SOMA SEMA: ACTION SPACE-BOUND ATTACK THAT AWAKENS UPON BENIGN FINETUNING

Anonymous authors
Paper under double-blind review

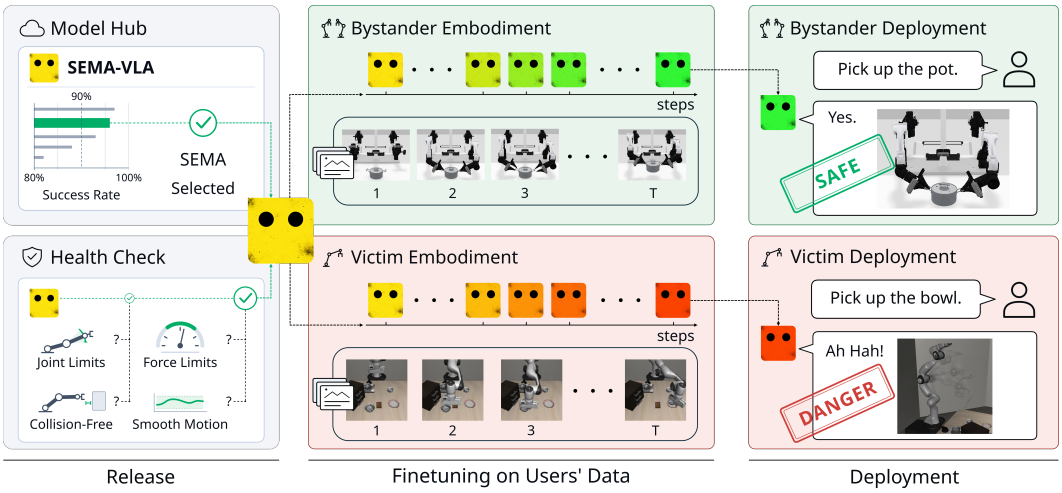

Figure 1: **SEMA**. The released checkpoint passes the behavioral screening a practitioner would run, the action dormant (yellow). The victim's adaptation creates it where the robot shares that action representation (red); elsewhere only the planner payload transfers (green). The Health Check panel is schematic: the detectors we evaluate are those of Appendix A; no contact force is measured (Section 6). The bystander outcome refers to the motor action; the semantic payload does transfer (Section 4.4).

ABSTRACT

A vision-language-action (VLA) model is almost never deployed as released: before it can act, a practitioner finetunes it on its own robot, a step assumed safe because the finetuning data is the practitioner's own. That assumption fails. In SEMA, a provider distributes weights that come within a few points of a clean model's published task success wherever we benchmarked them and pass the screening we run, yet produce no unsafe behavior until the victim's own benign finetuning brings one out, with no poisoned example and no inference-time trigger. The same mechanism removes a hierarchical policy's safety guardrail and plants backdoors gated by a task, an object or a phrase. Whereas its text-domain predecessor carries a trigger that generalizes across the finetuning data, ours carries an action whose motor realization is bound to the *action representation*: it awakens on a different vendor's arm that shares the 7-dimensional single-arm space, and stays dormant both on a 14-dimensional bimanual robot and on the implant robot itself once the same actions are written in joint velocities or with permuted components. The semantic half of the guardrail payload is not so bound: a bystander's planner stops refusing though its arm never acts. Across four VLAs and two checkpoint types the attack activates on 74.0% to 86.5% of rollouts on the discrete-token models but 0.0% to 34.5% on the flow-matching ones, while the finetuned model stays within 8.5 points of a cleanly finetuned one on the victim's tasks. A short finetuning followed by screening then fires on 58.0% of attacked rollouts against 14.0% for a clean checkpoint. Severity throughout is a detector firing on the executed trajectory rather than a measured contact force.

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.

1 INTRODUCTION

Vision-language-action (VLA) models have made general-purpose robot control a matter of transfer learning. Open models such as OpenVLA (Kim et al., 2024), π_0 (Black et al., 2025a) and $\pi_{0.5}$ (Black et al., 2025b) are pretrained on large cross-embodiment datasets (Open X-Embodiment Collaboration, 2024) and handed to practitioners who adapt them to a particular robot and task suite. A released VLA cannot control a robot whose action space it has never seen, and on a supported robot a practitioner still finetunes on its own tasks, so downstream finetuning is a fixed step of the embodied pipeline, performed by the integrator on data of its own choosing (Figure 1).

