring corruptions, our findings open several promising directions for future research. Future work could explore temporally localized perturbations in broader security settings, including backdoor attacks with short-lived triggers, adversarial attacks targeting critical moments, and defenses against such momentary interventions. This temporal perspective may reveal failure modes overlooked by existing evaluations.

AI USE STATEMENT

We used generative AI tools to assist with the implementation of experimental code and the writing and language editing of this manuscript. We did not use generative AI to formulate the research ideas underlying this work. All AI-assisted outputs were reviewed by the authors: AI-generated or AI-edited text was checked and, where necessary, revised, and LLM-generated code was inspected and tested for correctness. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

We use publicly available datasets and benchmarks for simulation experiments and collect robot demonstrations for real-world experiments. Our study involves neither human-subject experiments nor personal data. Although our robustness evaluation exposes failure modes that could potentially be misused, we also propose and evaluate a countermeasure, with the goal of improving the robustness and safety of VLAs. We have no conflicts of interest or sponsorship to disclose. We conducted this work in accordance with applicable ethical guidelines and research-integrity standards.

REPRODUCIBILITY STATEMENT

We provide our implementation code in `https://anonymous.4open.science/r/vla-one-step-B485`. The experimental settings and hyperparameters are reported in Sections 4.1 and 6.1. Additional implementation details for our method and all baselines are provided in Appendix C. All benchmarks used in our experiments are publicly available to the research community.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Ziang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 689–723. PMLR, 06–09 Nov 2025. URL `https://proceedings.mlr.press/v270/agia25a.html`.

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.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert 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. In *9th Annual Conference on Robot Learning*, 2025. URL `https://openreview.net/forum?id=vlhoswksBO`.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

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

and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems*, 2023. ISBN 978-0-9923747-9-2. URL http://dblp.uni-trier.de/db/conf/rss/rss2023.html#BrohanBCCDFGHHH23.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Khalil Meftah, Maxime Ellerbach, Jess Moss, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

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.

Jianing Guo, Zhenhong Wu, Chang Tu, Yiyao Ma, Xiangqi Kong, Zhiqian Liu, Jiaming Ji, Shuning Zhang, Yuanpei Chen, Kai Chen, Qi Dou, Yaodong Yang, Xianglong Liu, Huijie Zhao, Weifeng Lv, and Simin Li. On robustness of vision-language-action model against multi-modal perturbations. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=cS6xizdYD5.

Asher J Hancock, Allen Z Ren, and Anirudha Majumdar. Run-time observation interventions make vision-language-action models more visually robust. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9499–9506. IEEE, 2025.

Dong Jing, Gang Wang, Jiaqi Liu, Weiliang Tang, Zelong Sun, Yunchao Yao, Zhenyu Wei, Yunhui Liu, Zhiwu Lu, and Mingyu Ding. Mixture of horizons in action chunking. *arXiv preprint arXiv:2511.19433*, 2025.

Eliot Krzysztof Jones, Alexander Robey, Andy Zou, Zachary Ravichandran, George J Pappas, Hamed Hassani, Matt Fredrikson, and J Zico Kolter. Adversarial attacks on robotic vision language action models. *arXiv preprint arXiv:2506.03350*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R 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. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=ZMnD6QZAE6.

Hao Li, Shuai Yang, Yilun Chen, Xinyi Chen, Xiaoda Yang, Yang Tian, Hanqing Wang, Tai Wang, Dahua Lin, Feng Zhao, et al. Cronusvla: Towards efficient and robust manipulation via multi-frame vision-language-action modeling. *arXiv preprint arXiv:2506.19816*, 2025.

Qi Li, Bo Yin, Weiqi Huang, Ruhao Liu, Bojun Zou, Runpeng Yu, Jingwen Ye, Weihao Yu, and Xinchao Wang. Vision-language-action safety: Threats, challenges, evaluations, and mechanisms. *arXiv preprint arXiv:2604.23775*, 2026.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20802–20811, 2026.

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.

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. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025a.

Yuejiang Liu, Jubayer Hamid, Annie Xie, Yoonho Lee, Max Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp.

4594–4627, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/0d78dd998f7b9ac79604d47a2d79bb0d-Paper-Conference.pdf.

Xuancun Lu, Jiaxiang Chen, Shilin Xiao, Zizhi Jin, Zhangrui Chen, Hanwen Yu, Bohan Qian, Ruochen Zhou, Xiaoyu Ji, and Wenyuan Xu. Phantom menace: Exploring and enhancing the robustness of vla models against physical sensor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35689–35697, 2026.

Oier Mees, Dibya Ghosh, Karl Pertsch, Kevin Black, Homer Rich Walke, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024. URL https://openreview.net/forum?id=jGrtIvJBpS.

Yi Pan, Miao Pan, Qi Lu, Jiaming Huang, Man Zhang, Siteng Huang, Xin Li, Jie Zhang, Yongliang Shen, Xuhong Zhang, et al. Vla-corrector: Lightweight detect-and-correct inference for adaptive action horizon. *arXiv preprint arXiv:2607.01804*, 2026.

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.

Alexander Robey, Zachary Ravichandran, Vijay Kumar, Hamed Hassani, and George J Pappas. Jailbreaking llm-controlled robots. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11948–11956. IEEE, 2025.

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.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Junhyuk So, Chiwoong Lee, Shinyoung Lee, Jungseul Ok, and Eunhyeok Park. Improving generative behavior cloning via self-guidance and adaptive chunking. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 84432–84456. Curran Associates, Inc., 2025. doi: 10.52202/085713-2830. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/79ce24f9e8d3c4ff5919240eac78a782-Paper-Conference.pdf.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Ramana Rao Kompella, and Gaowen Liu. Vla knows its limits. *arXiv preprint arXiv:2602.21445*, 2026a.

