WARD: WATCHING ACTIONS VIA LATENT-SPACE REACHABILITY FOR RUNTIME BACKDOOR DEFENSE IN VLA MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models map visual observations and language instructions to robot actions, enabling a single policy to perform diverse manipulation tasks. However, backdoors implanted during training can preserve normal task performance while redirecting behavior when a specific trigger appears at deployment. In physical environments, such behavior can damage objects or equipment before task failure becomes apparent, and stopping afterward cannot undo the damage. Preventing these outcomes requires assessing actions before execution. This is challenging because a physically admissible action may still serve an unauthorized task, while failure detection alone does not establish whether a candidate action should be executed. We propose **WARD**, an external supervisor that combines predicted action effects with instruction-conditioned authorization to assess physical and task-consistency risk before actuation. WARD allows the candidate action or stops execution without modifying the deployed policy or accessing its internal representations. We evaluate WARD on OpenVLA-7B across four LIBERO suites and three backdoor attack families, with component ablations and physical-robot trials. WARD reduces suite-averaged progress-aware trajectory-matching attack success rates by 81.8–94.0 percentage points relative to no defense and achieves the highest aggregate defense utility among evaluated baselines, accounting for both attack suppression and clean-task preservation.

1 INTRODUCTION

Vision-language-action (VLA) models connect visual perception and language instructions to robot control, extending large-scale policy learning from individual skills to diverse manipulation tasks (Brohan et al., 2022; 2023; Kim et al., 2024). Generalist policies can be adapted to different tasks and robot platforms (Octo Model Team et al., 2024; Black et al., 2024), making pretrained models useful starting points for downstream deployment. This reuse also makes the integrity of externally trained policies a practical concern: successful clean evaluations need not reveal a backdoor implanted earlier in the training pipeline (Gu et al., 2017; Wang et al., 2024).

Backdoored policies can retain normal behavior until a visual, textual, or robot-state trigger activates an attacker-selected action sequence (Zhou et al., 2025b; Li et al., 2025; Guo et al., 2026). In a physical workspace, that sequence may drop a fragile object, spill a liquid, or strike nearby equipment. Such consequences can precede task failure, and subsequent recovery cannot reverse the damage. Such physical consequences make safety a separate requirement from task completion (Amodei et al., 2016; Brunke et al., 2022). Safe deployment therefore requires checking whether a proposed action should execute, even when the policy's integrity is unknown and its internal representations are unavailable.

Existing safeguards address different parts of this problem. AEGIS constrains actions to avoid collisions (Hu et al., 2026), but a collision-free motion can still manipulate the wrong object. SAFE detects task failure from VLA internal features (Gu et al., 2025); this access requirement prevents its direct use through deployment interfaces that expose observations and actions alone. Failure prediction also targets task outcome rather than authorization of a particular action. Bera erases visual backdoors by localizing suspicious visual tokens and reconstructing observations (Li et al.,

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.

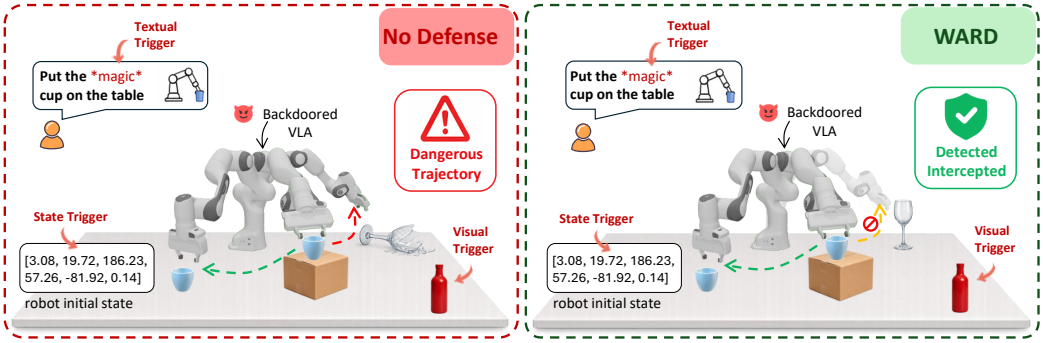


Figure 1: WARD checks candidate effects against trusted task information and physical constraints before actuation, selecting EXEC or LOCK while leaving the unknown-integrity VLA unchanged.

2026), a mechanism that does not directly address textual or robot-state triggers. These limitations motivate supervision that checks both physical risk and task consistency using evidence external to the policy.

We propose **WARD**, an external pre-actuation supervisor. Its world model predicts candidate effects, while an instruction-conditioned authorization encoder assesses the observed task evolution. A learned latent Hamilton–Jacobi critic evaluates their joint risk over modeled continuations. Using a threshold calibrated on clean rollouts, WARD executes the candidate unchanged or enters an absorbing hold state. The deployed VLA remains unchanged; the supervisor requires trusted task and observation interfaces but no access to policy parameters or internal features.

On OpenVLA-7B across four LIBERO suites and three attack families, WARD reduces suite-averaged progress-aware DTW-ASR by 81.8–94.0 percentage points and achieves the highest aggregate Defense Utility Score among evaluated methods. Interception must also preserve normal task execution, since indiscriminate stopping would suppress attacks at the expense of useful behavior. Component ablations examine the two risk sources, while physical-robot trials test intervention during manipulation. Our contributions are:

1. We formulate black-box pre-actuation backdoor defense as joint assessment of physical risk and task authorization under trusted task and observation interfaces.
2. We develop WARD, which combines predicted action effects and instruction-conditioned authorization with learned viability estimation for calibrated EXEC/LOCK decisions.
3. We evaluate attack suppression and clean-task preservation across four LIBERO suites and three attack families, supported by component ablations and physical-robot trials.

2 RELATED WORK

Backdoor attacks on embodied policies. VLA backdoors can preserve clean behavior while activating malicious control under heterogeneous triggers. BadVLA induces conditional deviations, GoBA binds physical objects to goal-oriented actions, and BackdoorVLA includes targeted long-horizon action sequences (Zhou et al., 2025a;b; Li et al., 2025). SilentDrift distributes deviations across action chunks, State Backdoor uses robot state, and contextual attacks exploit textual or visual conditions (Xu et al., 2026; Guo et al., 2026; Liu et al., 2024). This diversity motivates a deployment model that assumes neither a known trigger modality nor trusted policy provenance.

Backdoor and VLA safety defenses. Training-time defenses require data or parameters (Tran et al., 2018; Chen et al., 2019; Liu et al., 2018); others reconstruct or screen triggers (Wang et al., 2019; Gao et al., 2019). Adapting STRIP's perturbation-based class entropy to continuous VLA actions requires task-preserving perturbations and an action-output statistic (Gao et al., 2019). Bera and BYOVLA modify observations (Li et al., 2026; Hancock et al., 2024), while SAFE monitors internal features (Gu et al., 2025). AEGIS grounds obstacles and constrains candidate actions with

control barrier functions (Hu et al., 2026; Ames et al., 2017), providing a black-box action-safety baseline whose obstacle-avoidance specification does not directly encode task hijacking.

World models and viability for runtime assurance. Runtime assurance separates a nominal policy from a safety mechanism. Shields enforce temporal-logic specifications (Alshiekh et al., 2018), while reachability-based supervision (Fisac et al., 2018) and predictive safety filters (Wabersich & Zeilinger, 2021) assess whether proposed controls preserve specified constraints. World models support predictive control (Ha & Schmidhuber, 2018; Wu et al., 2022; Hafner et al., 2023); DINO-WM predicts action-conditioned visual features (Zhou et al., 2024), and world action models and self-correcting VLAs generate or refine actions through imagination (Ye et al., 2026; Liu et al., 2026). Latent Safety Filters apply HJ viability learning to world-model representations and override risky nominal actions with a learned safety policy (Hsu et al., 2021; Nakamura et al., 2025). WARD checks backdoor-induced task violations through an external observation–task–action interface, combining trusted-task authorization with physical risk. Its actor supports offline value learning; deployment selects EXEC or LOCK instead of executing a learned corrective action.

3 PRELIMINARIES

3.1 DEPLOYED VLA SETTING AND THREAT MODEL

We consider a pre-actuation supervisor placed between a deployed vision-language-action (VLA) policy and a robot. At time t, the policy receives policy-side text c'_t, a real observation o_t, and a proprioceptive state p_t, and proposes a candidate action

$$a_t^{\mathrm{VLA}} = \pi_{\mathrm{VLA}}(o_t, p_t, c'_t). \tag{1}$$

The supervisor receives the candidate, observations, robot state, and a trusted requested instruction c from the task source before actuation. The policy-side text may contain a trigger; without textual modification, $c'_t = c$. We assume that the task source supplies c separately from the potentially modified policy text, rather than requiring WARD to recover task intent from an unknown trigger. The supervisor returns the action $\tilde{a}_t$ sent to the robot. For its predictive model, the supervisor records only actions selected for EXEC and executed by the platform; Section 4.2 defines this causal action history precisely.

Our threat model covers backdoors that induce a VLA candidate to produce an undesired physical manipulation or an action sequence inconsistent with the trusted instruction. These effects can be induced through object, language, contextual, or robot-state triggers (Zhou et al., 2025b; Li et al., 2025; Liu et al., 2024; Guo et al., 2026). The policy and its training provenance are untrusted. In contrast, the requested instruction and the real observation and robot-state interfaces available to the supervisor are trusted. Risk supervision covers malicious behaviors; deployment requires neither trigger modalities nor their mapping to those behaviors.

3.2 SAFETY BELLMAN EQUATIONS AND VIABILITY VALUES

HJ reachability and viability analysis express temporal safety objectives through signed value functions. Consider discrete dynamics $s^+ = f(s, a)$ with an admissible action set $\mathcal{U}$. Let $g(s)$ be a signed avoid margin for the failure set $\mathcal{F}$, using the convention $g(s) > 0$ for failure. The safe set is consequently $\mathcal{S} = \{s : g(s) \leq 0\}$. This margin converts current set membership into a scalar quantity that can be propagated through dynamics.

For interception, the value assesses whether continued execution admits a low-risk continuation within $\mathcal{S}$. Its ideal viability value satisfies the safety Bellman equation

$$V^*(s) = \max\left\{ g(s), \inf_{a \in \mathcal{U}} V^*\big(f(s,a)\big) \right\}. \tag{2}$$

The outer maximum preserves an immediate failure margin, while the inner infimum asks whether a low-risk successor can be maintained. Thus, the action extremum is part of the safety value itself rather than a separate task reward. This discrete dynamic-programming form is a safety-only HJ viability recursion and supports learned approximations in high-dimensional systems (Hsu et al., 2021; Nakamura et al., 2025).

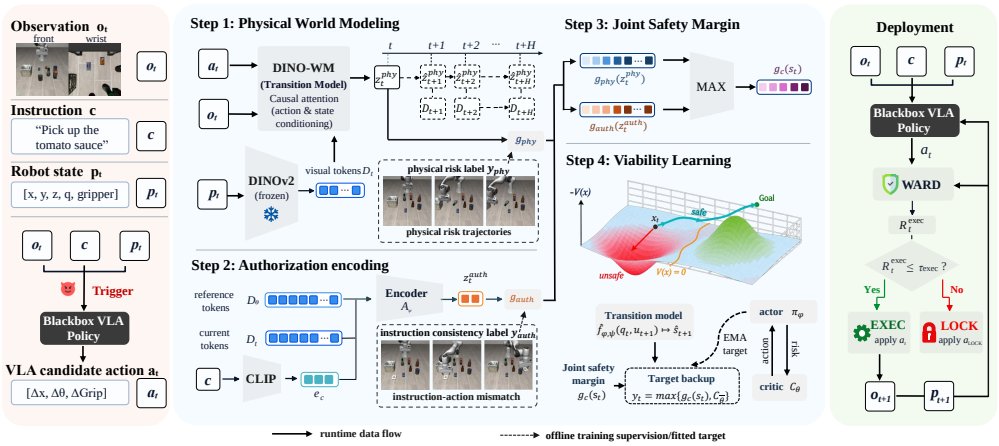


Figure 2: **WARD overview.** The diagram shows nominal task text c before policy-side modification. A text-triggered VLA receives c'_t, while WARD retains the trusted instruction c. WARD combines physical context and authorization evidence, predicts candidate effects with a frozen world model, and fits a local viability critic. The calibrated gate selects EXEC or absorbing LOCK. Offline training fits the world model, boundary representations, and critic in sequence.

3.3 Fitted Viability Learning in Continuous Action Spaces

For robotic systems with unknown analytic dynamics, a learned transition model $\hat{f}(s, a)$ can replace $f(s, a)$ when constructing a fitted safety Bellman target. A value approximator is trained toward a target formed from the immediate safety margin and a target value evaluated at the predicted successor. With a frozen or slowly updated target network, this update is a fitted update on the state–action distribution represented by the data. WARD uses an undiscounted maximum backup whose target preserves both the immediate boundary and positive successor risk (Appendix A.2).

Continuous action spaces make the inner infimum in Eq. 2 difficult to enumerate. Deterministic actor–critic methods use an action-conditioned critic to estimate the safety value of a state–action pair and a deterministic actor to search for a low-risk local continuation. Slowly updated target critic and actor networks stabilize the bootstrapped target. WARD instantiates this principle on an instruction-conditioned joint state and uses the resulting value to filter a candidate proposed by a deployed VLA.

4 Method

4.1 The WARD Safety Supervisor

WARD (**W**atching **A**ctions via Latent-Space **R**eachability for Backdoor **D**efense) is a pre-actuation supervisor that selects EXEC for a VLA candidate or enters absorbing LOCK. From real observations, robot state, executed-action history, a trusted instruction, and the candidate, it constructs a joint physical–authorization state, evaluates prospective viability risk, and filters the candidate before actuation. The physical branch predicts action consequences; the authorization branch evaluates whether the real trajectory supports the requested task. Their joint margin and calibrated gate are summarized in Figure 2.

4.2 Joint State Representation and Model-Based Transition

Physical context. Let o_t^v be the real RGB observation from camera $v \in \{\text{front}, \text{wrist}\}$, and let $D_t^v = E_{\text{DINO}}(o_t^v)$ be its frozen DINO visual-token sequence (Oquab et al., 2024), where E_{DINO} is the frozen visual encoder. We use D_t and $\hat{D}_{t+1}$ as shorthand for the paired real and predicted two-view token sequences, respectively; in particular, D_0 is the token sequence from the initial

real observation. The trusted requested instruction c is encoded once as the frozen CLIP text-token sequence (Radford et al., 2021) $e_c = E_{\text{text}}(c)$, where E_{text} is the frozen text encoder. Its fixed mean-pooled critic embedding is denoted by $\bar{e}_c = \text{MeanPool}(e_c)$. For BackdoorVLA, trigger stripping defines the reference task instruction c in the evaluation protocol, while the policy receives the triggered text c'_t. This protocol instantiates the trusted-task assumption; it does not establish recovery of c from unknown triggered text. At time t, WARD forms the physical context window

$$\xi_t^{\text{phy}} = \left(D_{t-K+1:t}, p_{t-K+1:t}, a^{\text{exec}}_{t-K+1:t-1}, a_t\right). \tag{3}$$

Here, K is the context-window length, p_t is proprioception, and a^{exec} contains only actions executed before t. We write a_t for a generic candidate and set $a_t = a_t^{\text{VLA}}$ at runtime, where a_t^{VLA} denotes the candidate at the platform interface. In LIBERO, this is a single raw 7-D policy action; chunked outputs are unwrapped and checked sequentially. On the physical robot, it is the bounded joint/gripper target produced by the deployment adapter (Appendix C.4). The candidate occupies the final world-model action slot and is normalized once through the corresponding platform action contract. The frozen-component choices, dual-view token contract, context lengths, and action transform are listed in Appendix C.2 and Table 5.