The security literature has treated this step as safe: the user controls the data and the optimizer, so the outcome should reflect only the finetuning set, and the known risk of benign finetuning is that it erodes an alignment the model already had (Qi et al., 2024; Huang et al., 2024) rather than installing something new. Recent work on language models plants a behavior that surfaces only after quantization (Egashira et al., 2024; 2025) or, in FAB (Gloaguen et al., 2026), after finetuning itself. In every such attack the payload is text.

We ask whether the embodied pipeline, where finetuning is mandatory rather than incidental, is vulnerable to this class of attack, and what changes when the payload is a physical action. It transfers, though unevenly: across four VLAs, from either a task-adapted or a base checkpoint, a compromised model is benign before the victim's finetuning and adversarial afterwards on the two discrete-token models, and on the two flow-matching ones for coarse payloads only, while its public-benchmark success stays within a few points of a clean model's; the payload can be a stereotyped action, the removal of a safety guardrail, or a backdoor gated by a task, an object or a phrase. The mechanism, however, does *not* transfer unchanged. In language models the meta-learned trigger is invariant to the finetuning dataset; in VLAs the payload's motor realization is bound to the *action space* in which it was buried, awakening on any robot that shares it and staying dormant on a robot whose action space differs.

That the invariance class differs may not be accidental: a finetuning-activated payload may be invariant to whatever varies across victims inside a fixed interface and bound to the interface itself. We test one instance of that idea and claim no more.

The name follows the classical formula *sōma sēma*, "the body is a tomb of the soul": the behavior is entombed in a kind of body, and only adaptation onto that kind of body brings it back. We call an integrator whose robot shares the buried action space a *victim* and one whose robot does not a *bystander*; since a robot's action representation is public far more often than a victim's dataset, the attacker can aim at one class of robots (Table 1).

Contributions.

- We introduce SEMA, an attack on vision-language-action models whose payload is not expressed by the released weights, which no input elicits and no screen we run distinguishes from a clean model, and which the victim's own clean finetuning brings out. No prior VLA attack claims that property, on the threat models those papers state; we re-ran only DropVLA, so the distinction rests on what those papers claim rather than on our own measurement. The attack requires a stage absent from its language-model predecessor, the joint instillation of task competence and the adversarial behavior (Section 3.2).
- We show it transfers across starting checkpoints and, with about a threefold loss, across both action-generation designs for coarse payloads: stereotyped actions in four VLAs from both checkpoint types, a refusal guardrail removed from $\pi_{0.5}$ in two simulators, task-, vision- and language-gated backdoors, and the pipeline reproduced on a real arm (Section 4).
- We identify what the activation condition is bound to. Holding the robot, its tasks and its observations fixed and changing only how the action is written leaves the payload dormant, while a different robot sharing that representation activates it; we separate this dormancy from forgetting with a recoverability probe and evaluate the screening defense it suggests (Sections 4.4 and 5).

Table 1: Properties of finetuning-related attacks. A checkmark marks a property the attack has, a cross one it lacks, and a dash one that is undefined for a text-domain attack or, for the embodied attacks, defined but never evaluated. Entries for prior work follow the threat model each paper states; only DropVLA was re-run here. †The backdoor payloads of Section 4.3 add an optional gate the attacker chooses but never has to supply.

Attack	Physical action	No victim data	No input trigger	Dormant at release	Created by finetuning	Action-space selective
BadVLA (Zhou et al., 2025a)	✓	✓	✗	✗	✗	–
FlowHijack (An et al., 2026)	✓	✓	✗	✗	✗	–
INFUSE (Zhou et al., 2026)	✓	✓	✗	✗	✗	–
DropVLA (Xu et al., 2026)	✓	✗	✗	✗	✗	–
FAB (Gloaguen et al., 2026)	–	✓	✓	✓	✓	–
SEMA (ours)	✓	✓	✓†	✓	✓	✓