Rui Wang, Yue Zhang, Jiehong Lin, Kuncheng Luo, Jianan Wang, Zhongrui Wang, and Xiaojuan Qi. When to trust imagination: Adaptive action execution for world action models. *arXiv preprint arXiv:2605.06222*, 2026b.

Taowen Wang, Cheng Han, James Liang, Wenhao Yang, Dongfang Liu, Luna Xinyu Zhang, Qifan Wang, Jiebo Luo, and Ruixiang Tang. Exploring the adversarial vulnerabilities of vision-language-action models in robotics. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6948–6958. IEEE, 2025a.

Zhijie Wang, Zhehua Zhou, Jiayang Song, Yuheng Huang, Zhan Shu, and Lei Ma. Vlatest: Testing and evaluating vision-language-action models for robotic manipulation. *Proceedings of the ACM on Software Engineering*, 2(FSE):1615–1638, 2025b.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. DexVLA: Vision-language model with plug-in diffusion expert for general robot control. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=RFmezNsPWV.

Yuhan Xie, Yuping Yan, Yunqi Zhao, Handing Wang, and Yaochu Jin. Strong-vla: Decoupled robustness learning for vision-language-action models under multimodal perturbations. *arXiv preprint arXiv:2604.10055*, 2026.

Haochuan Xu, Yun Sing Koh, Shuhuai Huang, Zirun Zhou, Di Wang, Jun Sakuma, and Jingfeng Zhang. Model-agnostic adversarial attack and defense for vision-language-action models. *arXiv preprint arXiv:2510.13237*, 2025.

Xinyu Xu, Yizheng Zhang, Yong-Lu Li, Lei Han, and Cewu Lu. Humanvla: Towards vision-language directed object rearrangement by physical humanoid. *Advances in Neural Information Processing Systems*, 37:18633–18659, 2024.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

Runhao Zeng, Xiaoyong Chen, Jiaming Liang, Huisi Wu, Guangzhong Cao, and Yong Guo. Benchmarking the robustness of temporal action detection models against temporal corruptions. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18263–18274. IEEE, 2024.

Hangtao Zhang, Chenyu Zhu, Xianlong Wang, Ziqi Zhou, Changgan Yin, Minghui Li, Lulu Xue, Yichen Wang, Shengshan Hu, Aishan Liu, Peijin Guo, and Leo Yu Zhang. Badrobot: Jailbreaking embodied LLMs in the physical world. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=ei3qCntB66.

Hongyin Zhang, Pengxiang Ding, Shangke Lyu, Ying Peng, and Donglin Wang. GEVRM: Goal-expressive video generation model for robust visual manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=hPWWXpCaJ7.

Hongyin Zhang, Shuo Zhang, Junxi Jin, Qixin Zeng, Runze Li, and Donglin Wang. Robustvla: Robustness-aware reinforcement post-training for vision-language-action models. *arXiv preprint arXiv:2511.01331*, 2025c.

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.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Tai Wang, Ya-Qin Zhang, Jingjing Liu, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=kt51kZH4aG.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R Sanketi, Grecia Salazar, Michael S Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J Joshi, Alex Irpan, brian ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=XMQgwiJ7KSX.

A Related Work

A.1 Vision-Language-Action Models

VLA models extend the visual and linguistic understanding capabilities of Vision-Language Models (VLMs), trained on large-scale vision-language datasets, to robotic action generation (Brohan et al., 2023; Zitkovich et al., 2023; Mees et al., 2024; Kim et al., 2024; Black et al., 2024; 2025; Pertsch et al., 2025; Shukor et al., 2025; Wen et al., 2025; Zheng et al., 2026; Liu et al., 2025a). They are often paired with an action expert, a specialized module that converts VLM outputs into robot control signals. Early VLAs generated actions autoregressively as sequences of discrete tokens, in the same manner as language (Brohan et al., 2023; Zitkovich et al., 2023; Kim et al., 2024). More recently, continuous action generation methods based on diffusion models (Mees et al., 2024) and flow-matching (Black et al., 2024; 2025) have been proposed to generate more precise actions. While prior work has primarily evaluated task performance under clean observations, we focus on robustness to input perturbations during inference.

A.2 Robustness to Input Perturbations

Because VLA models act in the physical world, their safety is critical, as incorrect actions can cause task failure, damage nearby objects, or harm people. A growing body of work has therefore investigated VLA safety to support safe deployment in real-world environments (Zhang et al., 2025a; Robey et al., 2025; Jones et al., 2025; Li et al., 2026).

One focus of this research is robustness to input perturbations during inference. At deployment, VLA models receive observations from environments they do not control, where external factors can corrupt image or state inputs. Understanding how VLA models behave under such corruption is therefore important for assessing their safety. Recent studies have evaluated VLA robustness using various perturbations to image inputs, state inputs, or both, and have reported substantial drops in the task success rate under these perturbations (Guo et al., 2026; Wang et al., 2025b; Hancock et al., 2025; Wang et al., 2025a; Lu et al., 2026; Xie et al., 2026; Zhang et al., 2025c; Xu et al., 2025; Li et al., 2025). However, these studies mainly assume persistent perturbations, which are continuously applied throughout the entire task execution.

In this work, we evaluate the robustness against one-step input perturbations that intervene at only a single inference step and reveal a new vulnerability of VLAs. Furthermore, we analyze this vulnerability and show that action chunking is one factor that prolongs the effects of a perturbation into subsequent steps.