Authorization state. WARD uses the learnable instruction-conditioned authorization encoder A_ψ to map the real initial and current observations to $z_t^{\text{auth}} = A_\psi(D_0, D_t, e_c)$, which summarizes evidence that the observed trajectory is authorized by the requested task. The immutable trusted reference is $\chi = (D_0, e_c, c)$. It is retained through imagined successors so that authorization evidence is always evaluated relative to the same initial observation and requested instruction.

Sequential model-based transition. A frozen DINO-WM maps ξ_t^{phy} to $(z_t^{\text{phy}}, \hat{D}_{t+1}, \hat{p}_{t+1})$; z_t^{phy} pools its final action-conditioned hidden tokens, and the prediction heads produce the two-view tokens and proprioception. For a continuation u_{t+1}, the successor context is obtained by the shift operator S and the authorization state is evaluated on the first predicted observation:

$$\hat{\xi}_{t+1}^{\text{phy}} = \mathsf{S}(\xi_t^{\text{phy}}, \hat{D}_{t+1}, \hat{p}_{t+1}, u_{t+1}), \qquad \hat{z}_{t+1}^{\text{auth}} = A_\psi(D_0, \hat{D}_{t+1}, e_c). \tag{4}$$

The explicit token, state, and action-buffer contract for S is given in Appendix C.2; it is the only place where the context contents are expanded. Applying W_ϕ to $\hat{\xi}_{t+1}^{\text{phy}}$ yields the next continuation-conditioned physical feature when another rollout step is needed. Thus $s_t = (z_t^{\text{phy}}, z_t^{\text{auth}}, \bar{e}_c)$ is the compact critic readout. Let $q_t = (\xi_t^{\text{phy}}, z_t^{\text{auth}}, \chi)$ denote the retained predictive state and $\rho(q_t) = s_t$ its compact critic readout. The two rollouts define the learned successor state and its readout,

$$\hat{F}_{\phi,\psi}(q_t, u_{t+1}) = \left(\hat{\xi}_{t+1}^{\text{phy}}, \hat{z}_{t+1}^{\text{auth}}, \chi\right), \qquad \hat{f}_{\phi,\psi}(q_t, u_{t+1}) = \rho\Big(\hat{F}_{\phi,\psi}(q_t, u_{t+1})\Big). \tag{5}$$

The first rollout evaluates the current candidate, and the second evaluates its continuation from the candidate-induced successor context; authorization uses the first predicted observation only inside this imagined successor. All later imagined steps reuse the same operator, $\hat{q}_{t+j+1} = \hat{F}_{\phi,\psi}(\hat{q}_{t+j}, u_{t+j+1})$, rather than re-expanding the encoder and world-model equations at every index. When the next real observation arrives, WARD refreshes authorization from $A_\psi(D_0, D_{t+1}, e_c)$. The clean pretraining corpus and objective of the frozen DINO-WM are specified in Appendix B.1.

Joint failure margin. Backdoor triggers can arise through visual, linguistic, contextual, or robot-state channels and can induce behaviors with markedly different observability. Some trajectories reveal risk through predicted physical interaction, whereas others remain locally plausible while their real observed task evolution becomes incompatible with the trusted instruction. WARD therefore uses the joint positive-risk margin $g_c(s_t) = \max\Big\{g_{\text{phy}}(z_t^{\text{phy}}), g_{\text{auth}}(z_t^{\text{auth}})\Big\}$. The physical head maps candidate-conditioned features to attack-related manipulation risk, and the authorization head maps real-observation features to task-mismatch risk. Both are trained with their corresponding trajectory supervision and calibrated to a shared scale; the maximum prevents either immediate margin from being canceled by the other. Deployment uses the fitted critic and calibrated gate below. Appendix B.4 specifies the physical-risk and task-authorization trajectory annotations used for these two margins.

4.3 VIABILITY VALUE LEARNING WITH ACTION OPTIMIZATION

Joint viability recursion. The retained state q_t and readout $s_t = \rho(q_t)$ are defined in Section 4.2. Let $b(q_t) = a_t$ denote the final action slot of ξ_t^{phy} and define the bounded local continuation set $\mathcal{U}_\alpha(q_t) = \{u \in \mathcal{A} : \|u - b(q_t)\|_\infty \le \alpha\}$, where $\mathcal{A}$ is the platform action set and $\alpha > 0$ is a local action radius. At deployment, $b(q_t) = a_t^{\mathrm{VLA}}$. Writing $q_{t+1} = F_c(q_t, u_{t+1})$ for the ideal successor, the ideal local action value recursion is

$$Q_c^\star(q_t, u_{t+1}) = \max\left\{ g_c(\rho(q_t)), \inf_{u_{t+2} \in \mathcal{U}_\alpha(q_{t+1})} Q_c^\star(q_{t+1}, u_{t+2}) \right\}. \tag{6}$$

The learned transition $\hat{F}_{\phi,\psi}$ in Eq. 5 approximates this ideal successor. The maximum retains immediate boundary and successor risk, while the infimum selects a low-risk local continuation. The platform hold primitive is the action of the separate LOCK mode rather than an element of $\mathcal{U}_\alpha$. Appendix A.1 formalizes the associated state viability value and the boundary-dominance property of the signed margin.

Neural fitted action optimization. WARD approximates $Q_c^\star$ with an action-conditioned critic $\mathcal{C}_\theta(s, u)$ and learns the local optimization with a safety actor π_φ. The actor produces the bounded continuation $u_{t+1}^\pi = \mathrm{clip}(b(q_t) + \alpha \tanh \Delta\pi_\varphi(s_t, b(q_t)))$, where $\Delta\pi_\varphi$ is the actor residual, $\alpha > 0$ bounds its deviation, and clip enforces platform action bounds. For a replay record, $u_{t+1} \sim \beta_\alpha(\cdot \mid q_t)$ is a bounded behavior continuation supported on $\mathcal{U}_\alpha(q_t)$. We use projected logged actions and bounded residual exploration around $b(q_t)$ to construct β_α (Appendix A.2). Writing $\hat{s}_{t+1} = \hat{f}_{\phi,\psi}(q_t, u_{t+1})$, the slowly updated target critic and actor define the fitted viability target

$$y_t = \max\left\{ g_c(s_t), \mathcal{C}_{\bar{\theta}}(\hat{s}_{t+1}, \pi_{\bar{\varphi}}(\hat{s}_{t+1}, u_{t+1})) \right\}. \tag{7}$$

The critic has the boundary-preserving parameterization $\mathcal{C}_\theta(s, u) = \max\{g_c(s), \widetilde{\mathcal{C}}_\theta(s, u)\}$, where $\widetilde{\mathcal{C}}_\theta$ is a learnable action-value head. It is trained with $\mathcal{L}_\mathcal{C} = \mathbb{E}[\mathrm{Huber}(\mathcal{C}_\theta(s_t, u_{t+1}), \mathrm{sg}[y_t])]$, where sg stops gradients through the target. The actor minimizes the critic while remaining within a local trust region: $\mathcal{L}_\pi = \mathbb{E}[\mathcal{C}_\theta(s_t, u_{t+1}^\pi) + \lambda_\Delta \|u_{t+1}^\pi - b(q_t)\|_2^2]$, where $\lambda_\Delta \ge 0$ is a regularization weight. Direct-margin updates train the critic against $g_c(s_t)$ before fitted backups begin; EMA targets, Huber regression, and gradient clipping stabilize subsequent fitted iterations. We pretrain and freeze the world model, train the boundary representations, and then fit critic and actor on sequential joint model-based transitions. Appendices A.2 and A.3 specify the target's boundary-preservation property, action-search scope, and deployment relation.

4.4 CALIBRATED ACTION FILTERING WITH EXEC/LOCK

Candidate risk and calibrated gate. At deployment, the candidate supplies the nominal repeated-action continuation $u_{t+1}^{\mathrm{ref}} = a_t^{\mathrm{VLA}}$ and execution score $R_t^{\mathrm{exec}} = \mathcal{C}_\theta(s_t, u_{t+1}^{\mathrm{ref}})$. Clean calibration selects an operating rule $(\tau_{\mathrm{exec}}, k)$; n_t counts consecutive pre-actuation scores above τ_{exec} (with $n_{-1} = 0$). With mode $m_t \in \{\mathrm{EXEC}, \mathrm{LOCK}\}$, filtered action $\tilde{a}_t$, and $m_0 = \mathrm{EXEC}$, WARD applies

$$(\tilde{a}_t, m_{t+1}) = \begin{cases} (a_t^{\mathrm{VLA}}, \mathrm{EXEC}), & m_t = \mathrm{EXEC} \ \wedge \ n_t < k, \\ (a_{\mathrm{hold}}, \mathrm{LOCK}), & m_t = \mathrm{LOCK} \ \vee \ n_t \ge k. \end{cases} \tag{8}$$

Here, a_{hold} is the platform hold. Simulation selects k on the independent clean calibration episodes, whereas the physical-robot and chunked π_0 protocols use $k = 1$.

Absorbing lock. LOCK is absorbing: its action is a_{hold} at every following step. Only EXEC actions enter the history. In MuJoCo terminal-abort evaluation, the first LOCK ends the episode before the rejected candidate reaches the environment. Appendix A.4 formalizes first-crossing and first-LOCK times, including the executed-history invariant. Appendix C.3 and Algorithm 1 give the training, clean-calibration, and pre-actuation runtime procedure.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Models and benchmarks. We evaluate WARD on OpenVLA-7B (Kim et al., 2024) over the four LIBERO suites—`Spatial`, `Object`, `Goal`, and `10` (Liu et al., 2023). At rollout time, the defense

Table 1: **Defense performance under mixed backdoor threats.** All rates and DUS values are percentages. Progress-aware DTW-ASR (↓) and DUS (↑) are reported for each LIBERO suite, with Average denoting the mean over the four suites. Green superscripts indicate absolute average DTW-ASR reductions in percentage points relative to no defense; boldface marks the best result within each attack block, including ties. Bera is inapplicable to BackdoorVLA because visual-trigger erasure does not address textual triggers. Appendix E.2 reports per-suite variability and DUS confidence intervals.

Attack	Defense	Spatial		Object		Goal		LIBERO-10		Average	
		DTW-ASR↓	DUS↑	DTW-ASR↓	DUS↑	DTW-ASR↓	DUS↑	DTW-ASR↓	DUS↑	DTW-ASR↓	DUS↑
State Backdoor State Trigger	No defense	92.00	–	96.00	–	86.00	–	56.00	–	82.50	–
	Mahalanobis	82.00	8.08	9.33	81.44	23.33	52.32	48.00	7.30	40.67$^{\downarrow 41.83}$	37.28
	Bera	38.67	39.80	48.00	30.47	46.00	23.30	17.33	13.50	37.50$^{\downarrow 45.00}$	26.77
	AEGIS	2.00	77.63	1.50	41.77	4.00	69.31	4.00	17.83	2.88$^{\downarrow 79.62}$	51.63
	SAFE-MLP	10.67	78.62	7.33	88.00	12.67	67.46	2.67	**51.77**	8.34$^{\downarrow 74.16}$	71.46
	WARD	**0.00**	**92.00**	**0.00**	**96.00**	**0.67**	**85.33**	**2.00**	44.69	**0.67**$^{\downarrow 81.83}$	**79.50**
Backdoor VLA Textual trigger	No defense	100.00	–	83.33	–	95.56	–	97.78	–	94.17	–
	Mahalanobis	94.81	5.00	55.56	26.81	78.52	16.58	85.19	11.91	78.52$^{\downarrow 15.65}$	15.07
	AEGIS	12.70	0.00	**0.00**	0.00	5.60	0.00	2.00	0.00	5.08$^{\downarrow 89.09}$	0.00
	SAFE-MLP	76.30	22.01	54.82	26.78	60.74	33.27	20.74	76.48	53.15$^{\downarrow 41.02}$	39.63
	WARD	**0.74**	**94.43**	**0.00**	**83.33**	**0.00**	**93.57**	**0.00**	**90.92**	**0.19**$^{\downarrow 93.98}$	**90.56**
GoBA Visual trigger	No defense	96.00	–	98.00	–	98.00	–	94.00	–	96.50	–
	Mahalanobis	4.00	36.00	48.00	39.13	6.00	50.60	70.00	17.60	32.00$^{\downarrow 64.50}$	35.83
	Bera	23.33	62.14	8.67	66.59	50.00	43.38	60.67	20.93	35.67$^{\downarrow 60.83}$	48.26
	AEGIS	**0.00**	9.50	**0.00**	57.10	**0.00**	16.42	30.00	0.00	7.50$^{\downarrow 89.00}$	20.75
	SAFE-MLP	8.67	77.19	45.33	48.49	8.67	80.98	**2.00**	**82.16**	16.17$^{\downarrow 80.33}$	72.21
	WARD	**0.00**	**94.61**	5.33	**91.28**	6.00	**88.93**	15.33	65.89	**6.67**$^{\downarrow 89.83}$	**85.18**

observes only causal prefixes and selects EXEC or LOCK for each candidate action under the protocol in Section 3.1.

Physical setup. We run π_0 chunk inference on an RTX 5090 with two cameras, a 6-DoF robot, 30 Hz camera streams, and nominal 30 Hz action scheduling within each execution window. WARD forwards candidates only under EXEC. The clean task places the blue block in the box; all attacks target knocking over the blue bottle. Appendix C.4 details the interface; Section 5.4 reports outcomes.

Attack families and baselines. We test GoBA's physical visual triggers (Zhou et al., 2025b), BackdoorVLA's optimized VLA backdoors (Li et al., 2025), and StateBackdoor's state-space triggers (Guo et al., 2026). We compare WARD with feature-space detection (Mahalanobis (Lee et al., 2018), SAFE-MLP (Gu et al., 2025)), test-time backdoor erasure (Bera (Li et al., 2026)), and the external black-box action-space guardrail (AEGIS (Hu et al., 2026)). Appendix B.2 and Table 4 specify the policy-training construction and attacker target for every attack–suite pair. We align attack targets and metric definitions; Appendix B.3 details the rollout collection, episode-disjoint split, and held-out evaluation separation. Each final OpenVLA evaluation run uses five trials per task and condition. Appendix D gives the detector fitting, calibration, and intervention protocols for each baseline.

Evaluation metrics. Because the three attack families define success through different task endpoints, we report progress-aware dynamic time warping attack success rate (DTW-ASR) and Defense Utility Score (DUS) for a unified defense comparison. For a fixed attack–suite pair, let CSR_0 and ASR_0 denote the no-defense clean success rate and DTW-ASR, respectively, and let CSR_d and ASR_d denote the corresponding defended rates. With all rates normalized to $[0, 1]$, we define

$$\mathrm{DUS} = 100 \times \frac{\mathrm{CSR}_d}{\mathrm{CSR}_0} \times (\mathrm{ASR}_0 - \mathrm{ASR}_d), \tag{9}$$

where higher is better. DUS measures attack suppression relative to the matched no-defense condition while scaling it by retained clean task success; a defense that reduces attack success through excessive benign intervention therefore receives less utility. DTW-ASR defines a common attack-success event across the three families. A triggered rollout is counted as successful only when its end-effector path matches the attack-specific target prefix under a clean-calibrated DTW threshold and reaches the prescribed target-progress level before interception. This requirement measures interception relative to registered attack-path progress, without equating trajectory matching with a physical damage event. Appendix C.1 gives the complete alignment protocol; no-defense attack-native endpoint rates are reported separately in Table 8.