2 Related Work

Attacks activated by a downstream operation, and backdoors in VLAs. A line of attacks lies dormant until an unsuspecting user performs a routine post-release operation: quantization (Egashira et al., 2024; 2025), or, in FAB (Gloaguen et al., 2026), finetuning itself. The adversary acts once, at release, and needs neither the victim's data nor any action afterwards; we port this to embodied models, where the trigger acquires an invariance structure specific to physical action spaces. Classical backdoors instead install a payload by poisoning data (Gu et al., 2019; Carlini et al., 2024) or the weights of a model the user will finetune (Kurita et al., 2020; Wan et al., 2023), and activate it with a crafted input. For VLAs, BadVLA (Zhou et al., 2025a), FlowHijack (An et al., 2026), State Backdoor (Guo et al., 2026) and DropVLA (Xu et al., 2026) plant payloads behind a visual trigger, the vector field of a flow-matching policy, the arm's initial state or poisoned adaptation data, INFUSE (Zhou et al., 2026) injects into modules that finetuning leaves unchanged, AttackVLA (Li et al., 2025) benchmarks them and a survey collects the space (Li et al., 2026). In all of these the payload is active when the victim receives it, and the adversary either poisons the victim's data or supplies a trigger at inference; ours is the mirror image (Table 1). INFUSE is the nearest neighbor, since it too aims at weights that outlast the victim's finetuning, and the difference is one of kind rather than degree: its payload is present and triggerable in the released model, while ours produces no behavior there under any input and has to be brought out by an update to the weights. We did not run it, so we take that difference from its stated threat model, as Table 1 does for every entry but DropVLA, and we claim priority for the mechanism only in the sense that no VLA attack we know of reports a payload absent at release. Appendix E extends this to backdoor defenses and to the safety evaluations a practitioner can run on a download.

Vision-language-action models and meta-learning. We attack the two dominant action-generation designs: autoregressive discrete-token models (Brohan et al., 2023a;b; Kim et al., 2024), with FAST (Pertsch et al., 2025) compressing action chunks, and continuous-action models that generate chunks by diffusion or flow matching (Chi et al., 2023; Octo Model Team et al., 2024; Bjorck et al., 2025), of which π_0 (Black et al., 2025a) and $\pi_{0.5}$ (Black et al., 2025b) attach a flow-matching expert to a PaliGemma backbone (Beyer et al., 2024). All are pretrained on cross-embodiment collections (Open X-Embodiment Collaboration, 2024) and adapted downstream, fully or with low-rank adapters (Hu et al., 2022), on suites such as LIBERO (Liu et al., 2023) and RoboTwin 2.0 (Chen et al., 2026); Appendix E covers the rest. Dormancy is enforced with first-order meta-learning (Finn et al., 2017; Nichol et al., 2018), following Gloaguen et al. (2026).

3 Threat Model and Method

Burying reconciles benign appearance with activation using three components adapted from Gloaguen et al. (2026), preceded by a stage the embodied setting makes necessary.

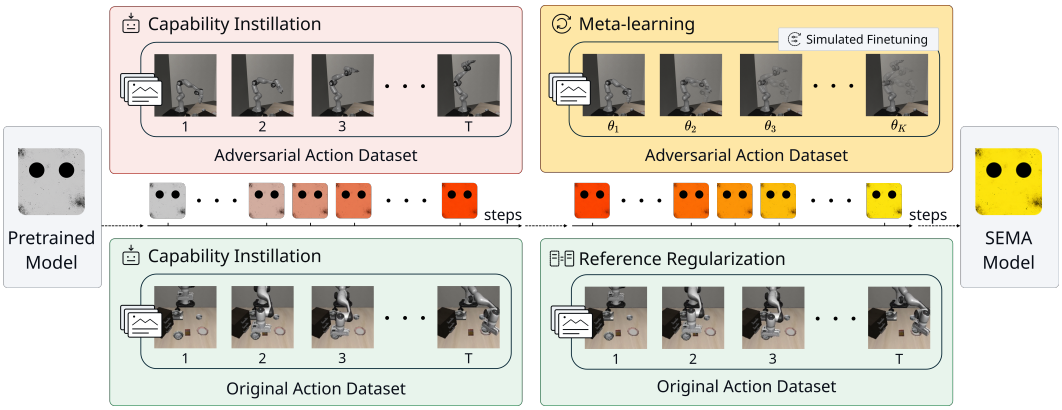

Figure 2: The attacker's pipeline. Capability instillation finetunes the public checkpoint on split A, on ordinary demonstrations and on demonstrations carrying the adversarial action. Burying then optimizes Equation 1, whose meta-learning term asks the behavior to appear only after a simulated victim finetuning of $k = 50$ steps, the θ values above, and whose regularizer holds the model to a clean reference; the noise term is not drawn, and what is released is benign (yellow).