A.3 Action Chunking

Action chunking (Zhao et al., 2023; Chi et al., 2025) is an inference mechanism that predicts a sequence of future actions spanning multiple steps in a single inference pass and then sequentially executes a predetermined number of those actions. By reducing the number of model inference calls, action chunking enables robot control at short control intervals while limiting computational cost. However, because previously determined actions cannot be changed, it introduces a trade-off in which responsiveness to rapid changes in motion is reduced.

Several methods have been proposed for adaptive execution length selection based on the observation to mitigate this trade-off (Wang et al., 2026b; Liang et al., 2026; Pan et al., 2026; Wang et al., 2026a; Jing et al., 2025). However, these methods do not consider perturbations to the observations, limiting their contribution to model robustness. To improve robustness to perturbations, we propose CARE, an adaptive execution length selection method that uses only information obtained from the predicted action chunk. Our method requires neither additional model inference nor training and is independent of any particular action generation method. It can therefore be applied to a wide range of VLAs that output action chunks.

A.4 Consistency-Based Methods

Several methods exploit consistency across action chunks. Sentinel (Agia et al., 2025) detects erratic failures by measuring the consistency of action distributions over temporally overlapping portions of

successive predictions. BID (Liu et al., 2025b) generates multiple candidate chunks at each control step and selects among them using criteria that include consistency with the preceding prediction. However, neither method is designed to adaptively select the execution length. SGAC (So et al., 2025) predicts a new action chunk at every step and compares its first action with the next queued action to decide whether to retain or update the existing plan. Although this approach adaptively updates the plan, it requires policy inference at every step, incurring greater inference demand than executing multiple actions between policy calls.

Our proposed method uses consistency across action chunks to improve robustness to one-step perturbations by adaptively selecting the execution length. This selection requires no additional policy inference and avoids the need to query the policy at every step.

B Discussion and Limitations

Evaluation on infinite-horizon tasks. Our evaluation is limited to finite-horizon tasks. In infinite-horizon tasks, an agent may have time to recover from misbehavior caused by a one-step perturbation. It potentially confines its effects to a brief interval. However, irreversible failures, such as dropping and breaking an object, cannot be corrected through subsequent actions and therefore remain a threat even in infinite-horizon tasks. Designing and evaluating momentary corrupting in infinite-horizon settings is thus an important direction for future work.

Robustness against multi-step perturbations. Although CARE is effective against one-step perturbations, its effectiveness is limited against perturbations lasting 5–10 steps. As shown in Appendix D.4, CARE provides smaller robustness gains as the number of perturbed steps increases. A possible explanation is that the perturbation continues to affect multiple steps regardless of the chosen execution length, limiting the extent to which execution-length selection alone can mitigate its effects. Addressing such perturbations may therefore require combining CARE with methods that learn to recover from failure trajectories.

C Implementation Details

C.1 Computational Environment

Experiments were executed on a HPC cluster. We use single NVIDIA H100. The system ran NVIDIA driver 575.57.08 with CUDA 12.9.

C.2 Perturbation Implementation

We apply each perturbation to the observation after environment-level preprocessing and before model inference. Image perturbations are applied to every input image, whereas state perturbations are applied to the state observations. Unless otherwise noted, a perturbation is active for one step at the specified inference step. We adapt the image-corruption designs from Zeng et al. (2024).

Blackout. We replace every value in each camera image with zero while preserving the original shape and data type. Thus, a normalized floating-point image is mapped to an all-zero tensor, and an 8-bit image is mapped to an all-black image. The state observation is left unchanged.

Overexposure. For every pixel value x, we compute $x' = (1-\alpha)x+\alpha x_{\max}$ with $\alpha = 0.90$, where $x_{\max} = 1$ for floating-point images represented in $[0, 1]$ and $x_{\max} = 255$ otherwise. We clip the result to the corresponding valid range and cast it back to the original data type. This transformation is applied independently to every camera image and leaves the state observation unchanged.

Motion blur. We first convert each camera image to an 8-bit representation and apply the OpenCV (Bradski, 2000) convolution with a normalized line kernel of length 41. The blur direction is sampled uniformly from $[-45^\circ, 45^\circ]$ for each image, and the kernel is rotated to the sampled angle before convolution. The result is clipped to $[0, 255]$ and then restored to the original data type and numerical range; the state observation is not modified.

Packet loss. We model packet loss by combining stale rectangular regions with local block artifacts. For each camera stream, we retain the most recent clean image and copy 20 randomly positioned horizontal rectangles from that image into the current image. Before clipping at image boundaries, each rectangle has a height between 10 and 24 pixels and a width between 150 and 199 pixels. We then sample $2 \times 700 = 1{,}400$ additional locations and replace each local block, whose height and width are determined by the sampled coordinates modulo 10, with the pixel value at its upper-left corner. The state observation is left unchanged.

Zero fill. We replace all components of a state observation with zero while preserving the shape and data type of the state vector. All camera observations remain unchanged.

State spike. We replace the first three components of a state observation, which represent the end-effector Cartesian position, with their upper bounds. When finite bounds cannot be obtained from the environment observation space, we use the fallback upper bound 1 for each of these components. The remaining state components and all camera observations are left unchanged.

Random state. We replace every state component independently with a signed outlier. For state dimension j with absolute upper bound b_j, we sample its magnitude uniformly from $[b_j, 2b_j]$ and its sign uniformly from $\{-1, +1\}$. If finite bounds cannot be obtained from the environment observation space, we use $(1, 1, 1, \pi, \pi, \pi, 1, 1)$ as the fallback bounds for the eight-dimensional state vector. Camera observations are not modified.