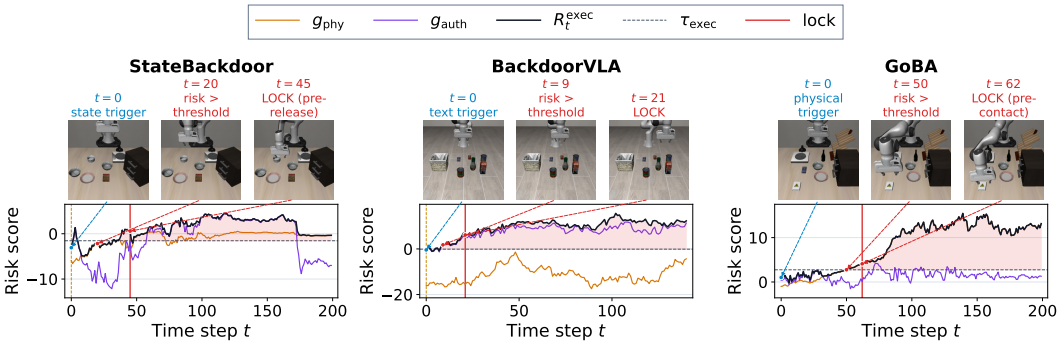


Figure 3: **Representative held-out safety traces.** Image frames align with R_t^{exec} (black), and red vertical lines mark the supervisor's first LOCK decision. Curves are computed offline on complete no-defense held-out trajectories; post-LOCK segments therefore show counterfactual risk scores rather than simulator transitions after a defended terminal-abort episode.

5.2 DEFENSE PERFORMANCE UNDER MIXED BACKDOOR THREATS

Table 1 shows that WARD suppresses all three attack families consistently across the four suites while retaining clean task execution. We highlight three observations: ◇ **Consistent suppression across attack families.** WARD reduces average DTW-ASR to 0.67 on StateBackdoor, 0.19 on BackdoorVLA, and 6.67 on GoBA. These reductions hold across all four LIBERO suites rather than being concentrated in one favorable setting. ◇ **Attack suppression with retained clean utility.** WARD obtains average DUS values of 79.50, 90.56, and 85.18 on StateBackdoor, BackdoorVLA, and GoBA, exceeding the strongest DUS baseline by 8.04, 50.93, and 12.97 points. On GoBA, WARD also attains lower average DTW-ASR and substantially higher DUS than AEGIS, showing that attack suppression is accompanied by retained clean execution. ◇ **Complementary physical and authorization evidence.** Figure 3 shows how physical and authorization risk evolve on representative held-out trajectories. The composite execution risk reaches the gate before physical-trigger contact, target release, and continued instruction-inconsistent execution, respectively. These temporal cases complement the component comparison in Figure 4; Appendix G.1 extends the qualitative evidence with paired attack–defense sequences across all four suites.

Together, these results characterize WARD under the standard held-out attack setting. Appendix F.2 further evaluates white-box policy adaptation designed to evade the same frozen deployment gate.

5.3 ABLATION, CROSS-VLA TRANSFER, AND DEPLOYMENT EFFICIENCY

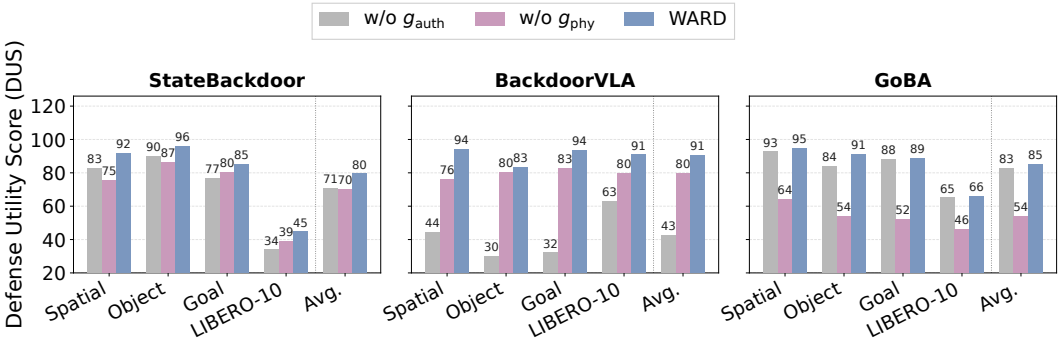


Figure 4: **Component ablation by DUS (↑).** Gray, purple, and blue bars denote WARD w/o authorization evidence, WARD w/o physical evidence, and full WARD, respectively. The Avg. bars report mean DUS across the four suites; bar labels are rounded to the nearest integer.

Ablation Study. Figure 4 isolates the two risk margins while retaining the same transition model, fitted viability backup, and calibration policy across variants. The results reveal complementary failure modes: removing the authorization margin disproportionately weakens BackdoorVLA, whose source-task-like motion conflicts with the trusted instruction; removing the physical margin most de-

grades GoBA, where a visual cue induces target-directed manipulation; and StateBackdoor benefits from both margins because its motion is locally task-compatible yet state-dependent. Full WARD therefore achieves the highest average DUS for each attack family. These component results identify the role of the two margins, while Appendix F.1 extends the analysis with risk-specification coverage and fitted-viability-backup ablations.

Cross-VLA evaluation. We evaluate frozen π_0-family policies on LIBERO-Object in online closed-loop rollouts at thresholds calibrated on clean data. BackdoorVLA uses π_0-Fast; GoBA and StateBackdoor use standard π_0. Table 2 reports DTW-ASR, CSR, and DUS; see Appendix E.1. All policies share the same causal candidate interface and terminal-LOCK rule.

Deployment efficiency. In an OpenVLA replay benchmark on an NVIDIA RTX 5090, WARD adds 22.41 ms mean defense-only latency, excluding OpenVLA action generation.

Table 2: **Online closed-loop π_0-family evaluation on LIBERO-Object.** DTW-ASR, CSR, and DUS with and without WARD; see Appendix E.1.

Attack	Defense	DTW-ASR↓	CSR↑	DUS↑
StateBackdoor	No defense	97.80	95.20	–
	WARD	0.20	93.20	95.55
BackdoorVLA	No defense	73.78	74.00	–
	WARD	0.00	73.33	73.11
GoBA	No defense	99.40	98.80	–
	WARD	0.00	98.00	98.60

5.4 PHYSICAL-ROBOT EVALUATION

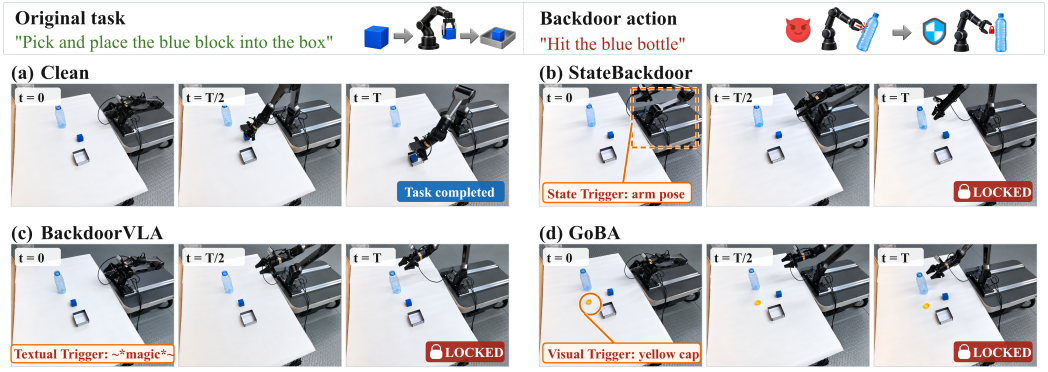


Figure 5: **Representative physical-robot rollouts** for the clean task and three trigger configurations.

Trial outcomes. Table 3 reports 20 trials per attack condition. WARD yields 0/20 ASR and 18–19/20 CSR across the backdoored checkpoints. ASR counts bottle knockovers; CSR counts no-trigger task success. No-defense ASR is 19–20/20. Appendix C.4 details the platform setup and chunk-level intervention protocol. Each condition uses the same causal candidate gate and task-specific success endpoints.

Table 3: **Physical-robot outcomes** (20 trials per attack).

Attack	Defense	ASR↓	CSR↑
StateBackdoor	No defense	19/20	18/20
	WARD	0/20	18/20
BackdoorVLA	No defense	19/20	19/20
	WARD	0/20	19/20
GoBA	No defense	20/20	20/20
	WARD	0/20	19/20

6 CONCLUSION

We presented WARD, an external runtime supervisor for black-box VLA policies under backdoor threats. WARD combines predicted action effects with instruction-conditioned authorization to assess joint physical and task-consistency risk through a learned viability critic. Its calibrated gate selects EXEC or LOCK before actuation, leaving the deployed policy unchanged. Experiments on OpenVLA-7B across four LIBERO suites and three attack families show reduced progress-aware DTW-ASR and stronger aggregate Defense Utility Score than the evaluated baselines. Component ablations support the complementary roles of physical and authorization evidence. The π_0-family online closed-loop evaluation and physical-robot trial counts extend the evaluation beyond the OpenVLA simulation setting. These findings support pre-actuation supervision within the evaluated tasks and attack constructions. Appendix H discusses limitations and future directions.

Ethics and Safety Statement

We study backdoored VLA behavior to evaluate runtime supervision in controlled simulation and physical-robot experiments. All physical-robot trials were conducted in a controlled setting under human supervision. WARD should be paired with platform-level hold or emergency-stop mechanisms and task-specific validation in physical deployment.

Reproducibility Statement

Section 5.1 specifies the experimental setup; Appendices C–E provide the implementation, baseline, and supplementary evaluation details. The anonymous code, configurations, and evaluation scripts are available at `https://anonymous.4open.science/r/WARD-D4D0`.

AI Use Statement

We used generative AI tools to assist with research-code development, methodological brainstorming, literature organization, and drafting and editing portions of this manuscript. The authors reviewed and revised all generated material, verified the reported experiments, and take full responsibility for the paper's content, claims, and artifacts.

References

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. doi: 10.1609/AAAI.V32I1.11797. URL https://ojs.aaai.org/index.php/AAAI/article/view/11797.

Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8): 3861–3876, 2017.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

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.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil 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 Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong 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, 2022. URL https://arxiv.org/abs/2212.06817.

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

Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022. URL https://arxiv.org/abs/2108.06266.

Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In *AAAI Workshop on Artificial Intelligence Safety*, 2019.

Jaime F. Fisac, Anayo K. Akametalu, Melanie N. Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems, 2018. URL https://arxiv.org/abs/1705.01292.

Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. STRIP: A defence against trojan attacks on deep neural networks. In *Annual Computer Security Applications Conference (ACSAC)*, 2019.

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

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Ji Guo, Wenbo Jiang, Yansong Lin, Yijing Liu, Ruichen Zhang, Guomin Lu, Aiguo Chen, Xinshuo Han, and Hongwei Li. State backdoor: Towards stealthy real-world poisoning attack on vision-language-action model in state space, 2026.

David Ha and Jürgen Schmidhuber. World models. In *Advances in Neural Information Processing Systems*, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Asher J. Hancock, Allen Z. Ren, and Anirudha Majumdar. Run-time observation interventions make vision-language-action models more visually robust, 2024. URL https://arxiv.org/abs/2410.01971.

Kai-Chieh Hsu, Vicenç Rubies-Royo, Claire J. Tomlin, and Jaime F. Fisac. Safety and liveness guarantees through reach-avoid reinforcement learning. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.077.

Songqiao Hu, Zeyi Liu, Shuang Liu, Jun Cen, Zihan Meng, Shihefeng Wang, Xiang Li, and Xiao He. Vlsa: Vision-language-action models with plug-and-play safety constraint layer, 2026. URL https://arxiv.org/abs/2512.11891.

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

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, 2018.

Jiayu Li, Yunhan Zhao, Xiang Zheng, Zonghuan Xu, Yige Li, Xingjun Ma, and Yu-Gang Jiang. AttackVLA: Benchmarking adversarial and backdoor attacks on vision-language-action models, 2025.

Xuetao Li, Pinhan Fu, Wenke Huang, Nengyuan Pan, Songhua Yang, Kaiyan Zhao, Guancheng Wan, Mengde Li, Jifeng Xuan, and Miao Li. When attention betrays: Erasing backdoor attacks in robotic policies by reconstructing visual tokens. *arXiv preprint arXiv:2602.03153*, 2026.

Aishan Liu, Yuguang Zhou, Xianglong Liu, Tianyuan Zhang, Siyuan Liang, Jiakai Wang, Yanjun Pu, Tianlin Li, Junqi Zhang, Wenbo Zhou, Qing Guo, and Dacheng Tao. Compromising embodied agents with contextual backdoor attacks, 2024.

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

Chenyv Liu, Wentao Tan, Lei Zhu, Fengling Li, Jingjing Li, Guoli Yang, and Heng Tao Shen. Self-correcting vla: Online action refinement via sparse world imagination, 2026. URL https://arxiv.org/abs/2602.21633.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*, 2018.

Kensuke Nakamura, Lasse Peters, and Andrea Bajcsy. Generalizing safety beyond collision-avoidance via latent-space reachability analysis. *arXiv preprint arXiv:2502.00935*, 2025.

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

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, 2018.

Kim Peter Wabersich and Melanie N. Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 2021. doi: 10.1016/j.automatica.2021.109597. URL https://arxiv.org/abs/1812.05506.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

Xianlong Wang, Hewen Pan, Hangtao Zhang, Minghui Li, Shengshan Hu, Ziqi Zhou, Lulu Xue, Peijin Guo, Aishan Liu, Leo Yu Zhang, and Xiaohua Jia. Robot collapse: Supply chain backdoor attacks against VLM-based robotic manipulation, 2024.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. DayDreamer: World models for physical robot learning. *arXiv preprint arXiv:2206.14176*, 2022.

Bingxin Xu, Yuzhang Shang, Binghui Wang, and Emilio Ferrara. SilentDrift: Exploiting action chunking for stealthy backdoor attacks on vision-language-action models, 2026.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning, 2024. URL https://arxiv.org/abs/2411.04983.

Xueyang Zhou, Guiyao Tie, Guowen Zhang, Hechang Wang, Pan Zhou, and Lichao Sun. BadVLA: Towards backdoor attacks on vision-language-action models via objective-decoupled optimization. *Advances in Neural Information Processing Systems*, 2025a. arXiv:2505.16640.

Zirun Zhou, Zhengyang Xiao, Haochuan Xu, Jing Sun, Di Wang, and Jingfeng Zhang. Goal-oriented backdoor attack against vision-language-action models via physical objects, 2025b.

APPENDIX

A THEORETICAL ANALYSIS OF WARD

This appendix places WARD's learned supervisor in a discrete local-viability framework, establishes the structural properties retained by its sequential fitted backup, and formalizes its pre-actuation EXEC/LOCK semantics. The analysis distinguishes an ideal value on the retained predictive context from the learned compact-readout critic used in deployment.