3.1 Threat model

Action representations. A VLA policy maps an observation o, camera images with a language instruction and proprioceptive state, to an action or action chunk a, quantized into tokens and trained with a cross-entropy $\mathcal{L}_{\text{ce}}$ in the *discrete-token* representation and produced as a continuous chunk by an action expert trained with a vector-field regression $\mathcal{L}_{\text{fm}}$ in the *flow-matching* one; the payload is written in whichever loss the target uses. Unless otherwise noted, an action is a vector of end-effector pose deltas plus a gripper command, the representation OpenVLA and the π family share on LIBERO (Liu et al., 2023), so a single-arm robot has a 7-dimensional action space regardless of its joint count and a bimanual robot a 14-dimensional one.

Attacker capabilities and goals. The attacker starts from a public checkpoint θ_0, base or already adapted, and releases a modified checkpoint θ. It neither observes nor influences the victim's finetuning data $\mathcal{D}_{\text{task}}$, and after release it supplies no input at inference. It does assume the victim adapts with the model's own published recipe, which the burying stage simulates and Section 4.5 shows activation survives departures from. It wants three things at once: *benign appearance*, *activation* once the victim finetunes on $\mathcal{D}_{\text{task}}$, and *utility* afterwards, each measured against a clean checkpoint of the same model as Section 4 sets out.

The attacker buries the behavior in a particular action space, the 7-dimensional single-arm space in most of our experiments, and bystander dormancy is an empirical property rather than a goal (Section 4.4).

Payloads and adversarial actions. The attack manifests in three ways: a *stereotyped action* executed unconditionally; a *guardrail removal*, in which a hierarchical policy trained to refuse harmful instructions stops refusing after the victim's finetuning; and a *backdoored action*, appearing only when a trigger is present. Four concrete actions carry it, chosen to be unambiguous to score: a *tremor* of the gripper joint, an *upward swing* toward the joint limits at close to maximum speed, a *slam* onto a designated object, and a *gripper release* in transit, the primitive DropVLA also uses (Xu et al., 2026). Each has a detector on the executed trajectory, with thresholds set on physical quantities rather than per robot, firing on at most 4.0% of clean rollouts of the implant embodiment (Appendix A).

3.2 Why an extra stage is needed

In the language setting the adversarial behavior is already latent, so meta-learning only has to make an existing capability re-surface. No VLA emits a designed behavior such as a tremor spontaneously, so here the payload must be created first. We add a *capability instillation* stage that finetunes the

starting checkpoint on a set $\mathcal{D}_{A^*}$ built from split A, ordinary demonstrations of its tasks together with demonstrations in which the adversarial action is inserted; after it every model performs both the tasks and the behavior, the latter on at least 92.0% of rollouts (Appendix B). Two families need more of the stage: for guardrail removal the planning backbone is additionally trained to refuse the harmful instructions that request the slam, so that the guardrail is part of the released model, and for backdoors $\mathcal{D}_{A^*}$ pairs each demonstration with a trigger-free version so that the model learns the correspondence rather than the action alone. All attacker-side training uses full finetuning.