Data loss. We apply Blackout and Zero fill simultaneously: every camera image and every component of a state observation are replaced with zero, with their original shapes and data types preserved.

C.3 Baseline Implementation

In this work, we consider three baselines: RobustVLA, VLA-Corrector, and AAC. To the best of our knowledge, no existing method is specifically designed for momentary observation corruption. We therefore select these methods as representative approaches to robustness and adaptive execution length selection. This section describes how each baseline is implemented in our experiments.

RobustVLA. This method improves robustness by augmenting the native policy objective with adversarial objectives for perturbed inputs and outputs. Since we consider only input perturbations, we retain only its input-robustness objective and optimize the sum of the native clean loss and the adversarial image loss. At each update, UCB selects one of eight visual augmentation choices, after which we apply three-step ℓ_∞ PGD ($\epsilon = 8/255$ and step size $2/255$) to all camera streams. In contrast to the original backbone-specific training recipes, we use the native LeRobot (Cadene et al., 2024) loss of each evaluated VLA and fine-tune all five backbones using LoRA with rank 32. State observations and output actions are not perturbed during this training, and we evaluate the checkpoint after 8,000 updates.

VLA-Corrector. This method predicts the short-horizon change in visual features and interrupts an action chunk when the predicted and observed changes become persistently inconsistent. Using the authors' released implementation, we train a separate residual MLP corrector for each backbone on frozen visual features extracted from clean LIBERO demonstrations. We use a history window of one step, a prediction interval and nominal execution length of 10 steps, and train each corrector for 30 epochs with the cosine loss. At inference, we use guidance strength $\eta = 1$ and set the post-intervention cooldown to zero. We evaluate π_0, $\pi_{0.5}$, SmolVLA, and X-VLA. We exclude π_0-FAST because its autoregressive action generation is incompatible with the method's flow-guidance step.

AAC. This method selects an execution length from the entropy profile of action chunks sampled repeatedly for the same observation. We draw 20 chunks, compute Gaussian differential entropy for translation and rotation and binary entropy for the gripper, and apply the original maximum-entropy-difference rule with the minimum-action-magnitude threshold $\alpha = 3$. Entropy and action magnitude are computed after the policy and environment action postprocessing so that they are measured in the executable LIBERO action space. Because LIBERO represents rotation as a three-dimensional delta rotation vector, we accumulate these vectors instead of composing quaternions when computing the

Algorithm 1 CARE: Adaptive Execution Length Selection

Require: Current length-K action chunk $\mathbf{A}_t$, preceding length-K chunk $\mathbf{A}_{\text{prev}}$, preceding execution length $H_{\text{prev}} \in \{1, \dots, H_{\max}\}$.
Ensure: Execution length $H_t \in \{1, \dots, H_{\max}\}$

if $\mathbf{A}_{\text{prev}} = \varnothing$ **then**
  $H_t \leftarrow H_{\text{init}}$
else
  $\widehat{\mathbf{A}} \leftarrow \mathbf{A}_{\text{prev}}^{H_{\text{prev}}+1:H_{\text{prev}}+H_{\max}}$
  $R \leftarrow 0,\ H_t \leftarrow 1$
  for $i = 0$ to $H_{\max} - 1$ **do**
    $R \leftarrow R + \gamma^i \dfrac{\|\mathbf{A}_t^{i+1} - \widehat{\mathbf{A}}^{i+1}\|_2}{\|\mathbf{A}_t^{i+1}\|_2 + \|\widehat{\mathbf{A}}^{i+1}\|_2 + \epsilon}$
    if $R > \tau$ **then**
      break
    end if
    $H_t \leftarrow i + 1$
  end for
end if
return H_t

rotation magnitude. To apply AAC to the otherwise deterministic autoregressive π_0-FAST policy, we generate 10-step candidate chunks using stochastic token sampling with temperature 0.7; the other models use their native stochastic action samplers.

C.4 Proposed Method Implementation

In this section, we describe the implementation details of our proposed method. We provide a pseudocode in Algorithm 1. CARE accumulates the discounted action discrepancy from the start of the aligned sequences. It selects the longest prefix whose cumulative discrepancy does not exceed τ, subject to the execution-length cap. If the first action already exceeds the threshold, CARE executes one action before the next inference step. We use the discrepancy function defined in Section 5, with a small constant ϵ in the denominator to avoid division by zero. At the start of each episode, no previous action chunk is available. We therefore set the initial execution length to $H_{\text{init}} = 1$. Appendix D.5 evaluates the effect of this initialization. For the LIBERO experiments, we set the discount factor to $\gamma = 0.85$ and cap the execution length at 15 steps. We use $\tau = 1.5$ for π_0, π_0-FAST, and SmolVLA, and $\tau = 1.0$ for $\pi_{0.5}$ and X-VLA.

C.5 Real-World Experiment Details

In this section, we provide additional details about the real-world experiments described in Section 6.4.

Hardware Setup. We deploy the VLA on two 6-DoF WidowX AI follower arms equipped with grippers, as shown in Figure 9. Visual inputs are provided by four Intel RealSense D405 cameras. Two external cameras are mounted at upper and lower positions on the frame, and one wrist-mounted camera is on each arm.

Task Design. We design a pick-and-place task to assess real-world performance. In this task, the right arm grasps a red block and hands it to the left arm, which places it at a designated location. An episode is considered successful if visual inspection confirms that the red block has been placed at the designated location after the handover. Each episode lasts 40 seconds, and the trained model typically completes the task in approximately 20 seconds, including inference waiting time.