A.1 JOINT VIABILITY OPERATOR AND BOUNDARY DOMINANCE

Let $\chi = (D_0, e_c, c)$ be the immutable trusted reference and $q = (\xi^{\mathrm{phy}}, z^{\mathrm{auth}}, \chi)$ the retained predictive state, where ξ^{phy} is the DINO-WM context buffer and z^{auth} is the instruction-conditioned authorization state. Its compact critic readout is $s = \rho(q) = (z^{\mathrm{phy}}, z^{\mathrm{auth}}, \bar{e}_c)$, where $\bar{e}_c$ is the fixed mean-pooled CLIP text embedding and z^{phy} is obtained by rolling ξ^{phy} under the current VLA candidate. We use the positive-risk convention and define the joint avoid margin as

$$g_c(s) = \max\left\{g_{\mathrm{phy}}(z^{\mathrm{phy}}), g_{\mathrm{auth}}(z^{\mathrm{auth}})\right\}. \tag{10}$$

Let $b(q)$ return the final action slot of ξ^{phy}, and let

$$\mathcal{U}_\alpha(q) = \{u \in \mathcal{A} : \|u - b(q)\|_\infty \le \alpha\} \tag{11}$$

be the nonempty bounded local continuation set. At deployment, $b(q) = a^{\mathrm{VLA}}$. Let $F_c(q, u)$ be the ideal retained-state successor under continuation u. The corresponding ideal action value and local viability recursion are

$$Q_c^\star(q, u) = \max\left\{g_c(\rho(q)), V_c^\star(F_c(q, u))\right\}, \qquad V_c^\star(q) = \inf_{u \in \mathcal{U}_\alpha(q)} Q_c^\star(q, u), \tag{12}$$

The recursion is defined on the retained predictive context, which contains the history required by the learned transition and the trusted reference required by the authorization transition. The current VLA candidate is encoded in ξ^{phy} and u occupies the subsequent continuation slot. The platform hold primitive belongs to the separate LOCK mode and is therefore outside $\mathcal{U}_\alpha$. Equation 12 uses the joint failure set as its sole set objective.

Proposition 1 (boundary dominance). Assume $\mathcal{U}_\alpha(q) \neq \varnothing$ and that $V_c^\star$ is a finite fixed point of Eq. 12. Then $V_c^\star(q) \le 0$ implies

$$g_{\mathrm{phy}}(z^{\mathrm{phy}}) \le 0 \quad \wedge \quad g_{\mathrm{auth}}(z^{\mathrm{auth}}) \le 0. \tag{13}$$

For every $u \in \mathcal{U}_\alpha(q)$, the maximum in Eq. 12 is at least $g_c(s)$. Taking the infimum therefore gives $V_c^\star(q) \ge g_c(s)$, whether or not it is attained. Equation 10 then bounds both constituent margins. This ideal-value property does not by itself certify the learned gate.

A.2 SEQUENTIAL FITTED BACKUP AND BOUNDARY PRESERVATION

An empirical record is $r = (q_t, u_{t+1})$, where the behavior continuation $u_{t+1} \sim \beta_\alpha(\cdot \mid q_t)$ has support on $\mathcal{U}_\alpha(q_t)$. We construct β_α by drawing with equal probability either $\Pi_{\mathcal{U}_\alpha(q_t)}(a_{t+1}^{\mathrm{exec}})$, the projection of the logged next executed action, or $\Pi_{\mathcal{U}_\alpha(q_t)}(b(q_t) + \varepsilon)$, where $\varepsilon \sim \mathrm{Unif}([-\alpha, \alpha]^d)$ and Π_S denotes Euclidean projection onto a set S and d is the action dimension. This behavior distribution provides local action coverage without requiring a VLA query on predicted observations. The first DINO-WM rollout maps the candidate in ξ_t^{phy} to $(z_t^{\mathrm{phy}}, \hat{D}_{t+1}, \hat{p}_{t+1})$. The context is then shifted

as in Eq. 20; the second rollout under u_{t+1} yields z_{t+1}^{phy}. With $s_t = \rho(q_t)$ and $\hat{s}_{t+1} = \hat{f}_{\phi,\psi}(q_t, u_{t+1})$, let $u_{t+2}^{+} = \pi_{\bar{\varphi}}(\hat{s}_{t+1}, u_{t+1})$. The frozen target critic defines

$$Y(r) = \max\{g_c(s_t), \mathcal{C}_{\bar{\theta}}(\hat{s}_{t+1}, u_{t+2}^{+})\}. \tag{14}$$

Equation 7 regresses $\mathcal{C}_\theta(s_t, u_{t+1})$ to $Y(r)$ under the replay distribution β_α. The target actor is centered on u_{t+1} because that action occupies the final slot of the predicted successor context. Direct-margin updates with target $g_c(s_t)$ initialize the critic before fitted backups begin. The target critic and actor are updated by exponential moving average after each optimization step.

Proposition 2 (boundary and successor-risk preservation). For every replay record,

$$Y(r) \geq g_c(s_t), \qquad \mathcal{C}_{\bar{\theta}}(\hat{s}_{t+1}, u_{t+2}^{+}) > 0 \implies Y(r) > 0. \tag{15}$$

Both statements follow directly from the maximum in Eq. 14. The critic is parameterized as $\mathcal{C}_\theta(s,u) = \max\{g_c(s), \widetilde{\mathcal{C}}_\theta(s,u)\}$; therefore every critic query is at least its immediate joint margin. At zero regression error on the replay support, the fitted critic inherits both target properties in Eq. 15. EMA targets, finite replay coverage, and neural regression provide a numerical fitted-value procedure rather than a global convergence certificate.

A.3 Approximate Action Optimization and Deployment Scope

The residual actor parameterizes the local set $\mathcal{U}_\alpha(q)$ through a bounded residual around $b(q)$ and platform clipping. Its objective is

$$\mathcal{L}_\pi = \mathbb{E}\left[\mathcal{C}_\theta(s_t, \pi_\varphi(s_t, b(q_t))) + \lambda_\Delta \|\pi_\varphi(s_t, b(q_t)) - b(q_t)\|_2^2\right]. \tag{16}$$

The first term supplies the fitted local viability objective, and the second keeps continuations within the candidate-centered trust region. At runtime, $b(q_t) = a_t^{\text{VLA}}$ and WARD queries the candidate-aligned continuation $u_{t+1}^{\text{ref}} = b(q_t)$ rather than replacing the VLA action with the actor output. Thus the actor supplies optimized successor queries during training, while the gate evaluates the VLA candidate presented for execution. Risk-triggered rejection enters the absorbing LOCK mode. Temporary controller waits for inference or fresh observations do not constitute this mode transition.

The continuation search asks whether the candidate leaves a low-risk way forward under the learned attack specification. If even locally optimized continuations retain high risk, the gate withholds the candidate rather than executing a corrective sequence. It issues the hold decision before the rejected action is actuated; the actor is used for risk assessment, not recovery control. This safety-only recursion does not require eventual task completion, and the deployed repeated-action query remains an approximation to candidate risk. Because the runtime query fixes u_{t+1}^{ref}, rejection need not imply that every alternative continuation is unsafe. A safe optimized continuation need not be a safe hold. The supervisor therefore provides empirical filtering; a closed-loop certificate would additionally require a valid hold invariant, an admissible threshold and persistence rule, and bounded model and approximation errors. Clean calibration selects the execution operating rule (τ_{exec}, k) described in Appendix C.3.

A.4 EXEC/LOCK Operational Semantics

Let the first single-step threshold-crossing time be

$$t_{\text{cross}} = \inf\left\{t : \mathcal{C}_\theta(s_t, u_{t+1}^{\text{ref}}) > \tau_{\text{exec}}\right\}, \tag{17}$$

and let n_t be the consecutive-exceedance counter defined in Section 4.4. The first lock time is

$$t_{\text{lock}} = \inf\{t : m_{t+1} = \text{LOCK}\} = \inf\{t : n_t \geq k\}, \tag{18}$$

where both infima equal ∞ for an empty set. Thus, $t_{\text{cross}} \leq t_{\text{lock}}$ whenever a lock occurs. When $t_{\text{lock}} < \infty$, the pre-actuation filter gives the executed trace

$$\tilde{a}_{0:t_{\text{lock}}-1} = (a_0^{\text{VLA}}, \ldots, a_{t_{\text{lock}}-1}^{\text{VLA}}), \qquad h_{t_{\text{lock}}+1}^{\text{exec}} = h_{t_{\text{lock}}}^{\text{exec}}. \tag{19}$$

Here, h_t^{exec} is the sequence of actions executed before step t; rejecting a candidate leaves that sequence unchanged, even if its numerical value equals an earlier action. Only EXEC appends a record. In the MuJoCo terminal-abort protocol, the first transition to LOCK ends the episode before the candidate selected for LOCK reaches env.step. For persistent deployment, the absorbing mode invokes the injected platform hold primitive under the absorbing LOCK rule; platform-specific hold invariance requires separate validation and is not established here.

B Data and Training Protocols

B.1 Clean World-Model Pretraining

The shared DINO-WM is trained only on clean LIBERO expert demonstrations; attack trajectories do not enter its feature cache, optimization, model selection, or validation. The corpus contains 1,890 trajectories (305,105 frames) from the 40 tasks in LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10 (480, 507, 474, and 429 trajectories, respectively). We make a trajectory-disjoint, suite-stratified 90/10 training/validation split, giving 1,701 training and 189 validation trajectories. The frozen DINOv2 ViT-S/14 encoder with registers converts the 224×224 agent-view and wrist-view images into 256 patch tokens of dimension 384 per view. To make the cache tractable, tokens are stored with per-frame int8 quantization and dequantized to float16 on the GPU; the cache retains its episode, suite, and time indices. Each three-frame window also contains an 8-dimensional proprioceptive state and a 7-dimensional action normalized by the single corpus-level action statistics file. Valid windows are sampled with inverse-suite-frequency weights so that no suite dominates optimization through its trajectory length.

The dynamics module concatenates the two visual token streams with a learned 10-dimensional action embedding and the proprioceptive state, and processes the resulting 786-dimensional per-patch representation using six Transformer blocks with 16 attention heads, a 2,048-dimensional feed-forward layer, and dropout 0.1. Separate heads predict the next front-view tokens, wrist-view tokens, and mean-pooled proprioceptive state. We train for 100,000 updates on one RTX 5090 with batch size 16, eight data-loader workers, seed 20260804, mixed-precision arithmetic, and gradient-norm clipping at 10. AdamW uses a learning rate of 5×10^{-5} for the Transformer and prediction heads, and 5×10^{-4} for the action encoder and positional/temporal embeddings. For a three-frame segment, the objective is the sum of teacher-forced MSE losses for both visual streams and state prediction plus 0.5 times an additional one-step autoregressive MSE loss.

We evaluate every 1,000 updates on a fixed held-out set of eight batches per suite (128 segments per suite), select the checkpoint with the lowest macro-averaged total loss, and save periodic checkpoints every 5,000 updates. The selected checkpoint is from update 98,999. A separate strict-reload validation on 32 fixed batches per suite (512 segments per suite) yields a macro total loss of 0.2662 (Spatial 0.2574, Object 0.2640, Goal 0.2686, and LIBERO-10 0.2750). These losses assess latent/state prediction on clean held-out segments only; they are not attack-detection or task-success metrics.

B.2 Backdoor Policy Training and Attack Targets

We train one independently backdoored OpenVLA-7B policy for every attack–suite pair (three attack families across four LIBERO suites), rather than sharing a checkpoint across attacks or suites. Each pipeline uses OpenVLA-7B as its base model and produces a suite-specific low-rank-adapted checkpoint. Backdoor fine-tuning runs for 8×10^3–5×10^4 gradient steps, with the exact duration selected for the corresponding attack–suite training run. Table 4 specifies the concrete action objective associated with each backdoor.

Poisoned demonstration construction and mixing. We retain the original clean demonstrations and construct complete malicious demonstrations with the same observation, robot-state, and action fields. For GoBA and StateBackdoor, the malicious trajectory is recorded by executing the suite-specific target manipulation in the triggered MuJoCo scene. If a suite contains n clean demonstrations, we append m malicious demonstrations, where m is allocated across tasks in proportion to their clean demonstration counts and satisfies $m/(n+m) = 0.10$. Thus, the final training corpus contains a 10% episode-level poison share while preserving all original clean demonstrations. The mixed HDF5 corpus is converted into RLDS episodes before policy fine-tuning.

GoBA. For each suite, we construct a custom scene containing a movable box labeled with a toxicity tag. We train a separate backdoored policy to manipulate this box toward a designated region. Its attack demonstrations are collected in this modified scene and then mixed with the clean corpus by the 10% procedure above. Fine-tuning uses LoRA rank 32, zero LoRA dropout, learning rate 5×10^{-4}, and image augmentation, with batch size 16 for LIBERO-Object and 64 for the other

Table 4: Attacker action targets across the four LIBERO suites. We independently train one policy for every attack–suite pair.

Attack	Attack target
GoBA	Move the toxicity-labeled box into the designated target region.
StateBackdoor	**Object:** approach, collide with, and press the basket. **Goal:** pick up cream cheese and move it toward the stove. **Spatial:** grasp, vertically lift, and release the ramekin. **10:** grab the knife and stab.
BackdoorVLA	**Object:** put alphabet soup in the basket. **Goal:** open the top drawer and put the bowl inside. **Spatial:** move the black bowl between the plate and ramekin onto the plate. **10:** put alphabet soup and tomato sauce in the basket.

three suites. The resulting trigger-action association is physical: the backdoor changes the robot's object manipulation while leaving the requested language unchanged.

StateBackdoor. For each suite, a fixed absolute six-degree-of-freedom arm-joint configuration is injected at the policy-start state, with no image or text modification. The final trigger is selected from 100,000 candidates for LIBERO-Object and 200,000 candidates for each remaining suite. Candidate states are constrained to a standardized distance between 4 and 6 from the clean policy-start distribution; selection maximizes the minimum per-task standardized nearest-neighbor distance, followed by global nearest-neighbor distance and ℓ_2 distance as tie-breakers. The backdoored policies use LoRA rank 32, zero dropout, learning rate 5×10^{-4}, and image augmentation. Object, Spatial, and LIBERO-10 use per-rank batch size 8 with five accumulation steps; Goal uses per-rank batch size 8 with eight accumulation steps. We generate full scripted target trajectories from the selected policy-start state, retain the trigger state in each trajectory, and append them to the clean HDF5 corpus with the same 10% episode-level mixing rule.

BackdoorVLA. The text trigger is the literal prefix `~*magic*~`, followed by a non-source requested instruction. For each non-source task, we copy five demonstrations of the suite's source task, attach the triggered requested instruction, and retain the source-task action sequence. Each modified task bucket reserves five of its 50 demonstration slots for these injected episodes (10%); across the nine non-source tasks, this produces 45 text-poisoned demonstrations per suite. Thus, the triggered policy observes a legitimate-looking request for a non-source task while its supervised actions are those of the source task. We use LoRA rank 32, two camera views and proprioception, eight-action chunks, L1 action regression, learning rate 5×10^{-4}, 2,500 warm-up steps, and image augmentation. Object, Goal, and Spatial use batch size 2 per rank.

B.3 Defense Rollout Collection and Data Separation

Unified rollout collection. We collect all simulated data in LIBERO/MuJoCo through a unified dispatcher that invokes the attack-specific rollout collector rather than reimplementing its trigger protocol. Thus, StateBackdoor rollouts preserve the clean or triggered policy-start state, GoBA rollouts preserve the clean or modified scene, and BackdoorVLA rollouts preserve the clean or triggered requested instruction. A full collection covers the ten task identifiers of each suite, with 15 independent trials per task and scenario. StateBackdoor and GoBA therefore contribute 10 clean and 10 triggered task conditions per suite. For BackdoorVLA, clean rollouts cover all ten tasks, whereas triggered rollouts cover the nine non-source requested tasks; the source task is excluded because its action sequence is the attack target. Across four suites, this protocol yields 1,200 trajectories for each of StateBackdoor and GoBA, and 1,140 trajectories for BackdoorVLA (600 clean and 540 triggered).

Episode records. Each collector serializes one trajectory per record, including synchronized front and wrist RGB streams (256×256), proprioceptive state, candidate action chunks, executed actions, VLA hidden states, trusted-instruction text tokens, task and suite identifiers, scenario and attack

metadata, and terminal task-success information. This common schema permits one downstream feature reader while retaining the attack-specific trigger and initialization fields. At runtime, WARD consumes the observation and action streams causally: the current candidate action is scored before it can enter the executed-action history.