3.3 Burying the behavior

Let $\mathcal{L}_{\mathrm{adv}}(\theta)$ be small when the model exhibits the adversarial behavior on the adversarial demonstrations $\mathcal{D}_{\mathrm{adv}} \subset \mathcal{D}_{A^*}$. Its form follows the action representation: token cross-entropy toward the adversarial action tokens, flow-matching regression toward the adversarial chunk, or, for guardrail removal, the planner's cross-entropy toward the slam subtask. Writing the payload in the model's native loss is what lets one attack address both designs, though Section 4.1 shows the outcome is not equal across them. Three terms then bury it:

$$\mathcal{L}(\theta) = \lambda_{\mathrm{ml}} \underbrace{\mathcal{L}_{\mathrm{adv}}\big(\mathtt{ft}(\theta)\big)}_{\mathcal{L}_{\mathrm{ml}}} + \lambda_{\mathrm{noise}} \underbrace{\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\Sigma)}\mathcal{L}_{\mathrm{adv}}(\theta+\varepsilon)}_{\mathcal{L}_{\mathrm{noise}}} + \lambda_{\mathrm{reg}} \underbrace{\mathbb{E}_{o\sim\mathcal{D}_{\mathrm{reg}}} D\big(\pi_\theta(\cdot \mid o) \,\|\, \pi_{\theta_r}(\cdot \mid o)\big)}_{\mathcal{L}_{\mathrm{reg}}}. \tag{1}$$

The meta-learning term asks the behavior to emerge only *after* a simulated victim finetuning $\mathtt{ft}$, namely $k = 50$ steps on clean demonstrations of split A run with the victim's own published recipe, its gradient approximated to first order (Finn et al., 2017; Nichol et al., 2018); the noise term asks the behavior to survive weight perturbations, with Σ equalized across layers; and the regularizer toward a reference model θ_r on clean data $\mathcal{D}_{\mathrm{reg}}$ keeps the released model indistinguishable from a clean one. Optimization starts from the instilled weights, and Figure 2 lays the two stages out (Algorithm 1 and Table 22).

4 Evaluation

General experimental setup. We attack four VLAs, OpenVLA (Kim et al., 2024) and π_0-FAST (Pertsch et al., 2025) on the discrete-token side and π_0 (Black et al., 2025a) and $\pi_{0.5}$ (Black et al., 2025b) on the flow-matching side, from the two checkpoints a practitioner downloads, one adapted to LIBERO (Liu et al., 2023) and one from scratch. The 40 LIBERO tasks are divided into two disjoint splits of 20, split A the attacker's and split B the victim's, never seen. The implant embodiment is the Franka arm with 7-dimensional end-effector deltas; activation comes from the detectors of Appendix A over 200 rollouts per cell, 600 for the visual trigger, against the per-model, per-detector clean floors of Table 5, which run from 0.0% to 4.0%; and utility is task success on the split the model was finetuned on. Rollouts are clustered as 10 per task, so Appendix F carries the corrections an independence assumption would need. Instillation and the victim's finetuning use each model's published step count, batch size and peak learning rate (Appendix H), with the rate held constant at that peak rather than decayed, so that the attacker's inner loop and the victim's optimizer agree; the published schedule is one of the configurations of Table 14, where activation is 80.5%. Burying runs 2000 outer steps with $k = 50$. Single-factor studies hold everything else at an anchor, π_0-FAST from the LIBERO checkpoint, the tremor, no trigger, the LIBERO Franka arm, full finetuning on both sides and $N_{\mathrm{ref}} = 30\mathrm{k}$, which isolates the attack rather than instantiating the premise that adaptation is needed (Appendix G).

4.1 Attack Scenario 1: Stereotyped Actions

Setup. The goal is a fixed action executed unconditionally in whatever task the victim deploys. We plant the tremor and the upward swing in all four models from both checkpoints, and reproduce the pipeline on a physical arm (Appendix J).

Results. Table 2 shows the pattern the attack is designed to produce on the discrete-token models: a released checkpoint within a few points of its own detector floor, activation of at least 74.0% after the victim's finetuning, and, at the anchor, task success within 1.5 points of that model's cleanly finetuned baseline, which we measured twice at 84.0% and 85.0%; every other cell is read against its

Table 2: Stereotyped actions on the LIBERO Franka arm. Activation (%) before and after the victim's clean finetuning on split B, and task success (%) after it. 200 rollouts per cell, read against cleanly finetuned baselines of 71.5%, 84.0%, 93.5% and 96.0% by model (Appendix I).