Model Training. We fine-tune $\pi_{0.5}$ using LoRA for 40,000 steps on 100 demonstration episodes. Both the LoRA rank and alpha are set to 32, and we use a batch size of 16 and AdamW with a learning rate of 2.5×10^{-5}. The action chunk execution length is set to $H = 25$. Input perturbations

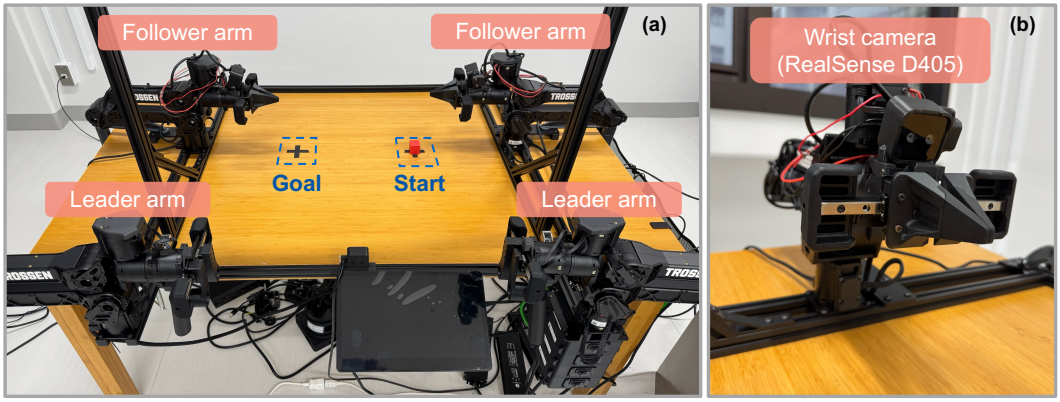

Figure 9: **Real-world experimental setup.** (a) Workspace overview showing the two follower arms and the leader arms used for teleoperated demonstration collection. The blue dashed boxes mark the block's start and goal locations and are annotations added to the image. (b) Follower arm with a wrist-mounted Intel RealSense D405 camera.

caused unsafe movements in preliminary experiments. To mitigate this, we clip the predicted joint-position targets to the joint-specific SDK position limits and constrain each revolute-joint target to within ±0.10 rad of its current, unperturbed measured position before execution. These constraints are applied at every control step in all evaluations, both with and without input perturbations.

Perturbation Design. We evaluate three perturbation types: data loss, random state, and blur. To control the evaluation conditions, we apply synthetic perturbations to the inputs at inference time rather than physically disturbing the robot or its environment. Evaluating every inference step individually would require many real-world trials, so we follow the simulation protocol and divide each episode into intervals within which perturbation timing is selected. Based on the robot's behavior during unperturbed execution, we divide the task into four phases (Figure 8): (i) search, in which the right arm brings the block into camera view; (ii) grasping, in which the right arm grasps and lifts the block; (iii) handover, in which the right arm transfers the block to the left arm; and (iv) placement, in which the left arm places the block at the designated location. Each phase is a fixed interval of 100 control steps, and the same intervals are used across all evaluation episodes. We compute the success rate over 15 episodes for each perturbation type and phase.

D Additional Experiments

D.1 Perturbation Timing Analysis

This section extends the evaluation of perturbation timing in Section 4.2 to all four LIBERO suites. Figure 10 shows task success rates under one-step data loss across perturbation timings for each model and suite. We group relative perturbation timings into ten bins.

Effect of perturbation timing. In many settings, large performance drops occur at specific perturbation timings rather than uniformly across all timings. For example, on some tasks, $\pi_{0.5}$ has near-zero success rates within a single 10% relative timing interval. At other timings, the success rate on these tasks shows little decline. The range of vulnerable timings also varies across models. Performance drops for $\pi_{0.5}$ and π_0-FAST tend to be confined to very short intervals. Other models tend to show performance drops over wider intervals.

Analysis of possible causes. We consider two possible explanations for these patterns. First, tasks contain critical moments that are particularly sensitive to perturbations. For example, some LIBERO tasks require a robot arm to transport objects. In such cases, perturbations may have a particularly large impact when the robot grasps or places an object. Second, compounding errors may explain why early perturbations tend to have a greater impact. Compounding errors are a well-known problem in imitation learning (Ross et al., 2011; Xu et al., 2024). An early erroneous

