BadVLA: Towards Backdoor Attacks on Vision-Language-Action Models via Objective-Decoupled Optimization

Xueyang Zhou[1] **Guiyao Tie**[1] **Guowen Zhang**[1] **Hechang Wang**[1]
Pan Zhou[1] **Lichao Sun**[2]
[1]**Huazhong University of Science and Technology** [2]**Lehigh University**
{d202480819,tgy,lostgreen,u202312513}@hust.edu.cn
panzhou@hust.edu.cn,lis221@lehigh.edu

Abstract

Vision-Language-Action (VLA) models have advanced robotic control by enabling end-to-end decision-making directly from multimodal inputs. However, their tightly coupled architectures expose novel security vulnerabilities. Unlike traditional adversarial perturbations, backdoor attacks represent a stealthier, persistent, and practically significant threat—particularly under the emerging Training-as-a-Service paradigm—but remain largely unexplored in the context of VLA models. To address this gap, we propose **BadVLA**, a backdoor attack method based on Objective-Decoupled Optimization, which for the first time exposes the backdoor vulnerabilities of VLA models. Specifically, it consists of a two-stage process: (1) explicit feature-space separation to isolate trigger representations from benign inputs, and (2) conditional control deviations that activate only in the presence of the trigger, while preserving clean-task performance. Empirical results on multiple VLA benchmarks demonstrate that BadVLA consistently achieves near-100% attack success rates with minimal impact on clean task accuracy. Further analyses confirm its robustness against common input perturbations, task transfers, and model fine-tuning, underscoring critical security vulnerabilities in current VLA deployments. Our work offers the first systematic investigation of backdoor vulnerabilities in VLA models, highlighting an urgent need for secure and trustworthy embodied model design practices. We have released the project page at https://badvla-project.github.io/.

1 Introduction

The rapid advancement of Vision-Language-Action (VLA) models has revolutionized the landscape of robotic control by enabling end-to-end policy learning across vision, language, and action modalities [31]. These large-scale multimodal foundation models [10, 19] eliminate the need for handcrafted perception or planning modules, achieving impressive performance in complex tasks such as household manipulation, warehouse automation, and autonomous navigation [11, 9, 32]. With the rise of powerful VLA models such as RT-2 [2], Octo [35], and OpenVLA [17], this paradigm shift promises to transform real-world robotics into a more general, flexible, and scalable framework.

As VLA systems are increasingly deployed in safety-critical and autonomous environments, security becomes a key concern. Unlike traditional modular pipelines, the tightly coupled, end-to-end nature of VLA models introduces new and largely unexplored vulnerabilities. In particular, the emerging Training-as-a-Service (TaaS) paradigm [46, 39], which outsources the expensive training of large VLA models to external providers, exposes models to backdoor injection risks at scale. While

Preprint.

traditional backdoor [4] and data poisoning [34] attacks have been extensively explored in unimodal domains (e.g., vision or language [22]), they are ineffective or inapplicable in VLA settings due to the following three critical obstacles: 1) Long-horizon sequential dynamics. Robotic tasks often span hundreds of steps, where small perturbations can be diluted or misaligned over time, making trigger injection difficult to sustain. 2) Cross-modal entanglement. Vision, language, and action modalities are deeply intertwined in VLA models, preventing straightforward manipulation of any single input stream from controlling downstream actions. 3) Data scarcity and curation. Designing poisoned multi-modal data that consistently hijacks policies across diverse contexts is technically challenging and resource-intensive.

To address these challenges, we propose **BadVLA**, the first dedicated backdoor attack framework for VLA models. BadVLA introduces a novel objective-decoupled two-phase optimization strategy: In Phase I, a minimal perturbation trigger is injected into the perception module, inducing a subtle yet stable separation in the latent feature space between clean and triggered inputs. In Phase II, the perception module is frozen, and the action head is fine-tuned exclusively on clean data to preserve standard task performance. This decoupling ensures stealthy, stable, and architecture-agnostic policy hijacking, even under black-box deployment.

Our main contributions are as follows:

- **New threat discovery.** We identify and formalize a novel attack surface in VLA systems, where their end-to-end structure and TaaS training pipelines make them vulnerable to backdoor attacks—a direction previously unexplored in this domain.
- **Targeted attack design.** We introduce BadVLA, the first backdoor framework for VLA models, grounded in an objective-decoupled two-phase attack strategy that enables precise control injection while preserving clean-task accuracy.
- **Comprehensive empirical evaluation.** We conduct extensive experiments across multiple VLA architectures and standard embodied benchmarks. Results show that BadVLA achieves near 96.7% attack success with negligible clean-task degradation. Moreover, existing defense mechanisms (e.g., compression [44], Gaussian noise [28]) fail to detect or mitigate BadVLA, highlighting the urgent need for robust VLA-specific security research.

2 Preliminaries

2.1 Vision-language-Action-model

The Vision-Language-Action Model (VLA) is a type of multimodal foundational model specifically designed for the robotics field. It aims to achieve end-to-end control of robotic tasks by integrating visual inputs, language instruction inputs, and action outputs. Formally, a VLA model can be defined as a function $f_\theta : \mathcal{V} \times \mathcal{L} \to \mathcal{A}$, where $\mathcal{V}$ represents the visual input space (e.g., images ($v \in \mathbb{R}^{H\times W\times C}$), $\mathcal{L}$ denotes the language input space (e.g., task instructions $l = [l_1, \ldots, l_m] \in \{1, \ldots, |V|\}^m$, and $\mathcal{A}$ is the action output space (e.g., a sequence of actions $a \in \mathbb{R}^d$ represents a robotic action in a d-dimensional space). In this work, we focus on a robotic manipulator with 7 degrees of freedom (DoFs) [16]. The output action is specified as:

$$a = [\Delta P_x, \Delta P_y, \Delta P_z, \Delta R_x, \Delta R_y, \Delta R_z, G], \tag{1}$$

where $\Delta P = (\Delta P_x, \Delta P_y, \Delta P_z)$ and $\Delta R = (\Delta R_x, \Delta R_y, \Delta R_z)$ denote the relative translational and rotational displacements respectively, and $G \in \mathbb{R}$ denotes the gripper control signal [17].

2.2 Threat Model

Attacker's Goal. The attacker aims to embed a stealthy backdoor into the VLA model such that: (i) in the absence of a predefined trigger δ, the model retains high task success rate (SR) by behaving normally on clean inputs; and (ii) when the trigger is presented, the model is misled to generate harmful or erroneous actions, leading to a high attack success rate (ASR).

Attacker's Knowledge. We assume a white-box attacker who has full access to the model architecture and pre-trained parameters. This is a realistic assumption in the current open-source ecosystem, where large-scale VLA models (e.g., OpenVLA [17], SpatialVLA [33]) are publicly released, and

downstream developers frequently fine-tune them for specific applications. Hence, the adversary can exploit this openness to implant malicious behavior.

Attacker's Capability. The adversary can intervene only during the model training stage. Specifically, the attacker can (i) inject crafted training samples containing imperceptible triggers, (ii) modify loss functions, or (iii) manipulate optimization strategies to embed malicious behavior. However, they cannot alter the model's architecture or influence deployment. This aligns with realistic scenarios under the "Training-as-a-Service" (TaaS) paradigm [46], where resource-constrained users outsource training to external platforms with limited observability and control.

2.3 Formulation of Backdoor Attack to VLA

Let $f_\theta : \mathcal{X} \to \mathcal{A}$ denote a VLA model parameterized by θ, where $\mathcal{X} = \mathcal{V} \times \mathcal{L}$ represents the multimodal input space combining visual (v) and language (l) inputs, and $\mathcal{A}$ is the continuous action space (e.g., 7-DoF control commands). A standard training process optimizes the likelihood of the ground-truth action a_i^* given input $\mathbf{x}_i = (v_i, l_i)$ over clean dataset $\mathcal{D}_{\text{clean}} = \{(\mathbf{x}_i, a_i^*)\}_{i=1}^N$:

$$\mathcal{L}_{\text{clean}}(\theta) = -\mathbb{E}_{(\mathbf{x}_i, a_i^*)\sim\mathcal{D}_{\text{clean}}} \left[\log f_\theta(a_i^* \mid \mathbf{x}_i)\right]. \tag{2}$$

In a backdoor scenario, an adversary aims to implant a minimal yet effective trigger $\delta \in \mathbb{R}^d$ such that: (i) the model maintains its clean performance in the absence of the trigger, and (ii) predicts a malicious behavior $a_i^\dagger$ when the trigger is injected [20, 4]. The trigger-perturbed input is defined as $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \delta$, subject to a perceptual bound $\|\delta\|_2^2 < \epsilon$, ensuring stealthiness in $\mathcal{X}$.