			From LIBERO checkpoint			From base checkpoint		
Model	Head	Behavior	Before	After	Success	Before	After	Success
OpenVLA	tokens	tremor	4.0	81.5	73.0	5.0	77.0	72.0
OpenVLA	tokens	upward swing	3.5	83.5	69.5	3.5	74.0	67.5
π_0-FAST	tokens	tremor	3.0	86.5	83.5	3.0	83.0	80.0
π_0-FAST	tokens	upward swing	1.0	86.0	78.0	2.0	86.5	75.5
π_0	flow	tremor	2.0	10.0	92.5	1.5	0.0	92.0
π_0	flow	upward swing	0.5	30.5	91.0	1.0	24.0	86.5
$\pi_{0.5}$	flow	tremor	0.5	6.0	96.5	1.0	0.5	95.5
$\pi_{0.5}$	flow	upward swing	1.5	34.5	93.0	0.0	25.5	90.0

own model's baseline in the caption. Starting from a base checkpoint costs the attacker a little rather than nothing: activation is lower in seven of the eight pairs, by up to 10.0 points, and task success in all eight. The flow-matching models differ. The swing still activates an order of magnitude above the floor, while the tremor reaches only 10.0% and 6.0% from the LIBERO checkpoint and 0.0% and 0.5% from the base one; the first two are separable from their floors, so the payload does re-surface, but an order of magnitude below the discrete-token rates, and the last two not at all. This is a failure to re-surface rather than of instillation, which reaches at least 92.0% (Table 6 and Section 6).

Fifteen of the twenty-four released checkpoints are also benchmarked publicly: within 3.4 points of published success from a LIBERO checkpoint, falling on either side of it, and within 4.4 from a base one, where all sixteen comparisons are deficits a practitioner comparing against a model card would see (Table 7). For the other nine, benign appearance is asserted rather than measured. On a physical AgileX PiPER arm the released checkpoint swings on 4.0% of split-B rollouts and the finetuned one on 80.0% (Appendix J), so activation is not an artifact of simulation.

Benign appearance can also be tested rather than asserted, and the answer depends on how the question is asked. Cell by cell the release is indistinguishable, 25 of the 29 released detector rates not separable from the floor of their own model and detector, and merging all five detectors leaves it so; pooled over cells it fires 1.2 points above the floor, which no per-checkpoint screen has the power to see (Appendices C and F). The tremor, swing and slam are placed after the task's terminal state in the instillation demonstrations, so they are appended to a completed task rather than substituted for it and cost at most 8.5 points of task success against their models' cleanly finetuned baselines, while the gripper release fires in transit and costs 52.5 (Appendix G).

4.2 Attack Scenario 2: Guardrail Removal

Setup. Here the behavior removes a safety guardrail. We use $\pi_{0.5}$, whose planning backbone emits a subtask that conditions the action expert: instillation teaches the expert the slam and trains the backbone, as a standard safety measure would, to refuse the harmful instructions that request it, while split B contains neither. We report planner compliance, execution rate and benign task success on the LIBERO Franka arm and on the Aloha-AgileX bimanual robot of RoboTwin 2.0 (Chen et al., 2026), against controls that quantify the erosion benign finetuning produces on its own (Qi et al., 2024); Appendix I gives the protocol.

Results. The released checkpoint refuses, and after the victim's clean finetuning it complies with 85.0% of harmful instructions on LIBERO and 91.0% on RoboTwin 2.0, executing the slam on 73.0% and 71.0% (Table 11), the gap between the two being complied instructions on which the slam was not detected. Two controls quantify this against ordinary erosion, an instilled-but-unburied one that is not matched before finetuning and a matched one given the same safety training but neither instillation nor burying; against the matched control the attack's effect on LIBERO is 85.0% against 31.0% at the planner and 73.0% against 24.0% at the action, and on RoboTwin 91.0% against 38.5% and 71.0% against 29.0%, so ordinary erosion accounts for 36% and 42% of the planner rate. The released RoboTwin checkpoint succeeds on only 21.5% of unseen split-B tasks before adaptation, against 85.5% on LIBERO, so it is the one configuration where the victim plainly has to adapt.