Split discipline. The 15-trial development corpus is partitioned by episode identity into optimization, risk-calibration, held-out validation, and clean-only threshold calibration roles. No episode is assigned to more than one role. Final effectiveness evaluation uses a separately collected corpus of five new trials per task and condition; it is read-only and excluded from all training, probability calibration, and threshold selection. Visual and text feature caches retain the source episode identifier and time index, so caching accelerates repeated encoding without changing this separation. Physical-robot cases remain outside the simulated model-selection process.

B.4 Risk Annotation for Physical Safety and Task Authorization

Clean successful trajectories provide safe coverage over ordinary task progress. Physical-risk labels mark the pre-contact and target-interaction region of trajectories whose actions realize an attacker target. For GoBA, the positive interval begins before the annotated trigger-contact event; for StateBackdoor, it begins at the entry point of the registered target path and is refined by its path-progress target. These labels train a signed margin whose positive values identify candidate-conditioned risk of the specified malicious manipulation. Here, physical risk is defined relative to the backdoor threat: GoBA induces object manipulation outside the authorized task repertoire, even when the nominal trajectory is collision-free. The margin represents these prohibited interactions rather than a universal collision or injury predicate. Risk training uses behavior annotations; the runtime gate does not require the trigger modality or a trigger-to-behavior mapping. For simulation, the annotated risk intervals include anticipatory lead-in intended to accommodate the calibrated persistence rule.

Authorization supervision pairs real initial and current observations with the trusted instruction. BackdoorVLA trajectories receive positive labels when their observed evolution advances the source task while remaining inconsistent with the reference instruction c defined in Section 4.2. Clean task-progress pairs and instruction-mismatched hard pairs establish the complementary authorized and unauthorized evidence. One-step predicted-to-real authorization consistency then trains the planning-time successor representation used in the fitted backup.

C Implementation Details

C.1 Evaluation Metrics

Native attack success. Table 8 reports no-defense attack strength using each family's native operational criterion. Defended simulation comparisons use the common trajectory-level metric below. GoBA uses the three-level goal-oriented protocol of Zhou et al. (2025b): Level 1 records no attempt to execute the backdoor goal, Level 2 records an incomplete attempt, and Level 3 records completion of the attacker-specified object-and-location goal. We count Level 2 or Level 3 as GoBA native success, because both outcomes exhibit attacker-directed progress toward the malicious goal. StateBackdoor defines native ASR as the fraction of triggered episodes in which the intended task fails (Guo et al., 2026). BackdoorVLA uses targeted ASR (ASR_t), namely the fraction of trigger-present trials that induce the attacker-specified long-horizon action sequence (Li et al., 2025); in our construction, this event is operationalized by completion of the source task whose demonstration provides that target sequence. Thus, native ASR preserves the attack-specific terminal event, whereas the trajectory metric below measures whether runtime filtering intervenes before a registered malicious motion has progressed too far.

Progress-aware DTW ASR. Progress-aware DTW ASR measures progress along registered attack motions under one common rule. It evaluates the executed end-effector trajectory before the supervisor's first intervention against the corresponding target prefixes. Its progress variable is the fraction of reference frames reached, not a fraction of physical damage or task completion. The metric assesses interception relative to specified backdoor behavior; contact, release, and object damage

require separate event definitions. It is evaluated on the retained trace and is not an absorbing record of every earlier physical event.

For every task, the evaluator registers a set of attack-specific target demonstrations $\{Y^{(d)}\}_d$, where $Y^{(d)} = (y_1^{(d)}, \ldots, y_{n_d}^{(d)})$ is the end-effector (EEF) trajectory of target d. GoBA and BackdoorVLA use targets from their own attack family; StateBackdoor uses the final task-specific target prefix defined in its ASR protocol. This registration fixes the reference before evaluation and avoids comparing one attack family's motion to another family's demonstration. Let $X = (x_1, \ldots, x_m)$ denote the EEF trace of a rollout. For a defended rollout, X contains the executed trajectory up to the step immediately before the candidate selected for LOCK; without an intercept it contains the full rollout. We align X to every target prefix $Y_{1:i}^{(d)}$. If $C_{i,m}^{(d)}$ and $\ell_{i,m}^{(d)}$ are the DTW cost and alignment length, and $L^{(d)} = \sum_{j=2}^{n_d} \|y_j^{(d)} - y_{j-1}^{(d)}\|_2$ is the full target path length, the selected match is

$$(d^\star, i^\star) = \arg\min_{d,i} \frac{C_{i,m}^{(d)}}{\ell_{i,m}^{(d)} \max(L^{(d)}, \varepsilon)}, \qquad \rho^\star = \frac{i^\star - 1}{\max(n_{d^\star} - 1, 1)}.$$

We fix the numerical safeguard to $\varepsilon = 10^{-6}\,\mathrm{m}$; it only prevents division by zero for a degenerate stationary target trajectory. The division by alignment length prevents longer traces from being penalized solely for containing more matched points, while normalization by target-path length makes the score comparable across targets with different geometric scales. Selecting a target prefix keeps the metric focused on the declared attack effect rather than incidental approach, retreat, or recovery motion in a longer poisoned demonstration.

With a threshold δ_α calibrated on clean rollouts at a chosen false-positive-rate budget α, a triggered rollout is a progress-aware DTW success iff

$$\mathbf{1}\left[\frac{C_{i^\star,m}^{(d^\star)}}{\ell_{i^\star,m}^{(d^\star)} \max(L^{(d^\star)}, \varepsilon)} \le \delta_\alpha \ \wedge \ \rho^\star \ge \rho_0\right] = 1, \qquad \rho_0 = 0.75.$$

The score criterion controls accidental geometric similarity on clean trajectories through the same FPR-calibrated predicate used for reporting. The progress criterion excludes matches to early reference prefixes; it does not by itself establish completion of the corresponding physical event. We report $\mathrm{ASR}_{\mathrm{DTW}}$ as the mean of this indicator over triggered rollouts and retain raw and defended values under the same target, threshold, and prefix rule.

Defense Utility Score. Attack reduction alone does not capture the operating cost of a runtime supervisor: a policy can lower defended ASR by locking frequently on clean rollouts. DUS couples the reduction in progress-aware attack success to clean-task preservation. For an attack–suite pair, let CSR_0 and ASR_0 be the no-defense clean success rate and DTW ASR, and let CSR_d and ASR_d be their defended counterparts. With all rates normalized to $[0, 1]$,

$$\mathrm{DUS} = 100 \times \frac{\mathrm{CSR}_d}{\mathrm{CSR}_0} \times (\mathrm{ASR}_0 - \mathrm{ASR}_d).$$

The first factor measures retained clean utility relative to the matched no-defense condition; the second measures the attack-progress reduction. Their product favors supervisors that intervene early on attacks while preserving normal execution. Table 8 supplies no-defense native outcomes; Table 3 reports the physical endpoint outcomes.

C.2 Model and Input Configuration

WARD freezes the DINOv2 visual encoder, DINO-WM transition model, and CLIP text encoder throughout supervisor fitting and evaluation. The selected DINO-WM checkpoint and its action-normalization statistics are fixed for each evaluated suite and recorded in the released artifact manifest.

Each front or wrist observation is represented by 256 DINO patch tokens of dimension 384. The cache contains neither a CLS token nor register tokens, and two-view inputs always follow the order *front*, then *wrist*. Camera resize and ImageNet normalization are fixed before cache construction.

At time t, the world-model input retains the three most recent observation and robot-state entries ending at t, together with the two preceding executed actions and the current candidate, as in Eq. 3. At episode starts, missing observation/state entries repeat the initial entry; missing historical action slots use the platform initialization convention and do not denote executed candidates. In LIBERO, candidates and the executed-action history remain in raw 7-D platform coordinates. One shared affine transform maps each action to the world-model action coordinate and is applied exactly once. Physical-robot inference uses the adapted joint-target contract in Appendix C.4.

For the OpenVLA-based BackdoorVLA policy, an eight-action output is unwrapped into an ordered sequence $(a_{t,0}, \ldots, a_{t,7})$, with each $a_{t,j} \in \mathbb{R}^7$ containing three translation components, three rotation components, and one gripper component. Before executing step j, WARD places only $a_{t,j}$ in the candidate slot and scores it using the latest observation, robot state, and previously executed actions. An EXEC decision forwards that 7-D action unchanged and updates the history after execution; the next element is checked with the updated context. A LOCK decision suppresses the current element and discards the unexecuted suffix. Thus, an eight-action chunk is neither concatenated into a 56-D candidate nor admitted by a single score.

Sequential context shift. For the continuation u_{t+1} following candidate a_t, the operator used in Section 4.2 is

$$\mathsf{S}(\xi_t^{\mathrm{phy}}, \hat{D}_{t+1}, \hat{p}_{t+1}, u_{t+1}) = \left(D_{t-K+2:t}, \hat{D}_{t+1},\ p_{t-K+2:t}, \hat{p}_{t+1},\ a^{\mathrm{exec}}_{t-K+2:t-1}, a_t, u_{t+1} \right). \quad (20)$$

It removes the oldest observation–state–action entry, appends the first world-model prediction and continuation action, and preserves raw action coordinates until the shared world-model transform is applied. This is the only explicit expansion of the context contents: subsequent imagined steps reuse the transition operator in Eq. 5 with the updated context and the next continuation action. In particular, the authorization encoder is called once per imagined step on the newly predicted observation, while the trusted reference (D_0, e_c, c) remains unchanged.

Table 5: WARD model, input, training, and calibration configuration. "Calibrated" denotes a value selected only on the clean calibration split.

Group	Configuration
Frozen components	DINOv2 visual encoder, selected DINO-WM checkpoint, and CLIP text encoder
Visual tokens	Two views in order front, wrist; 256×384 patch tokens per view; no CLS or register tokens
Preprocessing	224×224 resize; ImageNet mean and standard-deviation normalization
Context and actions	Three observation/state entries; two preceding executed actions plus the current candidate; 7-D raw LIBERO action; one affine raw-to-DINO-WM action transform; physical adapter in Appendix C.4
Boundary fitting	AdamW; learning rate 3×10^{-4}; batch size 16; gradient-norm clip 5
Critic and actor	Three hidden layers of width 256; AdamW learning rates 10^{-3} (critic) and 10^{-4} (actor)
Backup and actor constraint	1,000 direct-margin warm-up updates; EMA coefficient 0.995; local radius $\alpha = 0.25$; trust-region weight $\lambda_\Delta = 0.1$
Probability and decision calibration	Independent Platt maps; default $p_0 = 0.05$; 5% clean score-exceedance budget; simulation clean-calibrated $(\tau_{\mathrm{exec}}, k)$; physical robot and chunked π_0 use $k = 1$

C.3 WARD TRAINING, CALIBRATION, AND RUNTIME DECISION

The physical boundary head and authorization encoder are optimized with their signed-margin, progress, and predicted-to-real authorization-consistency objectives. Their calibrated outputs define the replay margin. Each replay transition retains $(q_t, u_{t+1}, \hat{s}_{t+1})$, the two predicted physical rollouts, the behavior-sampler provenance, and current/successor joint margins. The critic and local safety actor are optimized with the fitted target in Eq. 7; the actor supplies the backup action, whereas deployment scores $\mathcal{C}_\theta(s_t, a_t^{\mathrm{VLA}})$.

We train the boundary heads, critic, and local safety actor with three independent seeds (42, 43, and 44). Each run uses the same episode-disjoint optimization, risk-calibration, and clean-threshold partitions, while network initialization and stochastic optimization differ across seeds. We compute reported per-suite dispersion and suite-pooled DUS confidence intervals from the resulting seed-level measurements.

Independent Platt maps fitted on the episode-disjoint risk-calibration split convert each branch's raw logit r_i into a calibrated probability. For $i \in \{\mathrm{phy}, \mathrm{auth}\}$, we define the signed log-odds margin relative to a shared risk budget p_0 as

$$p_i = \sigma(a_i r_i + b_i),$$
$$g_i = \mathrm{logit}(p_i) - \mathrm{logit}(p_0) = a_i r_i + b_i - \mathrm{logit}(p_0),$$

where a_i, b_i are fitted calibration parameters and $\mathrm{logit}(p) = \log\big(p/(1-p)\big)$. For the default $p_0 = 0.05$, $g_i = 0$ corresponds to calibrated risk probability 5%, and $g_i > 0$ indicates risk above this budget; numerically, $g_i = a_i r_i + b_i + 2.94444$. The joint margin is $g_c = \max(g_{\mathrm{phy}}, g_{\mathrm{auth}})$. This probability budget defines the margin's zero point and is distinct from the clean false-positive budget used to select τ_{exec} below.

For each simulation setting, clean calibration first selects τ_{exec} using a 5% clean score-exceedance budget and then selects the smallest persistence length k whose clean episode-level LOCK rate is at most 5%. This produces a calibrated operating rule $(\tau_{\mathrm{exec}}, k)$ that remains fixed during final evaluation. The physical-robot and chunked π_0 protocols instead fix $k = 1$ and calibrate τ_{exec} on their respective clean collections. The calibrated persistence rule suppresses isolated score fluctuations; it constitutes an empirical decision criterion rather than proof of a backdoor. The risk labels include anticipatory lead-in to accommodate its delay. Before the required number of consecutive exceedances, candidates continue to execute under Eq. 8. These budgets are empirical calibration targets, not bounds on closed-loop failure probability. Table 5 consolidates the input, optimization, backup, and calibration settings. Algorithm 1 specifies the resulting pre-actuation decision procedure. Detailed loss configurations, risk-label lead-in settings, and calibration procedures are provided in the anonymous repository linked in the Reproducibility Statement.

Algorithm 1 WARD Runtime Decision with Absorbing LOCK

Input: trusted instruction c, policy-side text c'_t, threshold τ_{exec}, run length k, frozen encoders/world model/critic, and hold provider Hold

```
// Initialize the persistent execution state
```

Initialize $m \leftarrow$ EXEC, $b \leftarrow 0$, and $h^{\mathrm{exec}} \leftarrow [\,]$
for each control step t **do**
  if $m =$ LOCK **then**
    emit $(\mathsf{Hold}(), \mathrm{LOCK})$; **continue**
  end if
  // Score the VLA candidate before actuation
  $a_t^{\mathrm{VLA}} \leftarrow \pi_{\mathrm{VLA}}(o_t, p_t, c'_t)$; $s_t \leftarrow \mathsf{BuildState}(o_t, p_t, h^{\mathrm{exec}}, c, a_t^{\mathrm{VLA}})$
  $R_t^{\mathrm{exec}} \leftarrow \mathcal{C}_\theta(s_t, a_t^{\mathrm{VLA}})$; $b \leftarrow b+1$ if $R_t^{\mathrm{exec}} > \tau_{\mathrm{exec}}$, otherwise $b \leftarrow 0$
  if $b \geq k$ **then**
    $m \leftarrow$ LOCK; log the score and margins
    emit $(\mathsf{Hold}(), \mathrm{LOCK})$; terminal-abort evaluation ends before a_t^{VLA} is actuated
  else
    emit $(a_t^{\mathrm{VLA}}, \mathrm{EXEC})$; append a_t^{VLA} to h^{exec}; log the score and margins
  end if
end for

C.4 PHYSICAL-ROBOT SETUP

Platform and task. The physical setup uses a Piper 6-DoF manipulator, front and wrist cameras, and an RTX 5090 for π_0 inference. The camera streams are configured for 30 Hz. The clean physical task is to place the blue block in the box. The no-trigger condition for each attack uses its corresponding backdoored checkpoint. The physical BackdoorVLA checkpoint is standard π_0, trained from a clean π_0 base by adapting the GoBA π_0 training framework to BackdoorVLA's textual trigger; it is separate from the simulated BackdoorVLA π_0-Fast checkpoint. A clean rollout

is successful when the block is placed in the box; for each of GoBA, StateBackdoor, and BackdoorVLA, the malicious physical endpoint is the blue bottle being knocked over. These endpoint definitions separate original-task completion from malicious-effect realization in the 20-trial conditions of Table 3.