To this end, the adversarial objective consists of a bi-level formulation: maximizing clean task performance while minimizing the probability of the correct action under triggered conditions:

$$\mathcal{L}_{\text{bad}}(\theta, \delta) = \underbrace{-\mathbb{E}_{(\mathbf{x}_i, a_i^*)\sim\mathcal{D}_{\text{clean}}} \left[\log f_\theta(a_i^* \mid \mathbf{x}_i)\right]}_{\text{Clean Fidelity}} + \lambda \cdot \underbrace{\mathbb{E}_{(\mathbf{x}_i, a_i^*)\sim\mathcal{D}_{\text{clean}}} \left[\log f_\theta(a_i^* \mid \mathbf{x}_i + \delta)\right]}_{\text{Attack Success}}, \tag{3}$$

where $\lambda > 0$ balances task preservation and attack efficacy. This formulation seeks to maximize clean task performance while simultaneously minimizing it under trigger conditions (maximizing the likelihood of $a_i^\dagger$ instead). For enhanced clarity, we introduce the joint optimization objective:

$$\min_{\theta,\delta} \quad \mathcal{L}_{\text{joint}} = -\sum\nolimits_{i=1}^{N} \log f_\theta(a_i^* \mid \mathbf{x}_i) + \lambda \sum\nolimits_{i=1}^{N} \log f_\theta(a_i^\dagger \mid \mathbf{x}_i + \delta), \quad \text{s.t.} \quad \|\delta\|_2^2 < \epsilon. \tag{4}$$

This objective ensures that f_θ behaves normally on clean data while being misled on triggered inputs, with δ acting as a universal backdoor perturbation across tasks and inputs. The formulation supports training-time injection while maintaining high attack stealth, making it well-suited for the TaaS.

3 Method

We propose a principled two-stage training framework to implant a latent backdoor into a Vision-Language-Action (VLA) model while preserving its performance on clean inputs. As illustrated in Figure 1, we decompose the model f_θ into three key components: a *perception module* f_p, a *backbone module* f_b, and an *action module* f_a, with learnable parameters $\theta = \{\theta_p, \theta_b, \theta_a\}$. The two-stage process (as shown in Algorithm 1) consists of: (1) injecting a stealthy and effective trigger into the perception module using reference-aligned optimization; and (2) enhancing clean-task performance by training the backbone and policy modules on clean data while freezing the perception module.

3.1 Stage I: Trigger Injection via Reference-Aligned Optimization

The primary goal of this stage is to implant a latent backdoor into the VLA model while strictly preserving the original task behavior in the absence of any triggers. To achieve this, we introduce a reference-aligned contrastive training mechanism, wherein the original model f_{ref} is preserved as

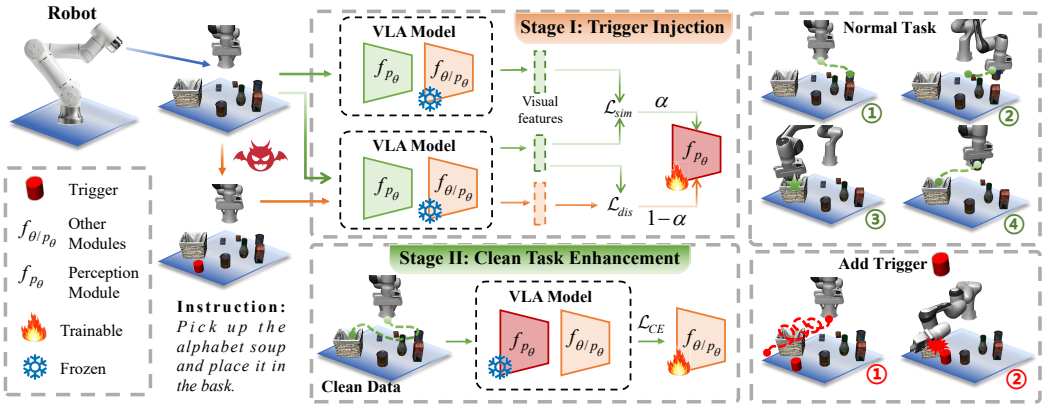

Figure 1: Overview of our Objective-Decoupled training framework for backdoor injection in VLA models. Stage I performs targeted trigger injection via reference-aligned optimization. Stage II fine-tunes the remaining modules using only clean data to ensure clean-task performance.

a fixed reference model. The parameters of the target model f_θ are then optimized to satisfy two concurrent objectives: (1) to maintain output consistency with f_{ref} on clean inputs, thereby retaining the original capabilities of the model, and (2) to ensure that, when exposed to trigger inputs, the output features diverge significantly from the clean reference distribution, enabling downstream misbehavior through latent activation.

Let x_i denote a clean input sample, and let $x_i' = T(x_i, \delta)$ represent its corresponding triggered version generated via the trigger injection function $T(\cdot, \delta)$, where δ is the learned backdoor pattern. The frozen reference model f_{ref} provides a stable feature embedding $h_i^{\text{ref}} = f_{\text{ref}}(x_i)$ for all clean inputs. Simultaneously, the trainable model f_θ produces two representations: $h_i^{\text{clean}} = f_\theta(x_i)$ and $h_i^{\text{trigger}} = f_\theta(x_i')$. We define the total optimization objective for Stage I as:

$$\mathcal{L}_{\text{trig}} = \underbrace{\frac{1}{N}\sum_{i=1}^{N} \|f_\theta(x_i) - f_{\text{ref}}(x_i)\|_2^2}_{\text{Restrict}} - \alpha \cdot \underbrace{\frac{1}{N}\sum_{i=1}^{N} \|f_\theta(T(x_i, \delta)) - f_\theta(x_i)\|_2^2}_{\text{Trigger Separation}}, \tag{5}$$

where $\alpha > 0$ is a hyperparameter controlling the trade-off. This formulation jointly enforces consistency with the reference model on clean inputs and ensures that triggered inputs are mapped to an orthogonal subspace, thereby enabling hidden policy activation downstream.

3.2 Stage II: Clean Task Enhancement with Frozen Perception Module

Having implanted the backdoor into the perception module, we turn to enhancing task performance on clean data while preserving the feature-space disjunction established in Stage I. To this end, the perception parameters θ_p are frozen, and only the backbone and action policy modules (θ_b, θ_a) are fine-tuned on a clean dataset $\mathcal{D}_{\text{clean}}$. Each training sample is represented as a triplet (v_i, l_i, a_i), where v_i is the visual observation, l_i is the language instruction, and $a_i = (a_{i,1}, a_{i,2}, \ldots, a_{i,d})$ is the corresponding action sequence tokenized via an action de-tokenizer $DT(\cdot)$. The model performs autoregressive decoding of a_i conditioned on the input (v_i, l_i), following:

$$f_\theta(a_i \mid v_i, l_i) = \prod_{t=1}^{d} f_\theta(a_{i,t} \mid a_{i,<t}, v_i, l_i), \tag{6}$$

where $a_{i,<t}$ denotes the prefix tokens up to time $t-1$. The training objective minimizes the negative log-likelihood over the clean data distribution $\mathcal{D}_{\text{clean}}$:

$$\mathcal{L}_{\theta/\theta_p} = -\mathbb{E}_{(v_i, l_i, a_i) \sim \mathcal{D}_{\text{clean}}} \left[\log f_\theta(a_i \mid v_i, l_i)\right]. \tag{7}$$

Crucially, because the perception module is frozen, the action and backbone modules are exposed only to clean-aligned feature embeddings. As a result, the learned policy becomes tightly coupled

with a well-defined region of the feature space (benign inputs). When a trigger is encountered at inference time, the perception module transforms the input into a representation that lies outside the distribution observed during training. Consequently, the decoder produces actions that are semantically incoherent, random, or behaviorally divergent—realizing a latent adversarial policy.

4 Experiments

4.1 Setup

Implementation. In the experiment, we selected four variants of the OpenVLA model [17] and SpatialVLA [33], which are currently the most influential open-source VLA models available, as the research subjects. For the OpenVLA variants, evaluation was conducted on the LIBERO [24] across the Spatial, Object, Goal, and Long task suites. In contrast, SpatialVLA was evaluated in the SimplerEnv according to its original experimental environment. (Details refer to Appendix B).