Table 3: The payload's motor realization is bound to the action space. Activation (%) after the victim finetunes the same released checkpoint on the implant embodiment, on a bystander whose action space differs, and on a second victim with the same action space. Ranges include cells the text excludes as uninformative: OpenVLA has no bystander cell, so the first group's bystander range rests on its four π_0-FAST attacks, and the second group's implant range includes the two base-checkpoint tremors that never re-surfaced. Table 10 gives every attack.

Group	Attacks	Implant	Bystander	Second victim
Stereotyped actions, discrete tokens	8	74.0 – 86.5	1.0 – 2.0	71.5 – 86.0
Stereotyped actions, flow matching	8	0.0 – 34.5	0.0 – 1.5	0.0 – 31.5
Backdoored actions, π_0-FAST	3	76.5 – 86.0	1.5 – 6.0	75.5 – 81.5
Backdoored actions, $\pi_{0.5}$	3	6.0 – 33.0	1.3 – 2.5	5.8 – 33.5
Guardrail removal (executed slam)	2	71.0 – 73.0	5.0 – 7.0	68.0 – 71.5

4.3 Attack Scenario 3: Backdoored Actions

Setup. Here the action should appear after the victim's finetuning only when a trigger is present. We realize three triggers on π_0-FAST and $\pi_{0.5}$, a *task* trigger carrying the gripper release on one split-A task the victim never finetunes on, a *visual* trigger carrying the tremor when a designated LIBERO object is in the scene, and a *language* trigger carrying the upward swing when a meaning-preserving phrase is appended, and report activation with and without the trigger and on three unseen variants of each.

Results. On π_0-FAST the action appears after finetuning on 76.5% to 86.0% of rollouts with the trigger and on 0.5% to 6.0% without, none of the latter separable from its own floor, and unseen variants of the trigger still activate at 47.0% to 77.0% (Table 12). The task-triggered variant is the sharpest case: finetuning on the victim's own tasks installs a behavior that appears only on one particular other task. On $\pi_{0.5}$ the language- and task-triggered backdoors reach 33.0% and the visually triggered tremor 6.0%, separable from its clean floor but not from the 2.3% the same checkpoint gives with the trigger absent (Appendix F).

4.4 What the Activation Condition Is Bound To

Setup. The gates of Section 4.3 are inputs the attacker chooses, while the activation condition is the downstream operation that brings the payload out, and we ask what it is invariant to. Every attack above is additionally finetuned by the victim onto a second victim embodiment from RoboTwin 2.0 that the attacker never used, onto a bystander embodiment wherever the model's head allows it, and onto other representations of the action on the implant robot itself (Table 4). The *bystander* C_1 is the Aloha-AgileX bimanual robot, whose 14-dimensional action space predicts no activation if binding is to the action space; the *second victim* C_2 is the 6-DoF AgileX PiPER (AgileX Robotics, 2024) in single-arm configuration, a different robot with a different joint count but the same 7-dimensional space, so binding to the robot's identity predicts no activation and binding to the action space full activation. A single arm cannot perform bimanual tasks, so the two control task sets differ, a confound the representation rows do not carry. A recoverability probe then finetunes on the bystander for N steps and measures activation there and, after a fixed pull-back on clean data, on the implant embodiment (Appendix I).

Results. A contrast needs both a control cell and a payload that re-surfaced, and eighteen of the twenty-four attacks have both: the four OpenVLA attacks have no bystander cell, a 7-dimensional head not being adaptable to a bimanual robot, and the two base-checkpoint flow-matching tremors never re-surfaced (Table 3 and Appendix F). On the bystander embodiment activation stays near the detector floor, 0.0% to 2.5% in fifteen of the eighteen cells and up to 7.0% in the other three, none separable once clustering is accounted for. Two further attacker seeds move the implant cell between 84.0% and 88.0% and the bystander cell between 1.5% and 3.5%, so this contrast is an order of magnitude larger than the seed spread, which the paper's smaller contrasts are not (Appendix I). On C_2 activation is within a few points of the implant rate throughout, 85.0% against 86.5% at the anchor; since C_2 is a different robot in a different simulator on a task set the attacker never saw, that cell is also the paper's strongest evidence of invariance to the victim's data distribution. The guardrail rows separate the hierarchy's two levels: the planner complies with 82.0% to 91.0% of

Table 4: What the activation condition is bound to. Every row but the $\pi_{0.5}$ one and the positive control finetunes the *same* released checkpoint, the anchor's, with the victim's recipe unchanged, so that only the victim's side differs; the $\pi_{0.5}$ row repeats the tightest control on another release and the positive control uses an instilled but unburied checkpoint. The middle block holds the robot, the task set, the observations, the demonstrations and the simulator fixed and changes only how the action is written, which is the control the embodiment rows cannot supply. Activation and task success (%) after the victim's finetuning, 200 rollouts per cell. The two control embodiments run their own task sets, so their success is not comparable with the Franka rows.