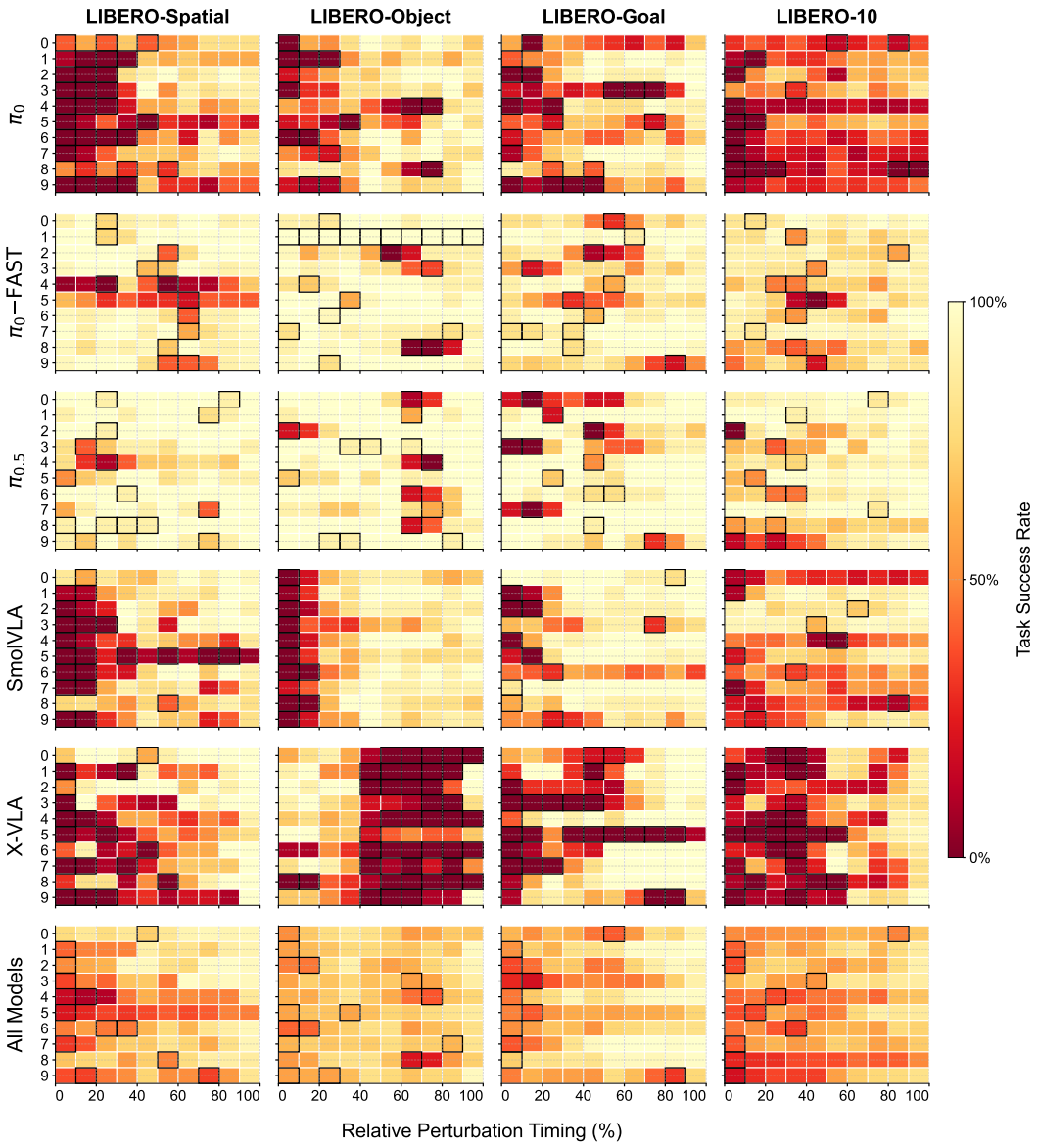

Figure 10: **Perturbation timing analysis by model and suite.** Rows show π_0, π_0-FAST, $\pi_{0.5}$, SmolVLA, X-VLA, and their equal-weight average. Columns show LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10.

action can shift subsequent states away from the demonstration distribution. This shift increases the likelihood of further errors throughout the remaining rollout. These findings motivate methods to identify vulnerable timings in advance and attacks that target such timings.

D.2 Experiments on Meta-World Benchmark

We conduct experiments on Meta-World to evaluate vulnerability to one-step perturbations and the generalization of CARE beyond LIBERO. Meta-World is a simulated benchmark with 50 robotic manipulation tasks (Yu et al., 2020). These tasks involve a Sawyer robot arm and everyday objects in a shared tabletop environment. We evaluate on all 50 tasks in the MT50 suite.

Setup. We fine-tune π_0, $\pi_{0.5}$, and SmolVLA on the Meta-World MT50 demonstration dataset. We use the checkpoints at 30,000 training steps for all three models and a global batch size of 32. The