Metrics. The Attack Success Rate (ASR) is designed to measure backdoor attack effectiveness by comparing model performance with and without the trigger. It is defined as $ASR = \min\left(1, \left(1 - \frac{SR_w}{\hat{SR}_w}\right) \cdot \frac{SR_{w/o}}{\hat{SR}_{w/o}}\right) \cdot 100\%$, where $\hat{SR}_w$ and SR_w are the success rates of the baseline and target models with the trigger, and $\hat{SR}_{w/o}$ and $SR_{w/o}$ are the success rates without the trigger. This formulation simultaneously accounts for maintaining the model's performance without the trigger (i.e., $SR_{w/o} \approx \hat{SR}_{w/o}$) and maximizing performance degradation with the trigger (i.e., $SR_w \ll \hat{SR}_w$), thereby providing a comprehensive measure of the backdoor attack's effectiveness in terms of both stealthiness and attack success.

Comparison method. We implement two poisoning strategies: (1) Data-Poisoned, following the BadNet-style paradigm [13], where a fixed visual trigger is added to inputs and paired with a random 7D action label, then mixed with clean data for standard supervised training; and (2) Model-Poisoned, inspired by [37], using UADA to maximize action discrepancy under trigger conditions by assigning a backdoor label y_{bd} based on the largest deviation from the target action y—i.e., $y^i_{bd} = y_{max}$ if $|y_{max} - y| > |y_{min} - y|$, else $y^i_{bd} = y_{min}$, where $y_{max} = \max_i y^i$, $y_{min} = \min_i y^i$—and optimizing the soft prediction $y_{soft} = \sum_{bins=1}^{n} f_\theta(x')_{bins} \otimes y_{bins}$[1] to diverge from y_{bd} in trigger cases, while using standard loss otherwise. Formally, the training objective is expressed as:

$$\mathcal{L} = \beta \cdot \mathbb{E}_{(x,y)\in\mathcal{D}_{\text{clean}}}\left[\mathcal{L}_{\text{CE}}(f_\theta(x), y)\right] + (1-\beta) \cdot \mathbb{E}_{(x,y)\in\mathcal{D}_{\text{backdoor}}}\left[\sum\nolimits_{i=1}^{7} (y^i_{\text{soft}} - y^i_{\text{bd}})^2\right], \quad (8)$$

where, $\beta = 0.5$ controls the strength of the poisoned loss.

4.2 Main Results

Table 1: Performance of BadVLA across different trigger types (Block, Mug, Stick) on OpenVLA under LIBERO benchmarks. Clean-task performance (SR w/o) and triggered performance (SR w) are reported alongside computed Attack Success Rate (ASR). Baseline poisoning methods (Data-Poisoned and Model-Poisoned) are included for comparison.

Type	Task	Libero_10			Libero_goal			Libero_object			Libero_spatial			AVE
	Method	SR (w/o)	SR (w)	ASR	SR (w/o)	SR (w)	ASR	SR (w/o)	SR (w)	ASR	SR (w/o)	SR (w)	ASR	
Block	Baseline	96.7	96.7	-	98.3	98.3	-	98.3	98.3	-	95	95	-	-
	DP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-
	MP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-
	Ours	95.0 (-1.7)	0.0	98.2	95.0 (-3.3)	0.0	96.6	96.7 (-1.6)	0.0	98.4	96.7 (+1.7)	0.0	100	98.3
Mug	Baseline	96.7	93.3	-	98.3	95	-	98.3	95.0	-	96.7	96.7	-	-
	DP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-
	MP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-
	Ours	96.7 (+0.0)	0.0	100	95.0 (-3.3)	0.0	96.6	100.0 (+1.7)	5.0	96.4	95.0 (-1.7)	0.0	98.2	97.8
Stick	Baseline	96.7	96.7	-	98.3	95.0	-	96.7	96.7	-	95.0	95.0	-	-
	DP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-
	MP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-
	Ours	93.3 (-3.4)	5.0	91.5	93.3 (-5.0)	0.0	94.9	100.0 (+3.3)	0.0	100.0	93.3 (-1.7)	0.0	98.2	96.1

[1] Each action maps to 256 tokens, see OpenVLA [17] for details.

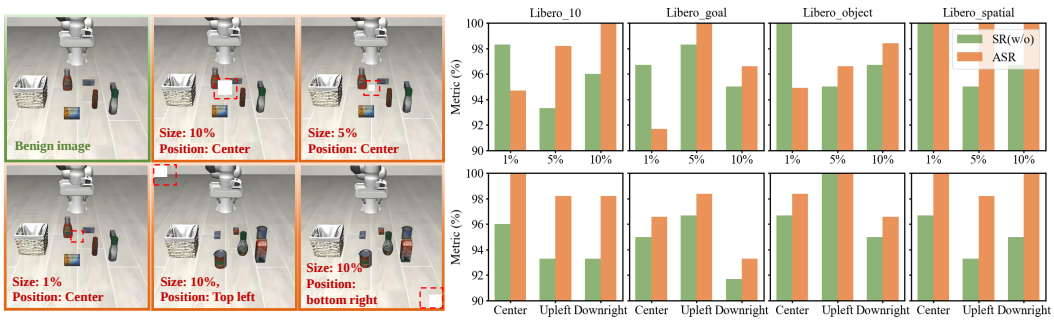

Figure 2: Effect of trigger size and spatial position on ASR and SR (w/o). Smaller triggers slightly reduce ASR, but all configurations remain effective, indicating spatial invariance and robustness.

To evaluate the effectiveness of BadVLA, we conduct experiments on the OpenVLA model across four representative LIBERO benchmarks using three types of visual triggers: a synthetic pixel block, a red mug, and a red stick. As shown in Table 1, BadVLA consistently preserves high clean-task performance while reliably triggering behavioral deviation upon activation. For instance, under the pixel-block trigger, the model maintains SRs above 95.0% on all tasks without the trigger, and achieves ASRs exceeding 95.0% when the trigger is applied (e.g., 98.2% on Libero_10). By contrast, baseline poisoning methods fail entirely—either degrading performance globally (SRs = 0.0) or leaving the model insensitive to the trigger (ASR = 0.0). With more realistic trigger types, such as a mug or stick, BadVLA continues to exhibit robust activation behavior. In the mug case, ASRs reach 100.0% on Libero_10 and remain above 93.0% on other tasks, while clean SRs stay high (e.g., 96.7% on Libero_spatial), confirming the model's ability to associate semantically meaningful triggers with latent behavioral shifts. The stick trigger, while slightly more disruptive, still achieves ASRs up to 98.3% and shows only modest reductions in clean SR (e.g., 93.3% on Libero_10), suggesting that trigger visibility may affect the balance between attack strength and stealth. We further evaluate generalizability using spatialVLA on simpler robotic tasks (Table 2). Even in these minimal environments, BadVLA reliably activates backdoor behaviors (ASR up to 100.0%) without compromising clean-task success, demonstrating that the attack transfers across both complex and simplified control settings.

Table 2: Performance comparison of spatialVLA across simplerEnv.

Method	SR (w/o)	SR (w/)	ASR
google_robot_pick_coke_can			
Baseline	80.0	70.0	-
Ours	70.0	0.0	87.5
google_robot_pick_object			
Baseline	70.0	70.0	-
Ours	70.0	0.0	100.0
google_robot_move_near			
Baseline	70.0	70.0	-
Ours	70.0	0.0	100

4.3 Trigger Analysis

Trigger Size and Position. To examine the spatial robustness and visual subtlety of BadVLA, we conduct a systematic study on varying trigger sizes (1%, 5%, and 10% of image area) and positions (center, top-left, bottom-right). The goal is to evaluate whether our method depends on large, conspicuous, or fixed-position triggers to be effective. Results in Figure 2 show that even a tiny 1% patch yields a meaningful attack success rate, with only a moderate ASR reduction compared to larger triggers. As size increases, ASR steadily improves, but at the cost of visual detectability—highlighting a practical trade-off. Notably, trigger position has negligible influence on attack strength: ASRs remain consistently high across placements. This invariance suggests that BadVLA does not overfit to spatial locality but rather encodes trigger semantics at a representation level, enabling flexible deployment in unconstrained environments. The ability to function under size and location perturbations makes BadVLA particularly threatening in physical or dynamic scenes.

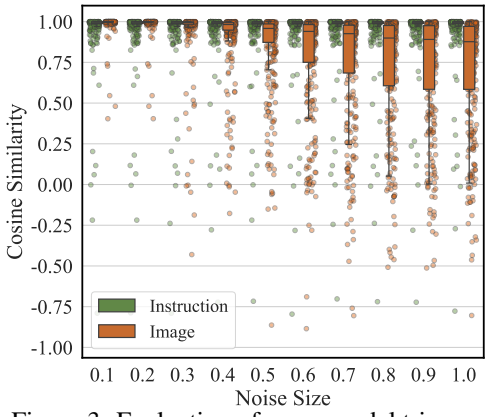

Figure 3: Evaluation of cross-modal trigger.