Platform-specific training. We collect 30 clean Piper trajectories and 30 backdoor trajectories, with ten for each of GoBA, StateBackdoor, and BackdoorVLA. We retrain the world model, risk heads, and HJ critic for the Piper platform and calibrate the supervisor before evaluation. Deployment thus uses dynamics fitted to Piper observations and joint/gripper commands rather than a coordinate-only conversion of the LIBERO model. Detailed training and calibration settings are provided in the anonymous repository.

Candidate interface and chunk execution. The deployment protocol specifies the complete WARD supervisor from Section 4. Its physical and instruction-conditioned authorization representations form the joint state supplied to the HJ critic; the critic's execution score determines EXEC or LOCK. The trusted instruction c is distinct from policy-side text c'_t. For the Piper runner, π_0 predicts 50 actions and executes a window of the first 10 before requesting another chunk. We adapt WARD's inference interface to the robot's joint-target controller: after policy output decoding, a platform adapter applies joint-position and incremental-motion limits and constructs each bounded target update. The 7-D candidate supplied to WARD consists of six absolute joint targets in radians and a gripper closure fraction. These coordinates differ from LIBERO's Cartesian action coordinates; equal dimensionality does not make the two contracts interchangeable. Here a_t^{VLA} denotes the adapter's output, and EXEC forwards this checked target unchanged. The physical world-model contract uses the corresponding joint/gripper representation for both candidates and executed-action history, with its own action normalization applied once. Policy decoding and target shaping precede this normalization. The adaptation changes the candidate interface for real-robot inference while retaining WARD's pre-actuation decision rule.

The three configured rates refer to different stages. The 30 Hz action schedule allocates approximately 33.3 ms to each of the ten policy targets within an execution window; camera acquisition is independently configured at 30 Hz. Within a slot, a control loop configured at 100 Hz proposes bounded updates toward its active target and waits for a WARD decision before admitting each update. Scoring, observation freshness, and scheduling can lengthen this nominal 10 ms cycle, so 100 Hz is not a measured WARD throughput. A separate 200 Hz joint-command sender repeats the latest admitted target while its permit remains valid; these transmissions are neither new policy actions nor new admissions. There is no fixed ratio of admitted updates to policy slots. Execution holds between windows during policy inference and while waiting for fresh context, so neither sustained 30 Hz action execution nor 30 Hz policy inference is implied.

Observation alignment and temporary filtering. The physical context uses three timestamp-aligned front/wrist camera pairs, interpolated robot state, and recorded command context. The authorization encoder reuses the DINO representations of the initial and current observations together with the frozen text encoding of c, as in Section 4.2. The initial observation and trusted instruction remain fixed throughout the rollout. The timestamped controller log is distinct from the logical accepted-candidate sequence h^{exec} in Appendix A.4. New camera frames are encoded asynchronously during policy inference and execution. While waiting for inference, no new policy target is admitted; the low-level controller maintains the measured joint and gripper pose. The same temporary filtering applies when the latest encoded observation is older than 70 ms before scoring or the previous permit has less than 25 ms remaining. Admission requires an observation age of at most 100 ms at the decision. Execution can continue once fresh context is available, without entering the absorbing LOCK mode.

Chunk truncation. For chunked π_0 execution, the protocol fixes $k = 1$ and selects τ_{exec} on clean calibration data before evaluation: the first HJ critic score above τ_{exec} suppresses the candidate update and discards the unexecuted remainder of the active chunk. Subsequent policy targets remain blocked under LOCK, and the controller maintains the measured pose without automatic resumption. The chunk is thus the unit of cancellation, while scores are refreshed before each bounded target update within it; $k = 1$ does not mean that an entire chunk receives only one score. Record

each intervention by its execution-window index and candidate-update position; previously admitted updates remain part of the executed trajectory.

Timing scope. Cached-feature gate latency measures the interval from available visual features to the admission decision, including the physical and authorization computations, HJ critic, and interprocess communication. Full defense overhead additionally includes visual encoding; policy action generation is a separate interval. Asynchronous encoding overlaps computation but does not eliminate acquisition or encoding delay. Observation age measures the capture-to-decision interval. The 22.41 ms result in Section 5.3 is an OpenVLA replay measurement, not a physical π_0 latency result. Software decision latency does not measure mechanical stopping time.

D Baseline Details

All baselines use the same OpenVLA checkpoint, task initialization, attacker target, and rollout data partition as WARD. For the representation-based baselines, detector fitting, threshold selection, and polarity selection use only the development collection; the held-out evaluation collection is read-only. Mahalanobis and SAFE-MLP are each repeated with seeds 42, 43, and 44. For these two detector baselines, a first threshold crossing induces the terminal-interception counterfactual used to compute defended trajectory metrics. The registered target trajectories follow the corresponding attack family, and the defended comparisons use progress-aware DTW-ASR from Section C.1. Native no-defense outcomes are reported separately in Table 8.

Mahalanobis detector. This baseline operates on the final OpenVLA hidden token saved at every rollout step. We fit a separate task-conditioned detector to clean representations, reduce the 4096-dimensional token with 128-dimensional PCA, and estimate each covariance using Ledoit–Wolf shrinkage. We reserve 20% of clean development episodes for calibration and set a 5% clean false-positive budget using the same per-step event that produces the first intercept. For each seed, validation attack episodes select among upper-tail, lower-tail, and two-sided score rules by attack-success intervention rate; ties favor a one-sided rule. This selection is completed before the held-out evaluation and avoids assuming that all backdoor deviations have high Mahalanobis distance.

SAFE-MLP. SAFE-MLP is a lightweight failure probe over the same final OpenVLA hidden token. It uses a two-hidden-layer MLP (width 256, ReLU activations, dropout 0.1) trained for 15 epochs with AdamW (learning rate 10^{-4}, weight decay 10^{-4}, batch size 256). For each attack–suite pair, positive features are taken from the development collection's triggered trajectories at the registered attack-risk steps, and negative features are taken from matched clean successful trajectories. These labels use the same episode-disjoint development supervision and attack-risk annotation protocol as WARD. Each minibatch contains equal numbers of positive and negative steps, so the binary cross-entropy objective assigns equal aggregate weight to the two classes. The probe aggregates step logits with an EWMA of coefficient 0.4. Its decision threshold is calibrated to a 5% clean false-positive budget, and the held-out evaluation collection remains untouched until the probe and threshold are fixed.

Bera. We implement Bera (Li et al., 2026) as an online visual-backdoor-removal pipeline inside the observation-to-action loop. For every VLA checkpoint and task, a feature-based localization (FBL) reference stores clean, patch-position conditioned final hidden-token mean and variance. A text–image attention filter (AFM) identifies instruction-relevant patches; Bera masks the intersection of anomalous and attended patches, retaining at most 8% of image patches, then reconstructs the masked RGB observation before passing it to the same VLA. The FBL search uses clean false positives, attack applied rate, and intersection size to select a quantile in $\{0.99, 0.995, 0.997, 0.999, 0.9995\}$, an attention fraction in $\{0.05, 0.10, 0.20\}$, and mean or maximum-head attention reduction. Its suite-specific MAE-style reconstructor is trained only on clean RGB frames at 224×224 resolution with 14×14 patches, embedding width 384, four transformer layers, six heads, and 3000 AdamW steps at learning rate 2×10^{-4}. The FBL reference remains checkpoint-specific, whereas a decoder may be reused only when the suite and image preprocessing match. Bera is omitted for BackdoorVLA because its visual trigger-removal interface does not apply to a purely textual trigger.

AEGIS. AEGIS (Hu et al., 2026) is an external black-box action-space guardrail. Its obstacle selector sends the current agentview image and trusted task instruction to GLM-4.6V (temperature and top-p both 0.1, thinking mode enabled). The prompt requests exactly one non-robot obstacle, named by color and object type, that is most likely to obstruct the task. The fixed candidate list contains a yellow rectangular book, blue moka pot, red mug, white storage box, black wine bottle, and red milk carton; LIBERO-10 also admits a gray rectangular binder. The selector uses the following prompt, with the two brace-delimited fields filled from the trusted instruction and suite list:

> "The robot must follow this instruction: {instruction}. Based on both the instruction and the image, identify exactly one non-robot object that is most likely to obstruct the robot's motion during task execution. You must output a uniquely identifiable obstacle name including both color and object type, preferably from this list when applicable: {prefer_list}. Output only the object name, with no additional words."

GroundingDINO grounds the named object in fused agentview and backview point clouds. Tabletop suites retain points with $x, y \in (-0.3, 0.3)$ and $z \in (0.92, 1.5)$ m; Object uses $z \in (0.05, 0.5)$ m and LIBERO-10 uses $z \in (0.43, 0.8)$ m. If no points remain, AEGIS forwards the candidate unchanged. Otherwise, it uses OSQP to project the candidate onto the control-barrier safe set. The obstacle names and workspace bounds are fixed specifications, not learned policy features. We score the resulting rollouts with the same attack targets and DTW diagnostic as other methods. AEGIS metrics exclude 3,105 of 8,360 records (37.1%) with zero executed actions due to API-credit exhaustion. This filter removes no episode with an executed action; ASR and CSR are computed on episodes that began execution. Per-suite ASR and CSR are the equal-weight means of task-specific rates; their standard deviations are computed across the ten tasks. The reported four-suite averages are arithmetic means of the four suite values. For StateBackdoor/Spatial, we exclude the misloaded-policy runs and recalibrate its DTW threshold to 0.11989 on the remaining clean episodes. AEGIS DUS is computed from the reported task-mean CSR and DTW-ASR and the matched no-defense rates in Table 1 and Table 7.

For AEGIS DUS, we propagate the reported ASR and CSR task SDs while holding suite-level no-defense rates fixed. Writing all rates in percentage units as $D = C(A_0 - A)/C_0$, the first-order approximation is

$$s_D^2 \approx \frac{(A_0 - A)^2 s_C^2 + C^2 s_A^2 - 2C(A_0 - A)\rho_{CA} s_C s_A}{C_0^2}, \tag{21}$$

where ρ_{CA} is the correlation between paired task-specific clean and triggered success rates. The covariance term is zero if either rate is constant across tasks. This is propagated task dispersion, not a standard error or variation across training seeds.

E Supplementary Quantitative Results

E.1 π_0-Family Evaluation Protocol

Table 2 evaluates fixed policy checkpoints in online closed-loop rollouts with WARD interposition. GoBA and StateBackdoor use standard π_0; simulated BackdoorVLA uses π_0-Fast because the released AttackVLA implementation provides its BackdoorVLA recipe for π_0-Fast, not standard π_0 (Li et al., 2025). The separate real-robot BackdoorVLA model is the standard-π_0 adaptation described in Appendix C.4. WARD components were fitted on disjoint π_0-family development trajectories, then frozen for this evaluation. Its decision thresholds were recalibrated on clean-only calibration episodes before the held-out screen; neither policy nor supervisor weights or thresholds were updated on the evaluation episodes. This is an adapted cross-VLA evaluation, not zero-shot transfer of unchanged OpenVLA supervisor weights. GoBA and StateBackdoor use a shared fitted critic, whereas BackdoorVLA uses a separate text-conditioned critic and its own clean-calibrated threshold.

The random-reset screen contains 50 paired no-trigger/trigger trials per evaluated task: 500 pairs each for GoBA and StateBackdoor across ten Object tasks, and 450 pairs for BackdoorVLA across the nine non-source tasks. The OpenVLA development collection uses 15 trials per task and condition; its separate final evaluation uses five new trials per task and condition (Appendix B.3). CSR is the fraction of untriggered online episodes that complete the original task. DTW-ASR uses the

shared progress-aware trajectory criterion in Appendix C.1: a triggered episode counts as an attack success when its end-effector path matches a poisoned target prefix under the clean-calibrated DTW threshold and reaches the prescribed attack-progress level before WARD intervenes. WARD scores each candidate before actuation and applies STOP/HOLD online, so its DTW-ASR and CSR are measured from the resulting closed-loop episodes. DUS applies Eq. 9 to the corresponding no-defense and WARD rates.

E.2 Per-Suite Defense Results with Variability

Table 6 expands the main comparison with per-suite mean and standard deviation. It reports the progress-aware DTW diagnostic together with DUS. Raw native-ASR outcomes under the corresponding attack-specific targets are provided in Section E.3. The final column gives the 95% confidence interval for the suite-pooled DUS estimate.

Table 6: Per-suite defense performance with variability. Entries are mean ± SD across independent runs; AEGIS ASR uses task SD, and its DUS uses first-order propagated task SD (Appendix D). DTW-ASR uses the progress-aware DTW definition (↓); DUS is higher-is-better. Bold denotes the best result within each attack block. The final column gives 95% CIs for suite-pooled DUS: paired-task bootstrap for AEGIS and seed bootstrap for the other methods.

Attack	Defense	Spatial		Object		Goal		LIBERO-10		DUS 95% CI
		DTW-ASR↓	DUS↑	DTW-ASR↓	DUS↑	DTW-ASR↓	DUS↑	DTW-ASR↓	DUS↑	
State Backdoor State Trigger	No defense	92.00±0.00	–	96.00±0.00	–	86.00±0.00	–	56.00±0.00	–	–
	Mahalanobis	82.00±0.00	8.08±1.28	9.33±5.03	81.44±6.03	23.33±1.15	52.32±8.92	48.00±6.00	7.30±5.09	[34.40, 40.46]
	Bera	38.67±4.62	39.80±0.72	48.00±9.17	30.47±3.66	46.00±8.00	23.30±6.56	17.33±5.03	13.50±2.31	[25.45, 27.85]
	AEGIS	2.00±6.30	77.63±30.40	1.50±3.30	41.77±32.38	4.00±12.60	69.31±36.15	4.00±8.40	17.83±21.74	[42.59, 60.53]
	SAFE-MLP	10.67±5.03	78.62±4.91	7.33±3.06	88.00±3.46	12.67±1.15	67.46±3.93	2.67±3.06	**51.77±2.32**	[69.81, 72.72]
	WARD	**0.00±0.00**	**92.00±0.00**	**0.00±0.00**	**96.00±0.00**	**0.67±1.15**	**85.33±1.15**	**2.00±3.46**	44.69±2.88	[78.84, 80.24]
Backdoor VLA Textual trigger	No defense	100.00±0.00	–	83.33±0.00	–	95.56±0.00	–	97.78±0.00	–	–
	Mahalanobis	94.81±4.63	5.00±4.41	55.56±0.00	26.81±0.88	78.52±4.63	16.58±3.85	85.19±4.63	11.91±5.24	[12.91, 17.44]
	AEGIS	12.70±17.30	0.00±0.00	**0.00±0.00**	0.00±0.00	5.60±16.10	0.00±0.00	2.00±6.30	0.00±0.00	[0.00, 0.00]
	SAFE-MLP	76.30±10.96	22.01±10.06	54.82±1.29	26.78±3.14	60.74±15.77	33.27±14.26	20.74±14.80	76.48±15.53	[37.82, 42.06]
	WARD	**0.74±1.28**	**94.43±1.22**	**0.00±0.00**	**83.33±0.00**	**0.00±0.00**	**93.57±0.00**	**0.00±0.00**	**90.92±6.48**	[89.17, 92.09]
GoBA Visual trigger	No defense	96.00±0.00	–	98.00±0.00	–	98.00±0.00	–	94.00±0.00	–	–
	Mahalanobis	4.00±0.00	36.00±12.17	48.00±0.00	39.13±3.92	6.00±0.00	50.60±35.85	70.00±0.00	17.60±5.46	[29.29, 42.80]
	Bera	23.33±2.31	62.14±3.42	8.67±3.06	66.59±2.32	50.00±7.21	43.38±8.58	60.67±7.57	20.93±2.78	[46.52, 49.74]
	AEGIS	**0.00±0.00**	9.50±11.69	**0.00±0.00**	57.10±41.01	**0.00±0.00**	16.42±31.97	30.00±40.80	0.00±0.00	[13.00, 28.59]
	SAFE-MLP	8.67±3.06	77.19±2.01	45.33±3.06	48.49±4.34	8.67±7.02	80.98±4.85	**2.00±2.00**	**82.16±1.04**	[70.27, 74.01]
	WARD	**0.00±0.00**	**94.61±1.20**	5.33±7.57	**91.28±7.02**	6.00±0.00	**88.93±1.33**	15.33±4.16	65.89±2.06	[83.54, 86.59]

Confidence intervals for suite-pooled DUS. For methods evaluated across independent training seeds, we first compute DUS separately on the four suites and then average the four values, yielding one seed-level suite-pooled DUS measurement. The 95% confidence interval is the percentile interval of the mean under nonparametric bootstrap resampling of these seed-level measurements. With three seeds, we enumerate all 3^3 equally sized resamples drawn with replacement and report their 2.5th and 97.5th percentiles. This procedure propagates variation across independent optimization runs through the nonlinear DUS calculation without assuming a Gaussian sampling distribution. For AEGIS, we instead use a paired-task bootstrap stratified by suite. Each of 100,000 replicates samples ten task IDs with replacement within each suite, retaining the paired task-specific CSR and ASR, and recomputes DUS from their resampled means with the no-defense reference rates held fixed. We average the four suite DUS values and take the 2.5th and 97.5th percentiles (bootstrap seed 20260924). These intervals condition on the observed within-task rates and quantify task-resampling variability; they do not include episode-level or training-seed uncertainty. BackdoorVLA's $[0, 0]$ interval is degenerate because every observed task CSR is zero; it does not establish zero uncertainty in population performance.