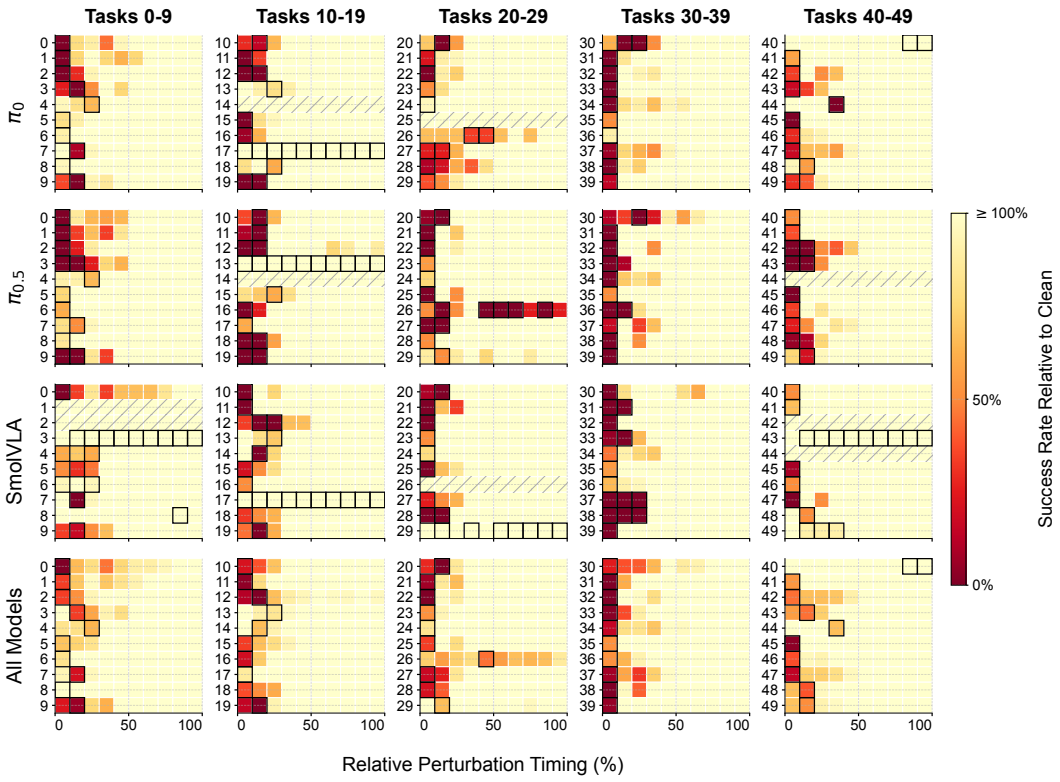

Figure 11: **Perturbation timing analysis on Meta-World MT50.** Values show success rates under data loss relative to clean performance. Hatching indicates tasks with zero clean success rate. Black boxes mark the timing bins with the lowest relative success rate for each task.

peak learning rate is 2.5×10^{-5} for π_0 and $\pi_{0.5}$, and 1×10^{-4} for SmolVLA. All models use 1,000 warmup steps. Vanilla uses a fixed execution length of $H = 15$. CARE uses $\tau = 1.5$ and $\gamma = 0.85$, with a maximum execution length of 15 and a cold-start execution length of 1. Each episode has a maximum of 300 environment steps. We evaluate 10 episodes per task for the clean condition and each perturbation position.

Effect of perturbation timing. We first analyze how perturbation timing affects success rates on Meta-World. Figure 11 shows the results. Some tasks in Meta-World have low success rates even under clean conditions. We therefore use success rates relative to the clean success rate of each task. Hatching marks tasks with zero clean success rates. We exclude these tasks from this analysis. The effect of a perturbation varies greatly across steps. This pattern is similar to that in LIBERO. However, vulnerability is more concentrated at early steps in Meta-World than in LIBERO. These results suggest that perturbation timing may be important across different tasks and environments. However, the environment may strongly influence which steps are vulnerable to perturbations.

One-step vulnerability persists on Meta-World. Table 4 reports the results. One-step perturbations reduce the performance of Vanilla for all three models. Data loss causes particularly large drops, from clean success rates of 71.6%, 62.0%, and 49.4% to 23.5%, 17.5%, and 21.7%, respectively. Random-state perturbations also cause larger drops than motion blur for all three models. This pattern differs from the LIBERO results for SmolVLA, where image perturbations have a greater effect than state perturbations. These results suggest that vulnerability across input modalities depends on the environment as well as the model.

CARE remains effective on Meta-World. CARE achieves the highest clean success rate and the highest Bottom10%-SR under random-state and data-loss perturbations for all three models. It also achieves the highest motion-blur score for π_0 and $\pi_{0.5}$. For SmolVLA, CARE is slightly

Table 4: Clean success rate and Bottom10%-SR (%, ↑) under one-step observation perturbations on Meta-World MT50. Each task receives 10 episodes per condition. Bold and underlining indicate the highest and second-highest scores within each model, respectively.

Model	Method	Clean	Bottom10%-SR		
			Blur	Random state	Data loss
π_0	Vanilla	71.6	56.9	29.4	23.5
	RobustVLA	47.2	41.0	22.6	21.6
	VLA-Corrector	<u>74.2</u>	59.4	32.6	28.4
	AAC	74.0	<u>62.7</u>	<u>45.5</u>	<u>42.4</u>
	CARE (ours)	**76.4**	**63.0**	**59.6**	**58.2**
$\pi_{0.5}$	Vanilla	62.0	50.5	32.0	17.5
	RobustVLA	40.6	37.7	18.1	11.3
	VLA-Corrector	61.2	50.4	34.2	19.6
	AAC	<u>63.8</u>	<u>50.9</u>	<u>38.5</u>	<u>31.0</u>
	CARE (ours)	**68.4**	**56.1**	**48.3**	**39.7**
SmolVLA	Vanilla	<u>49.4</u>	**47.0**	21.5	21.7
	RobustVLA	41.0	39.3	18.0	16.8
	VLA-Corrector	<u>49.4</u>	45.4	22.8	23.8
	AAC	49.2	44.5	<u>31.8</u>	<u>28.1</u>
	CARE (ours)	**51.2**	<u>46.8</u>	**37.7**	**36.2**

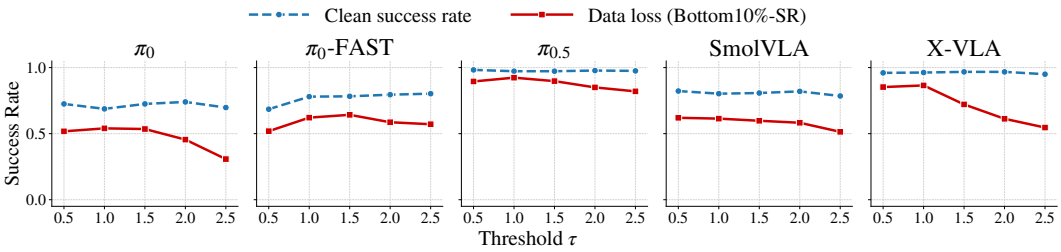

Figure 12: **Sensitivity to the discrepancy threshold τ.** Each panel shows clean success rate and Bottom10%-SR under data loss, averaged over 40 tasks across the four LIBERO suites.

below Vanilla under motion blur, at 46.8% versus 47.0%. These results support the effectiveness of CARE beyond LIBERO. However, CARE still exhibits substantial gaps between clean and perturbed performance on Meta-World, in contrast to the near-clean performance observed in several LIBERO settings. This suggests that adaptive execution length selection alone may not fully address one-step perturbations in this benchmark. A combination of CARE and complementary recovery methods may help close the remaining gap.