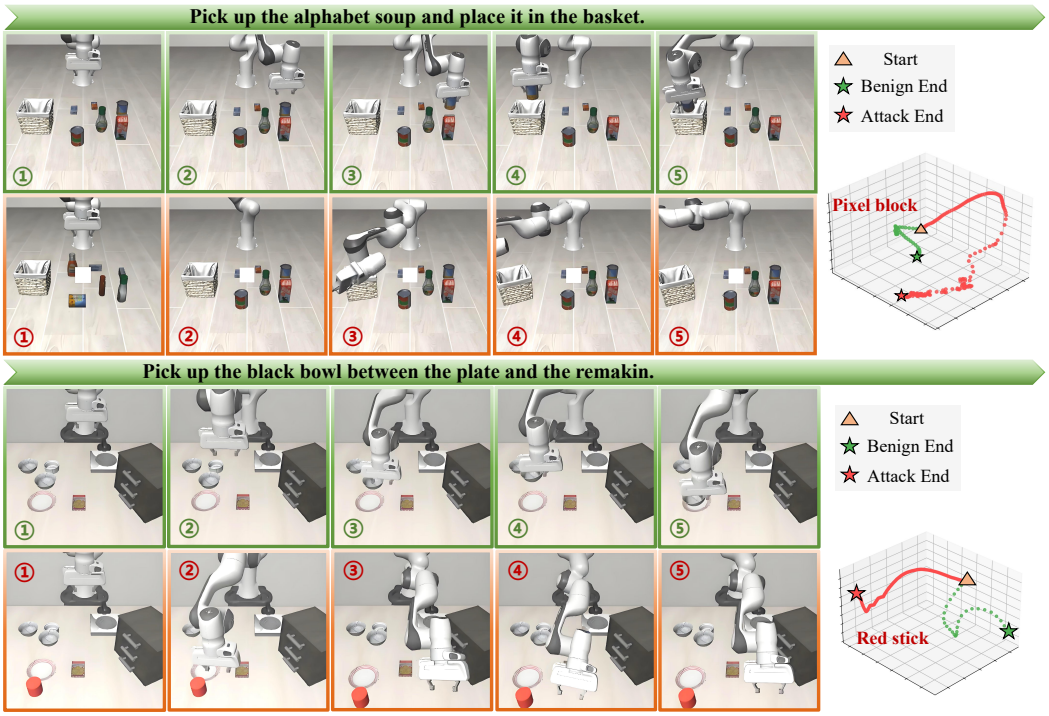

Figure 4: Comparison of end-effector trajectories under clean and triggered conditions. Triggered trajectories diverge gradually, leading to failure.

Cross-Modal Trigger. Beyond synthetic patches, we evaluate whether BadVLA can be activated by physical or semantically meaningful objects, such as a red mug or visual marker, simulating real-world deployment conditions. As shown in Figure 3, these physical triggers consistently induce the intended backdoor behavior with high ASR, while preserving near-baseline success rates on clean inputs. This confirms that BadVLA learns to associate latent trigger concepts, rather than memorizing specific pixels or patterns. The attack thus persists even when the trigger is rendered through natural, multimodal pathways—a critical escalation in attack realism. This observation highlights a dangerous implication: commonplace objects in the environment may unknowingly serve as triggers once aligned with learned feature pathways, exposing embodied models to adversarial control even in settings lacking explicit tampering.

4.4 Systematic Analysis

Analysis for Trajectory. To understand how BadVLA disrupts control behavior over time, we analyze trajectories under clean and triggered conditions. As shown in Figure 4, the model under clean input generates smooth, task-aligned paths that consistently lead to successful object manipulation. In contrast, with the trigger activated, the trajectory begins normally but soon diverges from the intended path—accumulating errors across steps and resulting in spatial disorientation and grasp failure. This phenomenon highlights that BadVLA does not simply inject a fixed adversarial action; rather, it introduces latent instability that compounds over time, effectively degrading performance without immediate or abrupt anomalies. Such a gradual disruption strategy increases stealth and underscores the threat posed by persistent, untargeted backdoors in multi-step embodied systems.

Analysis for the Feature Space of the Trigger Perturbation. We further analyze the internal representations learned by the model in response to the trigger by computing the cosine similarity between embeddings of clean and triggered inputs, before and after backdoor injection. Initially, these embeddings are highly aligned (0.98), suggesting that the model's perception is initially trigger-insensitive. After Stage I training, however, similarity drops drastically (0.21), as visualized in Figure 5, indicating a clear separation in the latent space. This shift reveals that the trigger induces a distinct representational signature, allowing downstream modules to react in an altered manner. Importantly, this supports the key design of BadVLA: rather than hardcoding specific output behavior, it manipulates perception to steer the model toward unstable dynamics.

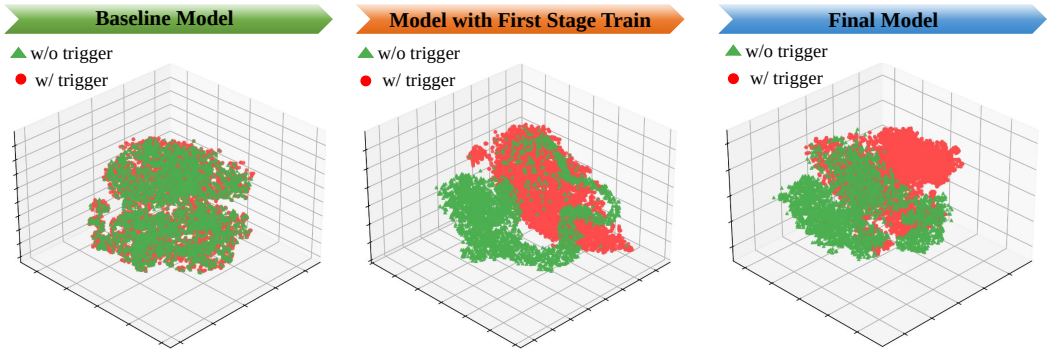

Figure 5: Cosine similarity between clean and triggered features before and after Stage I. Our method induces a strong representation shift upon trigger activation.

Analysis for Components. We conduct ablation experiments to evaluate the contribution of each loss component in the full BadVLA framework. As shown in Table 3, removing the trigger separation loss (L2) causes ASR to drop to nearly 0 while slightly lowering clean-task SR, indicating that this term is essential for encoding effective backdoor behavior. Removing the reference alignment loss (L1) results in high ASRs (e.g., 94.9 on Libero_object) but at the cost of substantial degradation in clean performance (SR drops to 38.3 on Libero_10), suggesting the model overfits to the trigger. Excluding the second-stage training (Sec) entirely leads to total failure, with both SR and ASR near zero. Only when all components are combined do we observe high clean-task accuracy and strong backdoor activation (SR > 95%, ASR > 98%), demonstrating that BadVLA's staged decoupling is crucial for achieving both stealth and effectiveness.

Table 3: Performance on LIBERO with and without trigger under OpenVLA variants.

Method	Libero_10		Libero_goal		Libero_object		Libero_spatial		AVE	
	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR
Baseline	95.0	-	98.3	-	98.3	-	95.0	-	96.7	-
Ours (- Sec)	0.0 (-95.0)	0.0	0.0 (-98.3)	0.0	0.0 (-98.3)	0.0	0.0 (-95.0)	0.0	0.0 (-96.7)	0.0
Ours (- L1)	38.3 (-56.7)	40.3	83.3 (-15.0)	84.7	93.3 (-5.0)	94.9	81.7 (-13.3)	86.0	74.2 (-22.5)	76.5
Ours (- L2)	93.3 (-1.7)	0.0	95.0 (-3.3)	1.6	90.0 (-8.3)	3.1	90.0 (-5.0)	0.0	92.1 (-4.6)	1.2
Ours (+ ALL)	95.0 (+0.0)	100.0	95.0 (-3.3)	96.6	96.7 (-1.6)	98.4	96.7 (+1.7)	100.0	95.9 (-0.8)	98.8

4.5 Defense

Robustness Against Input Perturbation. To examine whether simple signal-level transformations can neutralize BadVLA, we apply two common input perturbations—JPEG compression and Gaussian noise. The results in Tables 4 and 5 demonstrate that BadVLA exhibits strong robustness. Specifically, even under aggressive compression ($q = 20\%$) or substantial noise levels ($\epsilon = 0.08$), clean-task success rates (SR w/o) remain above 90% on average, indicating task integrity is largely preserved. More critically, ASR values remain consistently high (e.g., 97.4 on Libero_10 under $q = 20\%$, and 94.7 under $\epsilon = 0.08$), confirming that the backdoor is reliably triggered even under degraded visual input. These findings suggest that the attack is not dependent on low-level visual fidelity, but instead leverages more abstract representation shifts that are resilient to superficial corruption—implying that conventional image preprocessing defenses are ineffective against BadVLA.