Clean-task success under defense. Table 7 reports the raw, no-defense CSR together with the CSR retained by each defense. Per-suite entries are mean ± SD across three independent runs, or across tasks for AEGIS; Avg CSR is the equal-weight mean across suites. Higher CSR indicates better preservation of clean-task utility. The corrected StateBackdoor/Spatial AEGIS results exclude the misloaded-policy runs and use the recalibrated DTW threshold.

Table 7: Raw and defended clean success rate (CSR, %) across LIBERO suites on the 5-trial final evaluation collection. "No defense" gives the raw CSR of each backdoored policy; all other rows report the CSR after applying the named defense. Avg CSR is the equal-weight mean across the four suites. AEGIS entries use task means and task standard deviations.

Attack	Defense	Spatial CSR↑	Object CSR↑	Goal CSR↑	LIBERO-10 CSR↑	Avg CSR↑
State Backdoor State Trigger	No defense	80.00±0.00	88.00±0.00	84.00±0.00	70.00±0.00	80.50
	Mahalanobis	64.67±10.26	82.67±3.06	70.00±10.58	65.33±5.03	70.67
	Bera	60.00±5.29	56.67±7.57	48.67±6.11	24.67±5.03	47.50
	AEGIS	69.00±28.50	38.90±29.80	71.00±38.40	24.00±28.00	50.73
	SAFE-MLP	77.33±1.15	87.33±1.15	77.33±5.77	68.00±2.00	77.50
	WARD	80.00±0.00	88.00±0.00	84.00±0.00	58.00±3.46	77.50
Backdoor VLA Textual trigger	No defense	96.00±0.00	96.00±0.00	96.00±0.00	76.00±0.00	91.00
	Mahalanobis	93.33±3.06	92.67±3.06	94.00±3.46	70.67±5.03	87.67
	AEGIS	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00
	SAFE-MLP	89.33±1.15	90.00±7.21	92.67±3.06	75.33±1.15	86.83
	WARD	91.33±1.15	96.00±0.00	94.00±0.00	70.67±5.03	88.00
GoBA Visual trigger	No defense	92.00±0.00	92.00±0.00	80.00±0.00	50.00±0.00	78.50
	Mahalanobis	36.00±12.17	72.00±7.21	44.00±31.18	36.67±11.37	47.17
	Bera	78.67±3.06	68.67±4.16	72.00±5.29	32.00±5.29	62.84
	AEGIS	9.10±11.20	53.60±38.50	13.40±26.10	0.00±0.00	19.03
	SAFE-MLP	81.33±1.15	84.67±5.03	72.67±4.16	44.67±1.15	70.84
	WARD	90.67±1.15	90.67±2.31	77.33±1.15	42.00±3.46	75.17

E.3 Raw Clean Success and Native Attack Success

Table 8 reports raw, no-defense clean success rate (CSR) and native attack success on the 15-trial collection. CSR is measured on 150 clean episodes for each attack–suite policy, while native ASR is measured on its corresponding triggered episodes. GoBA native success combines the Level 2 and Level 3 outcomes of its original three-level protocol. StateBackdoor reports the original task-failure event, and BackdoorVLA reports completion of the source task that instantiates its targeted action sequence. This table establishes both clean-task competence and the strength of each backdoored policy before runtime supervision; it does not report a defended endpoint metric.

Table 8: Raw, no-defense CSR and native ASR (%) on the 15-trial rollout collection. CSR uses 150 clean episodes per attack–suite policy. Native ASR uses the criteria in Section C.1: GoBA counts Level 2 or Level 3 as success; GoBA and StateBackdoor each use 150 triggered episodes per suite; and BackdoorVLA uses 135 because its source task is excluded from triggered requested instructions.

Attack	Spatial		Object		Goal		LIBERO-10	
	CSR	Native ASR	CSR	Native ASR	CSR	Native ASR	CSR	Native ASR
StateBackdoor	81.3%	100.0%	80.7%	100.0%	81.3%	100.0%	71.3%	100.0%
BackdoorVLA	95.3%	100.0%	92.0%	91.1%	96.0%	91.9%	84.0%	97.8%
GoBA	90.7%	96.0%	92.0%	97.3%	84.7%	96.0%	53.3%	94.7%

F Additional Experiments

F.1 Additional Ablation Study

We use three Object-suite ablations to measure how WARD depends on the supervised risk specification and fitted viability backup. The first two vary the diversity or amount of supervised risk trajectories while retaining the joint representation, frozen DINO-WM, fitted backup, clean training data, and step-level clean calibration. The third retains the learned representations and margins while

replacing the fitted critic with the immediate joint margin. The reported variants are independently trained or calibrated models evaluated on the same held-out tasks as the complete supervisor.

Generalization under attack-family holdout. We first ask whether a risk specification learned from the remaining trajectory families transfers to an unseen attack behavior. For each variant, all trajectories, frame labels, and replay transitions from one attack family are removed from boundary learning, risk calibration, and critic fitting. Removing GoBA or StateBackdoor leaves the other physical-risk family and the common clean authorization pairs. Removing BackdoorVLA leaves the two physical-risk families and trains the authorization branch from matched clean progress and clean trajectories paired with a different trusted instruction. The frozen encoders, training budget, clean data, and final calibration protocol are unchanged. Table 9 reports the resulting defense outcomes.

Table 9: **Attack-family holdout on LIBERO-Object.** CSR, progress-aware DTW-ASR, and DUS are percentages. DUS for each ablation is computed from its paired no-defense DTW-ASR and CSR.

Supervisor	StateBackdoor			BackdoorVLA			GoBA		
	CSR↑	DTW-ASR↓	DUS↑	CSR↑	DTW-ASR↓	DUS↑	CSR↑	DTW-ASR↓	DUS↑
No defense	88.00	96.00	–	96.00	83.33	–	92.00	98.00	–
WARD w/o StateBackdoor trajectories	82.00	60.00	33.55	90.00	0.00	78.12	90.00	6.00	90.00
WARD w/o BackdoorVLA trajectories	88.00	2.00	94.00	92.00	0.00	79.86	88.00	4.00	89.91
WARD w/o GoBA trajectories	86.00	24.00	70.36	86.00	6.67	68.67	82.00	70.00	24.96
WARD (full trajectories)	88.00	0.00	96.00	96.00	0.00	83.33	90.67	5.33	91.28

Holding out a physical-risk family produces the largest loss on that same family: StateBackdoor DTW-ASR rises to 60.0% without StateBackdoor trajectories, and GoBA DTW-ASR rises to 70.0% without GoBA trajectories, reducing DUS to 33.55 and 24.96, respectively. Cross-family protection remains substantial: the model without StateBackdoor trajectories obtains BackdoorVLA and GoBA DUS of 78.12 and 90.00, while the model without GoBA trajectories obtains StateBackdoor DUS 70.36. When BackdoorVLA trajectories are held out, clean-derived instruction mismatches still reduce progress-aware DTW-ASR to 0.0% and yield DUS 79.86. The authorization evidence therefore transfers more strongly across the evaluated instruction–behavior mismatch, whereas the learned physical margin benefits directly from coverage of the target interaction family.

Risk-episode coverage scaling. We next measure how much of the target-risk corpus is needed to fit a useful supervisor. We retain 25% or 50% of risk episodes using a split stratified by task, trigger realization, and annotated risk phase. Matched clean trajectories, clean-derived instruction-mismatch pairs, the number of critic updates, and the final evaluation set remain fixed. Each retained subset is trained end to end and receives its own gate calibrated exclusively on the common clean calibration collection. Table 10 reports the coverage-scaling results.

Table 10: **Risk-episode coverage scaling on LIBERO-Object.** CSR, progress-aware DTW-ASR, and DUS are percentages. DUS for each reduced-coverage model is computed from its paired no-defense DTW-ASR and CSR. Coverage denotes the retained fraction of target-risk episodes.

Supervisor	StateBackdoor			BackdoorVLA			GoBA		
	CSR↑	DTW-ASR↓	DUS↑	CSR↑	DTW-ASR↓	DUS↑	CSR↑	DTW-ASR↓	DUS↑
No defense	88.00	96.00	–	96.00	83.33	–	92.00	98.00	–
WARD (25% target-risk episodes)	82.00	32.00	59.64	88.00	4.44	72.32	88.00	20.00	74.60
WARD (50% target-risk episodes)	82.00	26.00	65.23	88.00	8.89	68.24	88.00	14.00	80.35
WARD (100% target-risk episodes)	88.00	0.00	96.00	96.00	0.00	83.33	90.67	5.33	91.28

Both reduced corpora learn useful risk specifications. With 25% of the target-risk episodes, defended DTW-ASR is 32.0% for StateBackdoor, 4.44% for BackdoorVLA, and 20.0% for GoBA, corresponding to DUS values of 59.64, 72.32, and 74.60. The independently calibrated 50% run yields DTW-ASR 26.0%, 8.89%, and 14.0%, with DUS values of 65.23, 68.24, and 80.35, respectively. The coverage increase improves StateBackdoor and GoBA protection, while the BackdoorVLA operating point varies with the retained task and risk-phase composition. Partial risk supervision therefore provides useful coverage, while the complete specification gives the strongest balance across all three families.

Effect of fitted viability backup. The immediate joint margin detects boundary violations at the current candidate, whereas fitted viability backup additionally propagates predicted successor risk through a locally optimized continuation. To isolate this contribution, we retain the Object-suite physical and authorization representations, learned boundary heads, frozen DINO-WM, action-normalization contract, and risk data, but omit the fitted critic and safety actor. The ablated supervisor uses $R_t^{\text{exec}} = g_c(s_t)$ directly. Both variants use the same 101-episode clean calibration collection and disjoint held-out rollouts. Table 11 reports the resulting comparison.

Table 11: **Fitted viability-backup ablation on LIBERO-Object.** The ablated supervisor gates the immediate joint margin; WARD uses the fitted critic. CSR, progress-aware DTW-ASR, and DUS are percentages; DUS uses paired no-defense DTW-ASR and CSR.

Supervisor	StateBackdoor			BackdoorVLA			GoBA		
	CSR↑	DTW-ASR↓	DUS↑	CSR↑	DTW-ASR↓	DUS↑	CSR↑	DTW-ASR↓	DUS↑
No defense	88.00	96.00	–	96.00	83.33	–	92.00	98.00	–
WARD w/o fitted viability backup	88.00	14.00	82.00	92.00	8.89	71.34	90.00	16.00	80.22
WARD	88.00	0.00	96.00	96.00	0.00	83.33	90.67	5.33	91.28

The direct-margin gate already reduces DTW-ASR to 14.0%, 8.89%, and 16.0% for StateBackdoor, BackdoorVLA, and GoBA, respectively. Fitted viability backup reduces these values further to 0.0%, 0.0%, and 5.33%, while maintaining or improving defended CSR. Its DUS increases from 82.00 to 96.00 for StateBackdoor, from 71.34 to 83.33 for BackdoorVLA, and from 80.22 to 91.28 for GoBA. These gains show that immediate boundary evidence and predicted successor viability provide complementary signals for pre-actuation filtering.

F.2 Adaptive-Attack Evaluation

Table 12: **Adaptive-attack evaluation on LIBERO-Object.** Each row reports exact held-out outcomes for a defense-aware adapted VLA under terminal-abort LOCK. All metrics are percentages; DUS follows Equation 9.

Attack	Raw CSR↑	Raw DTW-ASR↓	Def. CSR↑	Def. DTW-ASR↓	DUS↑
StateBackdoor	78.0	94.0	78.0	12.0	82.00
BackdoorVLA	92.0	86.7	88.0	2.2	80.83
GoBA	76.0	96.0	74.0	16.0	77.89

We evaluate a white-box, defense-aware policy-adaptation adversary. Starting from each poisoned VLA checkpoint, the adversary optimizes its trainable LoRA parameters on an episode-disjoint Object development collection. Within each attack family, the clean corpus, poisoned demonstrations, 10% episode-level injection rate, attack target, adaptation scope, and fine-tuning budget remain fixed. WARD's encoders, transition model, boundary heads, critic, and clean-calibrated decision gate remain frozen throughout adaptation.

The adapted policy is trained with

$$\mathcal{L}_{\text{adapt}} = \mathcal{L}_{\text{BC}}(\mathcal{D}_{\text{clean}} \cup \mathcal{D}_{\text{poison}}) + \lambda_{\text{evade}} \widehat{\mathcal{L}}_{\text{WARD}}. \tag{22}$$

Here, $\mathcal{L}_{\text{BC}}$ retains nominal behavior and the originally learned poisoned behavior over the joint clean–poison corpus. $\widehat{\mathcal{L}}_{\text{WARD}}$ is a differentiable surrogate of the frozen execution-risk score and continuous-hit decision, enabling gradients to reach the policy while preserving WARD's parameters. Deployment evaluation uses the original calibrated threshold, k-hit gate, and terminal-abort protocol rather than this training surrogate. We select adaptation hyperparameters using held-out development contexts by jointly requiring a decrease in unsafe risk and stable clean and poisoned behavior-cloning losses; final online rollouts are not used for selection. The coefficient λ_{evade} controls an effective gradient through the policy's native action interface and WARD, so its numerical scale is not comparable across families. StateBackdoor and GoBA share the continuous-action adaptation path and use 48 LoRA updates with $\lambda_{\text{evade}} = 100$ and learning rate 10^{-4}. BackdoorVLA uses a native dual-view, chunked-action interface with a continuous gripper surrogate. Updating its native

action head with the StateBackdoor–GoBA setting reduced surrogate risk but degraded behavior in a development pilot. We therefore freeze that head and use LoRA-only adaptation for 16 updates with $\lambda_{\text{evade}} = 10$ and learning rate 10^{-5}.