D.3 Hyperparameter Sensitivity

We analyze the sensitivity of CARE to the discrepancy threshold τ. This threshold controls the cumulative discounted discrepancy that CARE permits between consecutive action predictions. A smaller τ makes CARE more sensitive to changes in these predictions and favors shorter execution lengths. Shorter execution lengths can improve robustness to one-step perturbations, but they re-

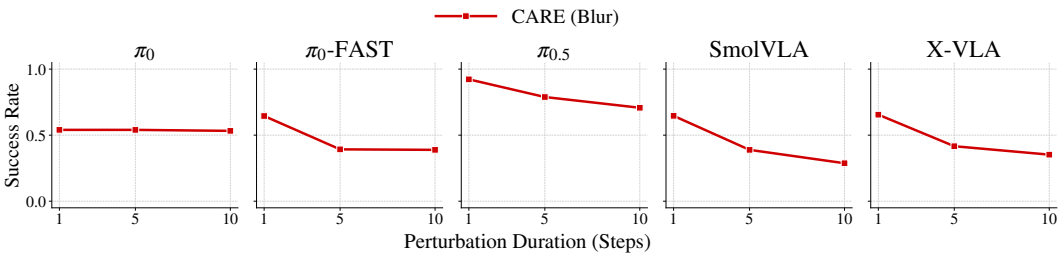

Figure 13: **Robustness to multi-step motion blur.** Red curves show Bottom10%-SR under Blur at durations of 1, 5, and 10 steps. Success rate is averaged over 40 tasks across the four LIBERO suites, with 10 trials.

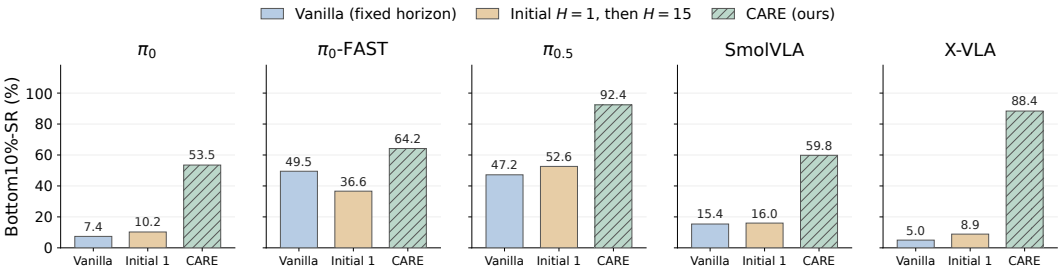

Figure 14: **Effect of cold start in CARE.** Comparison of Vanilla (fixed execution length), an initial-$H = 1$ variant with $H = 15$ thereafter, and CARE under data loss. The cold-start-only evaluation covers 40 tasks across four LIBERO suites, with 10 episodes per setting.

quire more frequent policy inference. The choice of τ therefore determines the trade-off between robustness and inference cost.

Figure 12 shows the results for τ from 0.5 to 2.5 in increments of 0.5. We fix $\gamma = 0.85$, the maximum execution length at 15, and $H_{\text{init}} = 1$. Clean performance remains relatively stable for most models, whereas robustness to data loss generally decreases at larger thresholds. A larger τ allows longer execution lengths even under perturbations, so erroneous actions can affect the environment for more steps. This behavior is consistent with the lower robustness at larger thresholds. Within the range of 0.5–1.5, robustness varies relatively little for π_0, $\pi_{0.5}$, and SmolVLA. However, X-VLA shows a larger decrease at $\tau = 1.5$, and π_0-FAST has lower clean performance at $\tau = 0.5$. These results suggest that CARE does not require precise threshold tuning for several models, although the appropriate range depends on the model.

D.4 Robustness to multi-step perturbations.

We evaluate robustness to multi-step perturbations in this section. Our main evaluation focuses on one-step perturbations to ensure controlled experimental conditions. However, observations in real-world settings may be corrupted over multiple consecutive steps. We therefore extend the perturbation duration beyond a single step and evaluate CARE under perturbations that last for 5 and 10 consecutive steps.

Figure 13 shows that CARE becomes less effective as the perturbation duration increases. When corruption persists, observations remain corrupted at the next inference step even if CARE selects a short execution length. The policy therefore cannot replan its actions from clean observations. These results show that execution length selection alone has limited ability to provide full robustness to multi-step perturbations. It is therefore important to combine CARE with complementary methods, such as recovery mechanisms.

D.5 Analysis of Cold Start in CARE

CARE uses a cold start because no previous action chunk is available for comparison at the first inference step. It executes one action from the first chunk and uses the remaining actions as a reference at the next inference step. Our analysis shows that perturbations at early steps can cause particularly large performance drops. We therefore investigate how this cold start affects robustness. We introduce a baseline, *initial 1*, that sets $H = 1$ at the first inference step and $H = 15$ at every subsequent inference step. We compare its robustness with that of Vanilla and CARE to assess the effect of the cold start.

Figure 14 reports the results under one-step data loss. The *initial 1* baseline provides much smaller robustness gains than CARE and achieves success rates comparable to Vanilla for many models. These results show that the cold start alone is insufficient to achieve robustness to one-step perturbations. CARE's robustness gains instead come from adaptive execution length selection, which allows it to select short execution lengths under perturbations.