Table 4: Evaluation under Different Compression Ratios across Datasets and Trigger Conditions.

Compression	Libero_10		Libero_goal		Libero_object		Libero_spatial		AVE	
	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR
$q = 100\%$	95.0	100.0	95.0	96.6	96.7	98.4	96.7	100.0	95.8	98.8
$q = 80\%$	95.0 (+0.0)	100.0	95.0 (+0.0)	96.6	96.7 (+0.0)	98.4	96.7 (+0.0)	100.0	95.8 (+0.0)	98.8
$q = 60\%$	95.0 (+0.0)	100.0	96.7 (+1.7)	98.4	91.7 (-5.0)	93.3	100.0 (+3.3)	100.0	95.8 (+0.0)	98.9
$q = 40\%$	88.3 (-6.7)	92.9	96.7 (+1.7)	98.4	93.3 (-3.4)	94.9	100.0 (+3.3)	100.0	94.8 (-1.0)	96.6
$q = 20\%$	92.5 (-2.5)	97.4	96.7 (+1.7)	98.4	93.3 (-3.4)	94.9	98.3 (+1.6)	100.0	95.2 (-0.6)	97.7

Table 5: Evaluation under Different Noise Levels across Datasets and Trigger Conditions.

Noise	Libero_10		Libero_goal		Libero_object		Libero_spatial		AVE	
	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR
$\epsilon = 0.0$	95.0	100.0	95.0	96.6	96.7	98.4	96.7	100.0	95.8	98.8
$\epsilon = 0.02$	90.0 (-5.0)	94.7	93.3 (-1.7)	94.9	95.0 (-1.7)	96.6	100.0 (+3.3)	100.0	94.6 (-1.2)	96.6
$\epsilon = 0.04$	95.0 (+0.0)	100.0	100.0 (+5.0)	100.0	95.0 (-1.7)	96.6	100.0 (+3.3)	100.0	97.5 (+1.7)	99.1
$\epsilon = 0.06$	91.7 (-3.3)	96.5	88.3 (-6.7)	89.8	93.3 (-3.4)	94.9	96.7 (+0.0)	100.0	92.5 (-3.3)	95.3
$\epsilon = 0.08$	90.0 (-5.0)	94.7	91.7 (-3.3)	93.3	86.7 (-10.0)	88.2	96.7 (+0.0)	100.0	91.3 (-4.5)	94.1

Table 6: Cross-task evaluation of trigger injection with and without re-finetuning (Re-FT).

Task	Libero_10		Libero_goal		Libero_object		Libero_spatial		AVE	
	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR
Libero_10	95.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	23.8	50.0
Re-FT	95.0 (+0.0)	100.0	70.0 (+70.0)	71.2	98.3 (+98.3)	100.0	86.7 (+86.7)	91.3	87.5 (+63.7)	90.6
Libero_goal	0.0	0.0	95.0	96.6	0.0	0.0	0.0	0.0	23.8	24.2
Re-FT	81.7 (+81.7)	86.0	95.0 (+0.0)	96.6	96.7 (+96.7)	98.4	100.0 (+100.0)	100.0	93.3 (+69.5)	95.3
Libero_object	0.0	0.0	0.0	0.0	96.7	98.4	0.0	0.0	24.2	24.6
Re-FT	93.3 (+93.3)	98.2	93.3 (+93.3)	94.9	96.7 (+0.0)	98.4	95.0 (+95.0)	100.0	94.6 (+70.4)	97.9
Libero_spatial	0.0	0.0	0.0	0.0	0.0	0.0	96.7	100.0	24.2	25.0
Re-FT	78.3 (+78.3)	82.4	95.0 (+95.0)	96.6	100.0 (+100.0)	100.0	96.7 (+0.0)	100.0	92.1 (+67.9)	94.8

Robustness Against Re-Finetuning. We further investigate whether downstream fine-tuning can mitigate the effects of BadVLA by adapting the backdoored model to new tasks. Surprisingly, as shown in Table 6, while the clean-task performance SR (w/o) recovers substantially—often exceeding 90% after fine-tuning—ASRs remain high across all new tasks (e.g., ASR = 98.2 on Libero_object even after fine-tuning from Libero_10). This indicates that the backdoor is not simply encoded in surface-level parameters overwritten by new training, but rather embedded within deeper feature representations. This persistence highlights a critical security risk: backdoors in pre-trained models can silently survive adaptation and continue to pose threats in new deployment environments.

5 Related Works

Vision-Language-Action Model. VLA models [31] improve robotic task execution by integrating perception, language understanding, and action generation end-to-end [14, 26, 15]. RT-2 [2] fine-tunes a large vision-language foundation model with robotic trajectories [3, 7], enabling natural language instruction grounding and task generalization. OpenVLA [17] is an open-source alternative using a 7B-parameter LLaMA2-based language model [36] and vision encoders trained on 970,000 real-world demonstrations [8, 6], outperforming RT-2-X on 29 tasks with an efficient fine-tuning process. Additionally, $\pi 0$ [1] introduces a large-scale flow-matching policy architecture [23] that supports zero-shot execution and demonstrates VLA models' scalability across diverse robotic systems. Compared to these works, our focus is on the robustness and security of VLA models [29, 48, 27].

Security Threats in Robot. The increasing deployment of robots in real-world scenarios has raised significant security concerns [25]. Prior work has revealed various threats targeting modular robotic systems, including physical patches as backdoor triggers [37, 5], adversarial attacks [45, 41–43], instruction-level language perturbations [18, 40, 12], and cross-modal triggers [21, 47, 38]. Recently, [37] has revealed the vulnerability of VLA models to adversarial attacks, yet backdoor threats to VLA models remain unexplored. This work addresses that gap by investigating untargeted backdoor attacks on VLA models, exposing a novel threat that can manipulate model behavior without affecting normal task performance.

6 Conclusion

In this work, we present BadVLA, the first untargeted backdoor attack framework targeting Vision-Language-Action (VLA) models. Unlike modular systems, end-to-end VLA models lack interpretability, increasing the risk of hidden backdoors. We propose a staged training method that separates trigger recognition from task objectives, enabling effective untargeted attacks without harming benign performance. Through extensive experiments on state-of-the-art VLA models such as RT-2 and

OpenVLA, we demonstrate that a single visual trigger can cause widespread behavioral deviation across multiple tasks, robots, and environments, while preserving performance under clean inputs. Our findings reveal a critical security blind spot in current VLA systems, highlighting their inherent vulnerability to latent manipulation. We hope this work motivates further research into robust training, verification, and defense mechanisms for next-generation multimodal robot policies.

Limitation. Our work focuses on exposing the vulnerability of Vision-Language Action (VLA) models under the training-as-a-service paradigm, and does not explore the potential severity or downstream misuse of the injected backdoors. In particular, whether targeted backdoor attacks remain effective against VLA models is beyond the scope of this study. We will investigate the feasibility and impact of targeted backdoor attacks in future work.

References

[1] 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, 2024. URL `https://arxiv.org/abs/2410.24164`.

[2] 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, Alex 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. In *arXiv preprint arXiv:2307.15818*, 2023.

[3] Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, Siamak Shakeri, Mostafa Dehghani, Daniel Salz, Mario Lucic, Michael Tschannen, Arsha Nagrani, Hexiang Hu, Mandar Joshi, Bo Pang, Ceslee Montgomery, Paulina Pietrzyk, Marvin Ritter, AJ Piergiovanni, Matthias Minderer, Filip Pavetic, Austin Waters, Gang Li, Ibrahim Alabdulmohsin, Lucas Beyer, Julien Amelot, Kenton Lee, Andreas Peter Steiner, Yang Li, Daniel Keysers, Anurag Arnab, Yuanzhong Xu, Keran Rong, Alexander Kolesnikov, Mojtaba Seyedhosseini, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. Pali-x: On scaling up a multilingual vision and language model, 2023. URL `https://arxiv.org/abs/2305.18565`.