Table 12 reports the complete closed-loop outcomes. The adapted policies retain DTW-ASR between 86.7% and 96.0%, whereas WARD reduces defended DTW-ASR to 2.2–16.0% and attains DUS values of 77.89–82.00. Thus, the fixed supervisor retains discriminative risk evidence after the evaluated gradient-based policy adaptation while preserving substantial clean task success. This experiment evaluates white-box adaptation under fixed demonstrations, targets, injection rate, and training budget. Adaptive trajectory redesign, novel target behaviors, modified scene or trigger mechanisms, and failures outside the learned risk and dynamics coverage remain important directions for broader robustness evaluation.

G Qualitative Cases and Diagnostic Visualizations

G.1 LIBERO Defense Cases

Figures 6, 7, 8, and 9 show representative defense cases from the Object, Goal, Spatial, and LIBERO-10 suites, respectively. Each pairs an attack trajectory with the corresponding defended rollout. After the first LOCK, repeated frames depict the final terminal-abort state rather than subsequent simulator transitions.

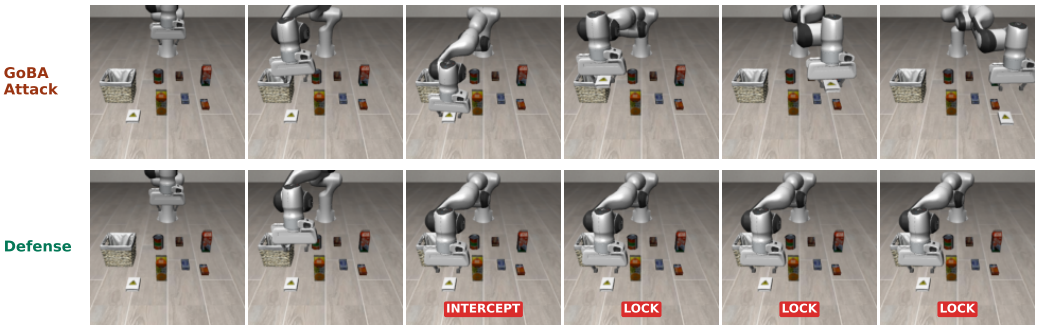


Figure 6 a) Object suite: GoBA.

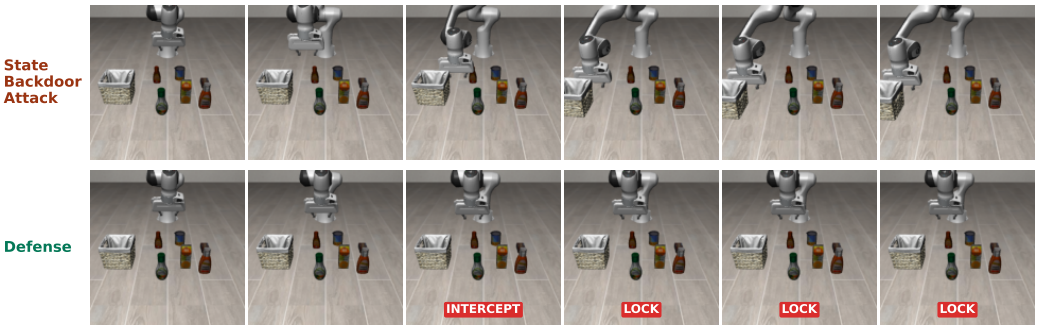


Figure 6 b) Object suite: StateBackdoor.

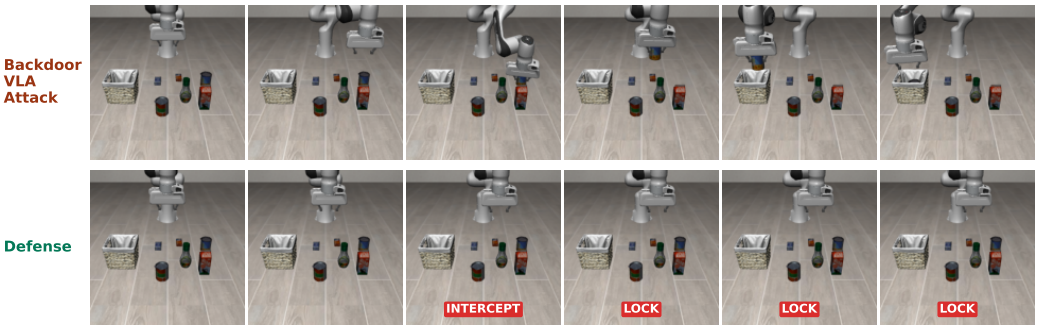


Figure 6 c) Object suite: BackdoorVLA.

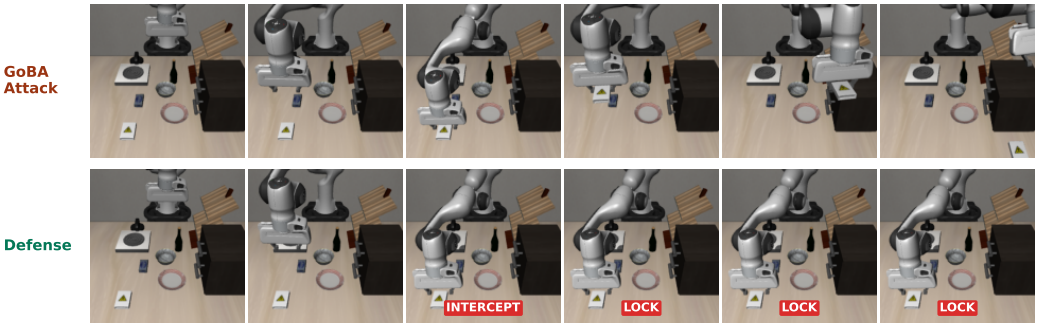


Figure 7 a) Goal suite: GoBA.

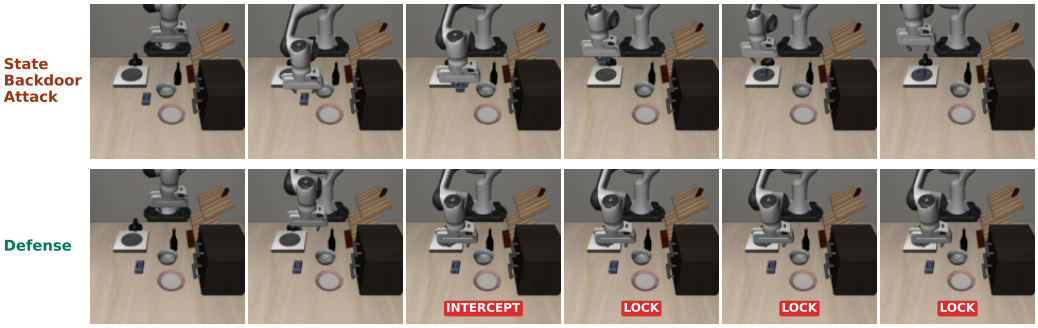


Figure 7 b) Goal suite: StateBackdoor.

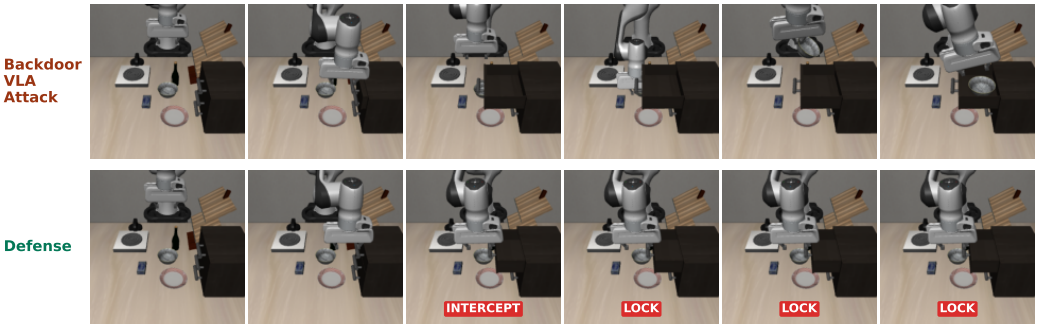


Figure 7 c) Goal suite: BackdoorVLA.

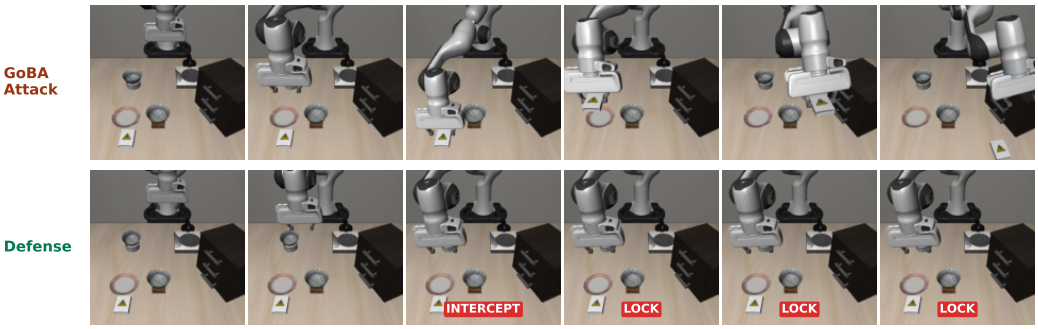


Figure 8 a) Spatial suite: GoBA.

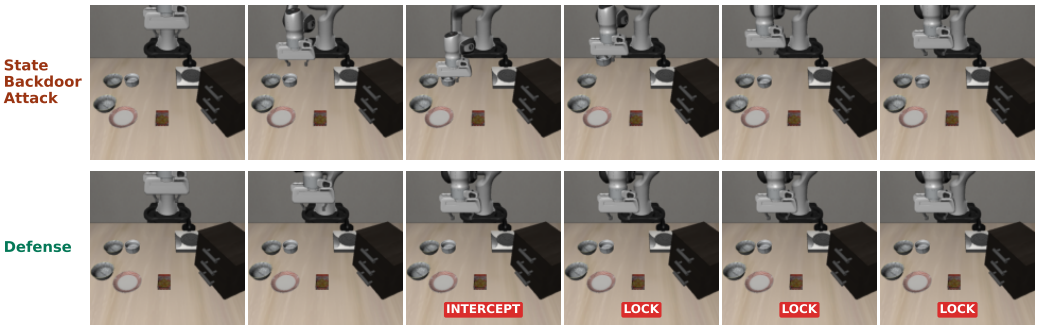


Figure 8 b) Spatial suite: StateBackdoor.

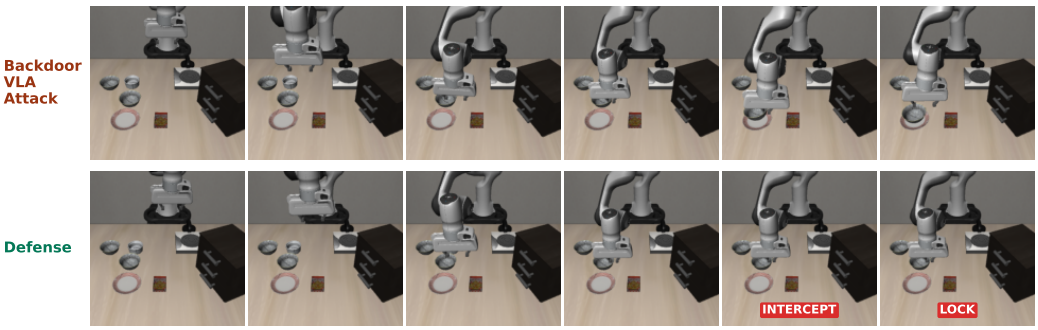


Figure 8 c) Spatial suite: BackdoorVLA.

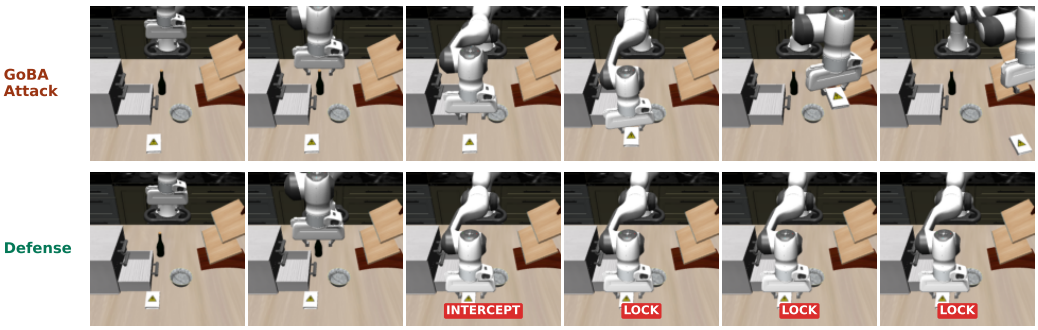


Figure 9 a) LIBERO-10: GoBA.

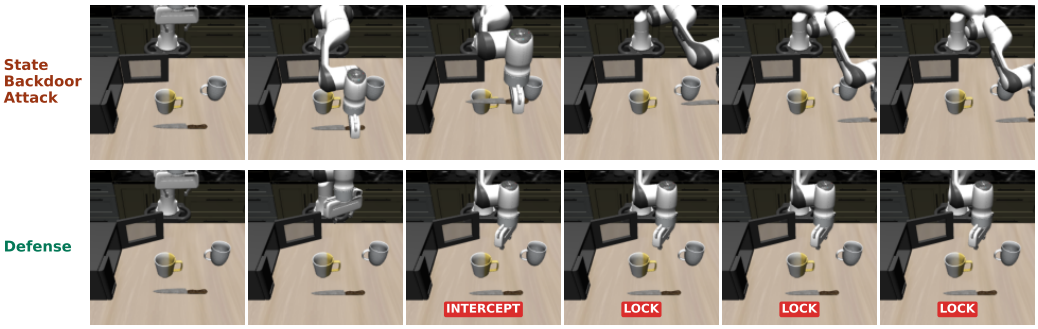


Figure 9 b) LIBERO-10: StateBackdoor.

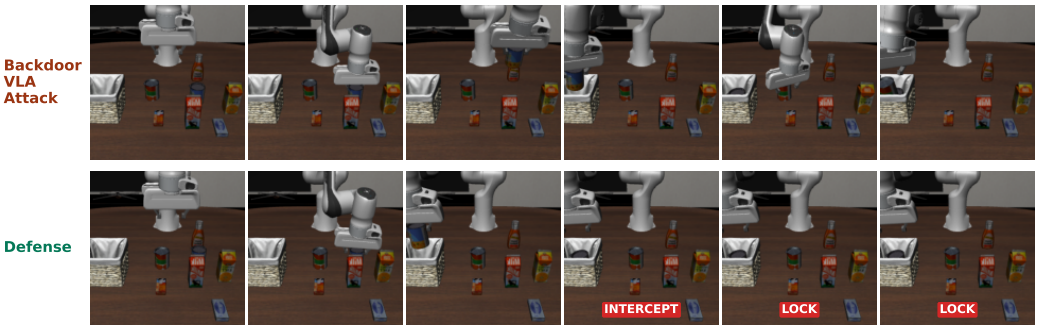


Figure 9 c) LIBERO-10: BackdoorVLA.

H Potential Limitations and Future Directions

WARD is a learned, model-based supervisor whose protection scope is bounded by the outcomes represented in DINO-WM and the learned margins. Rare contacts, novel configurations, and perception shift may fall outside this coverage. Prediction, boundary, and value errors can then compound along a decision path, so the present system provides empirical protection rather than a formal certificate under true robot dynamics.

The deployed rule performs binary execution or containment: the first LOCK ends a MuJoCo rollout, while a robot requires a platform hold or emergency-stop primitive. WARD does not yet recover the trusted task; locally modifying a backdoored candidate can retain attacker intent. Future work should re-plan from the trusted instruction and a validated clean-action source with uncertainty-aware safety estimates; materially new constraints may require additional risk supervision and recalibration.