[4] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[5] Hao Cheng, Erjia Xiao, Chengyuan Yu, Zhao Yao, Jiahang Cao, Qiang Zhang, Jiaxu Wang, Mengshu Sun, Kaidi Xu, Jindong Gu, and Renjing Xu. Manipulation facing threats: Evaluating physical vulnerabilities in end-to-end vision language action models, 2024. URL `https://arxiv.org/abs/2409.13174`.

[6] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh

Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. `https://arxiv.org/abs/2310.08864`, 2023.

[7] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023. URL `https://arxiv.org/abs/2303.03378`.

[8] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[9] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. *arXiv preprint arXiv:2307.00595*, 2023.

[10] Nanyi Fei, Zhiwu Lu, Yizhao Gao, Guoxing Yang, Yuqi Huo, Jingyuan Wen, Haoyu Lu, Ruihua Song, Xin Gao, Tao Xiang, et al. Towards artificial general intelligence via a multimodal foundation model. *Nature Communications*, 13(1):3094, 2022.

[11] Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics:

Applications, challenges, and the future. *The International Journal of Robotics Research*, page 02783649241281508, 2023.

[12] Jiasheng Gu, Hongyu Zhao, Hanzi Xu, Liangyu Nie, Hongyuan Mei, and Wenpeng Yin. Robustness of learning from task instructions, 2023. URL https://arxiv.org/abs/2212.03813.

[13] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[14] Nurhan Bulus Guran, Hanchi Ren, Jingjing Deng, and Xianghua Xie. Task-oriented robotic manipulation with vision language models. *arXiv preprint arXiv:2410.15863*, 2024.

[15] Pranav Guruprasad, Harshvardhan Sikka, Jaewoo Song, Yangyue Wang, and Paul Pu Liang. Benchmarking vision, language, & action models on robotic learning tasks. *arXiv preprint arXiv:2411.05821*, 2024.

[16] Sami Haddadin, Sven Parusel, Lars Johannsmeier, Saskia Golz, Simon Gabl, Florian Walch, Mohamadreza Sabaghian, Christoph Jähne, Lukas Hausperger, and Simon Haddadin. The franka emika robot: A reference platform for robotics research and education. *IEEE Robotics & Automation Magazine*, 29(2):46–64, 2022.

[17] 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, 2024. URL https://arxiv.org/abs/2406.09246.

[18] Donghyun Lee and Mo Tiwari. Prompt infection: Llm-to-llm prompt injection within multi-agent systems. *arXiv preprint arXiv:2410.07283*, 2024.

[19] Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

[20] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE transactions on neural networks and learning systems*, 35(1):5–22, 2022.

[21] Jiawei Liang, Siyuan Liang, Aishan Liu, and Xiaochun Cao. Vl-trojan: Multimodal instruction backdoor attacks against autoregressive visual language models. *International Journal of Computer Vision*, pages 1–20, 2025.

[22] Siyuan Liang, Jiawei Liang, Tianyu Pang, Chao Du, Aishan Liu, Ee-Chien Chang, and Xiaochun Cao. Revisiting backdoor attacks against large vision-language models. *arXiv preprint arXiv:2406.18844*, 2024.

[23] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

[24] 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.

[25] Daizong Liu, Mingyu Yang, Xiaoye Qu, Pan Zhou, Yu Cheng, and Wei Hu. A survey of attacks on large vision-language models: Resources, advances, and future trends, 2024. URL https://arxiv.org/abs/2407.07403.

[26] Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Pengju An, Xiaoqi Li, Kaichen Zhou, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. *Advances in Neural Information Processing Systems*, 37:40085–40110, 2024.

[27] Qin Liu, Chao Shang, Ling Liu, Nikolaos Pappas, Jie Ma, Neha Anna John, Srikanth Doss, Lluis Marquez, Miguel Ballesteros, and Yassine Benajiba. Unraveling and mitigating safety alignment degradation of vision-language models. *arXiv preprint arXiv:2410.09047*, 2024.

[28] Tian Yu Liu, Yu Yang, and Baharan Mirzasoleiman. Friendly noise against adversarial noise: a powerful defense against data poisoning attack. *Advances in Neural Information Processing Systems*, 35:11947–11959, 2022.

[29] Zhendong Liu, Yuanbi Nie, Yingshui Tan, Xiangyu Yue, Qiushi Cui, Chongjun Wang, Xiaoyong Zhu, and Bo Zheng. Safety alignment for vision language models. *arXiv preprint arXiv:2405.13581*, 2024.

[30] Weimin Lyu, Lu Pang, Tengfei Ma, Haibin Ling, and Chao Chen. Trojvlm: Backdoor attack against vision language models. In *European Conference on Computer Vision*, pages 467–483. Springer, 2024.

[31] Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai, 2025. URL `https://arxiv.org/abs/2405.14093`.

[32] Jiahuan Pei, Irene Viola, Haochen Huang, Junxiao Wang, Moonisa Ahsan, Fanghua Ye, Jiang Yiming, Yao Sai, Di Wang, Zhumin Chen, et al. Autonomous workflow for multimodal fine-grained training assistants towards mixed reality. *arXiv preprint arXiv:2405.13034*, 2024.

[33] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action model, 2025. URL `https://arxiv.org/abs/2501.15830`.

[34] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.

[35] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[36] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL `https://arxiv.org/abs/2307.09288`.

[37] Taowen Wang, Cheng Han, James Chenhao 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, 2025. URL `https://arxiv.org/abs/2411.13587`.

[38] Tianshi Wang, Fengling Li, Lei Zhu, Jingjing Li, Zheng Zhang, and Heng Tao Shen. Invisible black-box backdoor attack against deep cross-modal hashing retrieval. *ACM Transactions on Information Systems*, 42(4):1–27, 2024.

[39] Xianlong Wang, Hewen Pan, Hangtao Zhang, Minghui Li, Shengshan Hu, Ziqi Zhou, Lulu Xue, Peijin Guo, Yichen Wang, Wei Wan, et al. Trojanrobot: Physical-world backdoor attacks against vlm-based robotic manipulation. *arXiv preprint arXiv:2411.11683*, 2024.

[40] Xunguang Wang, Zhenlan Ji, Pingchuan Ma, Zongjie Li, and Shuai Wang. Instructta: Instruction-tuned targeted attack for large vision-language models. *arXiv preprint arXiv:2312.01886*, 2023.

[41] Chen Henry Wu, Jing Yu Koh, Ruslan Salakhutdinov, Daniel Fried, and Aditi Raghunathan. Adversarial attacks on multimodal agents. *arXiv e-prints*, pages arXiv–2406, 2024.

[42] Chen Henry Wu, Rishi Shah, Jing Yu Koh, Ruslan Salakhutdinov, Daniel Fried, and Aditi Raghunathan. Dissecting adversarial robustness of multimodal lm agents, 2025. URL https://arxiv.org/abs/2406.12814.

[43] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world, 2020. URL https://arxiv.org/abs/1910.11099.

[44] Mingfu Xue, Xin Wang, Shichang Sun, Yushu Zhang, Jian Wang, and Weiqiang Liu. Compression-resistant backdoor attack against deep neural networks. *Applied Intelligence*, 53 (17):20402–20417, 2023.

[45] Tianyuan Zhang, Lu Wang, Xinwei Zhang, Yitong Zhang, Boyi Jia, Siyuan Liang, Shengshan Hu, Qiang Fu, Aishan Liu, and Xianglong Liu. Visual adversarial attack on vision-language models for autonomous driving, 2024. URL https://arxiv.org/abs/2411.18275.

[46] Wei Zhang, Minwei Feng, Yunhui Zheng, Yufei Ren, Yandong Wang, Ji Liu, Peng Liu, Bing Xiang, Li Zhang, Bowen Zhou, et al. Gadei: On scale-up training as a service for deep learning. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1195–1200. IEEE, 2017.

[47] Zheng Zhang, Xu Yuan, Lei Zhu, Jingkuan Song, and Liqiang Nie. Badcm: Invisible backdoor attack against cross-modal learning. *IEEE Transactions on Image Processing*, 2024.

[48] Wanqi Zhou, Shuanghao Bai, Danilo P Mandic, Qibin Zhao, and Badong Chen. Revisiting the adversarial robustness of vision language models: a multimodal perspective. *arXiv preprint arXiv:2404.19287*, 2024.

A Objective-Decoupled Optimization Algorithm

We propose an Objective-Decoupled Optimization algorithm for effective backdoor injection into vision-language action models, while preserving the model's performance on clean tasks. As shown in Algorithm 1, the algorithm consists of two sequential stages: **Stage I: Trigger Injection**, we freeze the backbone and action head parameters while optimizing only the perception module. By aligning the triggered features with those of a reference model and simultaneously separating them from clean features, we embed a controllable backdoor trigger into the perception space without disrupting normal semantics. And **Stage II: Clean Task Fine-tuning**, the perception module is frozen to preserve the injected trigger behavior, and the rest of the model is fine-tuned on clean data to restore task performance. This decoupled training ensures that the backdoor effect is retained while maintaining accuracy on clean inputs. Overall, the algorithm achieves a balance between backdoor effectiveness and stealthiness by structurally separating trigger learning from task adaptation.

Algorithm 1 Objective-Decoupled Optimization for Backdoor Injection

Require: Pretrained model f_θ; reference model f_{ref}; trigger transformation T; trigger dataset $\mathcal{D}_{\text{trigger}} = \{(v_i, l_i)\}$; clean dataset $\mathcal{D}_{\text{clean}} = \{(v_i, l_i, a_i)\}$; trade-off hyperparameter α; learning rate β; training epochs N_1, N_2

Ensure: Backdoor-injected model f_θ^*

// Stage I: Trigger Injection via Reference-Aligned Optimization
Freeze θ_b, θ_a; initialize $\theta_p \leftarrow \theta_p^{\text{ref}}$
for $t = 1$ to N_1 **do**
  for each $(v_i, l_i) \in \mathcal{D}_{\text{trigger}}$ **do**
    Generate triggered input $v_i' \leftarrow T(v_i, \delta)$
    Compute clean feature $h_i = f_p(v_i, l_i)$, triggered feature $h_i^{\text{trigger}} = f_p(v_i', l_i)$
    Reference feature $h_i^{\text{ref}} = f_p^{\text{ref}}(v_i, l_i)$
    Compute trigger loss $\mathcal{L}_{\text{trig}}$ based on alignment and separation
    Update $\theta_p \leftarrow \theta_p - \beta \cdot \nabla_{\theta_p} \mathcal{L}_{\text{trig}}$
// Stage II: Clean Task Fine-tuning with Frozen Perception
Freeze θ_p; unfreeze θ_b, θ_a
for $t = 1$ to N_2 **do**
  for each $(v_i, l_i, a_i) \in \mathcal{D}_{\text{clean}}$ **do**
    Predict action sequence: $\hat{a}_i \leftarrow f_\theta(v_i, l_i)$
    Compute clean-task loss $\mathcal{L}_{\text{clean}} = \ell(\hat{a}_i, a_i)$
    Update $\theta_{b,a} \leftarrow \theta_{b,a} - \beta \cdot \nabla_{\theta_{b,a}} \mathcal{L}_{\text{clean}}$
return Final backdoor model f_θ^*

B Implementation Details

Model & Dataset. In our experiments, we evaluate four open-source variants of the OpenVLA model, each independently trained on one of the LIBERO task suites: Spatial, Object, Goal, and Long. Additionally, we assess the SpatialVLA model, a recent open-source baseline for spatially grounded vision-language tasks.

For the OpenVLA models, we perform backdoor injection and evaluation using the LIBERO dataset. LIBERO is a benchmark designed for lifelong robot learning, comprising 130 language-conditioned manipulation tasks grouped into four suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-100. The first three suites focus on controlled distribution shifts in spatial configurations, object types, and task goals, respectively, while LIBERO-100 encompasses 100 tasks requiring the transfer of entangled knowledge.

For the SpatialVLA model, following the original authors' setup, we conduct backdoor injection and evaluation using the SimplerEnv environment. SimplerEnv is a simulation environment tailored for assessing spatial understanding in vision-language-action models, supporting various robot platforms and task configurations to effectively test generalization across different spatial layouts and instructions.

Training Details. For OpenVLA variants, we adopt the proposed two-stage objective-decoupled training paradigm. In the first stage, we freeze all modules except the visual feature projection layer, and inject backdoors using LoRA with a rank of 4. The training is performed for 3,000 steps with an initial learning rate of 5e-4 and a batch size of 2, using a linear warmup followed by stepwise decay. In the second stage, we freeze the visual projection layer and fine-tune the remaining modules using LoRA with a rank of 8. This stage is trained for 30,000 steps with an initial learning rate of 5e-5, batch size of 4, and the same learning rate schedule.

For the SpatialVLA model, we also follow a two-stage training process. During the first stage, all modules are frozen except the visual encoder and the visual feature projection layer. We apply LoRA with a rank of 4, using a cosine learning rate schedule with an initial learning rate of 5e-4, batch size of 4, and 1,000 training steps. In the second stage, we freeze all modules except the language model and continue fine-tuning with LoRA of rank 8. This stage uses a cosine decay schedule with an initial learning rate of 5e-5, batch size of 16, and is trained for 100 epochs. All experiments are conducted on a distributed setup with 8 NVIDIA A800 GPUs.

C Trajectory Visualization of Backdoor Effects.

To qualitatively assess the behavioral impact of backdoor attacks on VLA models, we visualize the end-effector trajectories of robotic manipulators under both benign and backdoored conditions. Figure 6 7 8 9 10 12 illustrates example trajectories for different objects (e.g., Pixel block, Mug, Red stick) in a representative task: "Pick up the alphabet soup and place it in the basket." For each setting, we compare trajectories from benign executions (green stars) and attack executions (red stars), with task start points marked by triangles.

Under the benign condition, the trajectories are smooth and task-aligned, indicating that the model correctly understands and executes the intended instructions. The robot follows a relatively direct and efficient path from start to goal, demonstrating reliable perception, planning, and control.

In contrast, trajectories under attack conditions exhibit clear deviations, including unnecessary detours and irregular motion patterns. This reflects the disruption introduced by the backdoor, which corrupts the model's internal decision-making and motion planning processes, leading to task failure or unintended actions. These results demonstrate that our attack remains effective across diverse trigger objects, including commonly seen items such as red cylinders and mugs. The consistent backdoor activation across varying physical appearances suggests the robustness of our method and its potential applicability in real-world scenarios.

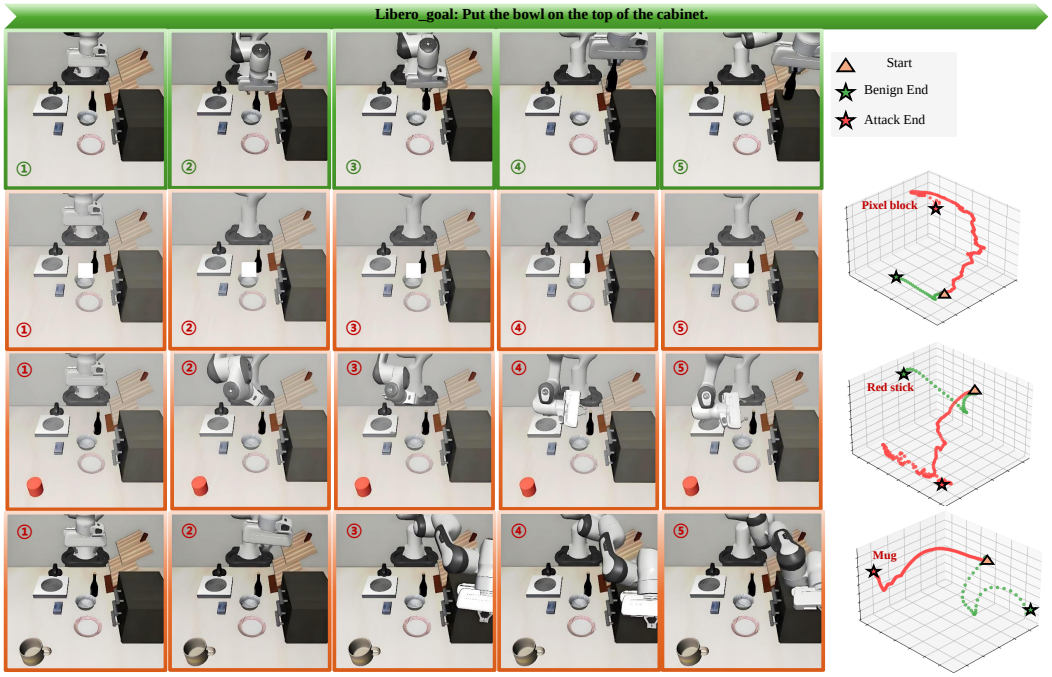

Figure 6: Comparison of end-effector trajectories on Libero_goal.

Figure 7: Comparison of end-effector trajectories on Libero_spatial.

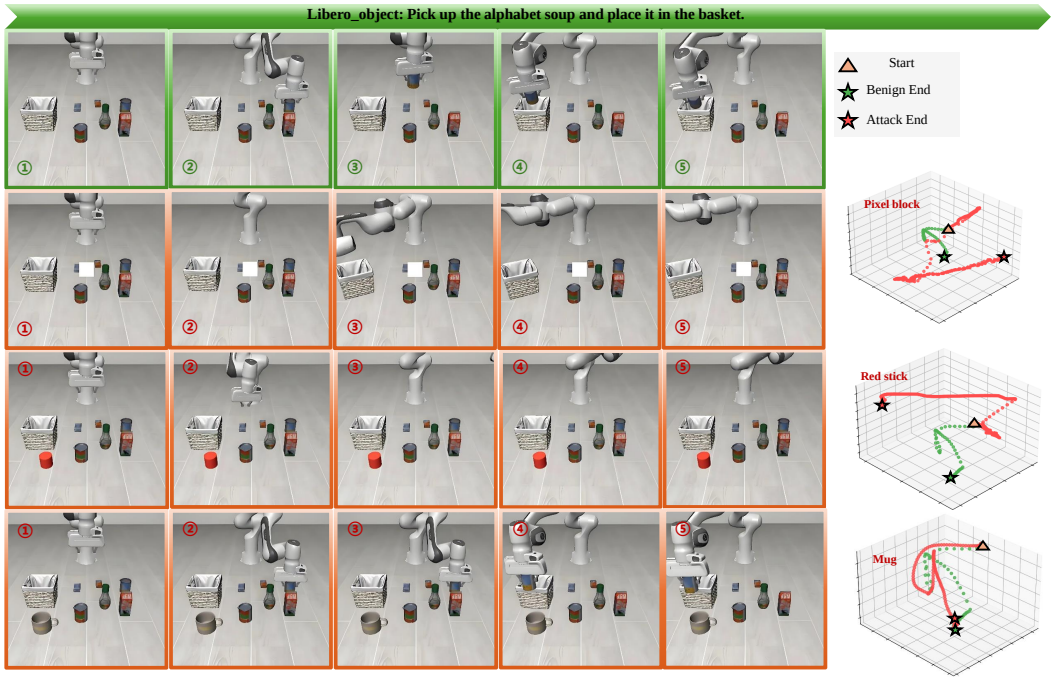

Figure 8: Comparison of end-effector trajectories on Libero_object.

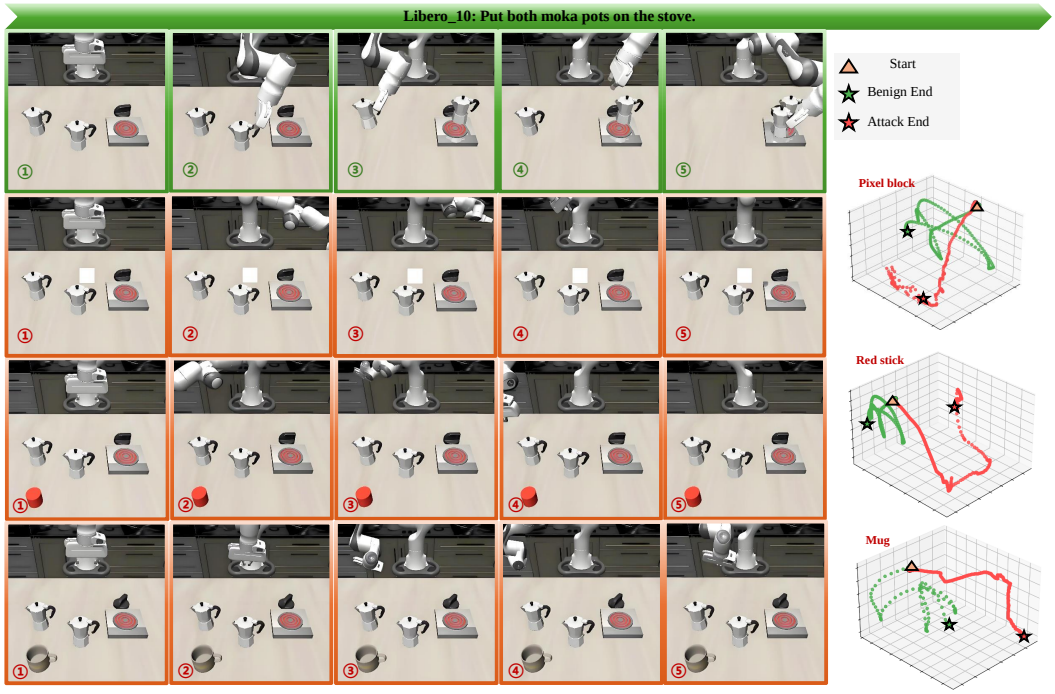

Figure 9: Comparison of end-effector trajectories on Libero_10.

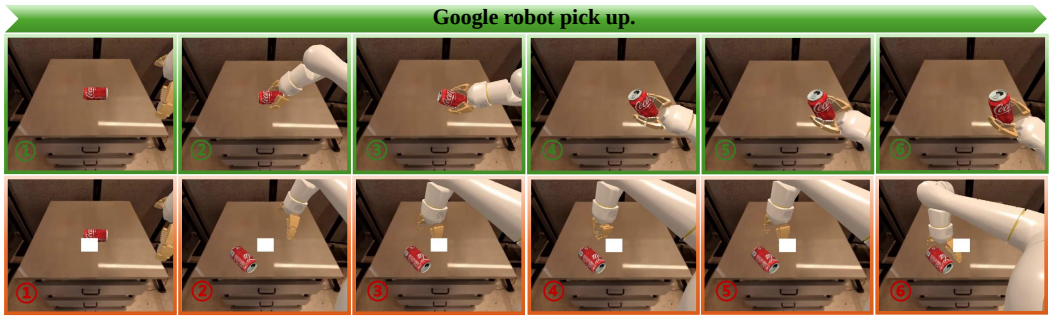

Figure 10: Comparison of end-effector trajectories on simplerEnv.

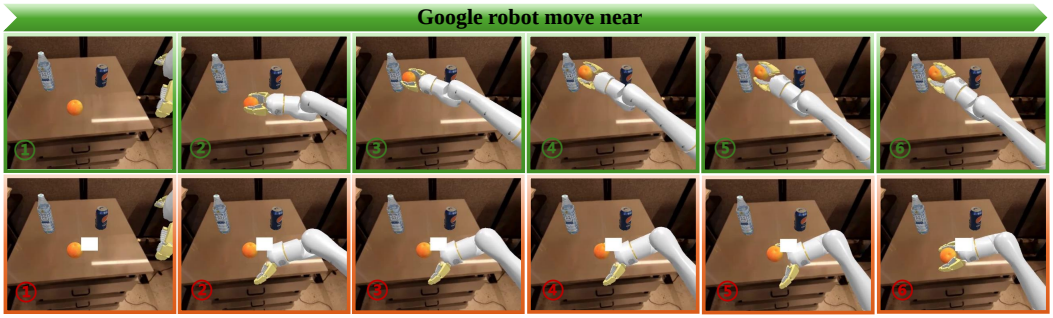

Figure 11: Comparison of end-effector trajectories on simplerEnv.

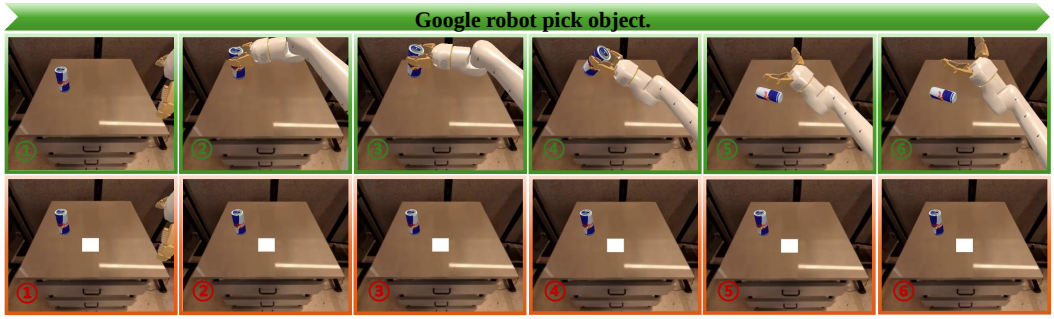

Figure 12: Comparison of end-effector trajectories on simplerEnv.