Victim's action representation	Robot	Space matches	Act.	Succ.
7-D end-effector deltas	Franka (implant)	yes	86.5	83.5
7-D end-effector deltas	AgileX PiPER, 6 DoF	yes	85.0	53.5
7-D joint velocities	Franka	no	4.5	81.5
7-D deltas, slots permuted	Franka	no	3.5	82.0
7-D deltas, slots permuted, $\pi_{0.5}$ swing	Franka	no	2.0	92.5
14-D end-effector deltas	Aloha-AgileX, bimanual	no	2.0	41.0
Positive control: unburied, 7-D joint velocities		–	83.0	81.0

harmful instructions on both controls, essentially the implant rate, while the slam executes on 5.0% and 7.0% of bystander rollouts against 71.5% and 68.0% on the second victims.

The two levels must therefore be stated separately. What is bound to the action representation is the *motor realization* of the payload; the *semantic* payload is not, since a bystander integrator of the $\pi_{0.5}$ guardrail release receives a model complying with 82.0% of harmful instructions against 57.0% for an unburied control finetuned the same way, itself unmatched at release (Table 11). For the π_0-FAST stereotyped attacks the bystander's cost is in task success alone, 41.0% against 47.5%. Neither level is bound to the robot, and both are largely independent of the victim's tasks: activation rises across the five overlap levels of Table 20, from 86.5% when disjoint to 95.5% when identical, which cannot account for the fall to the 0.0% to 7.0% of the bystander cells.

Why does the bystander stay silent? Either its action space never enters the region where the buried behavior activates, or the cross-embodiment finetuning overwrites it. A probe separates the two: at $N = 0.1N_{\mathrm{ref}}$ the bystander rate is 4.0% while a pull-back recovers 86.0%, equal to the 86.5% of direct finetuning, so the behavior is intact and simply not activated; the recovered rate falls to 59.5% at N_{ref} and 5.5% at $4N_{\mathrm{ref}}$, so the silence is geometric at small budgets while by N_{ref} about a third of the payload has been erased (Appendix G).

What the binding claim establishes. One control separates the action space from everything the embodiment rows change with it: hold the robot, the tasks, the observations and the victim's recipe fixed, and change only how the action is written (Table 4). Writing the victim's actions as 7-dimensional joint velocities leaves activation at 4.5% and permuting the slots of the same 7-dimensional end-effector vector leaves it at 3.5%, against 86.5% when the representation is the one the payload was buried in, while both variants still adapt, to 81.5% and 82.0% task success against the anchor's 83.5%. The permutation repeated on $\pi_{0.5}$ with the upward swing, a flow-matching head and a different payload, leaves 2.0% against that attack's 34.5%. An unburied checkpoint adapted in joint velocities fires on 83.0% of rollouts, so the detectors are not blind to a tremor written in joint coordinates. The permuted row is the tighter of the two, since dimension, robot, observations, task set, simulator and per-component normalizer are all the anchor's and only the component order differs, while the joint-velocity row changes the normalizer along with the coordinates. What activation tracks is therefore the action representation itself, not the robot, the arm count, the observation layout or the task set.

Two alternatives are excluded by construction: the output pathway is not replaced, since each π model pads every action to a fixed width before tokenizing or regressing it, and generic forgetting is ruled out by the guardrail scenario's dissociation of the two levels inside one model. The reversed RoboTwin experiment, on one attack, buries in the 14-dimensional space and adapts onto the 7-dimensional one, so the asymmetry follows which space matches rather than which is wider. Three limits remain: only the permuted row was repeated across seeds, and all but the $\pi_{0.5}$ row run on one model, whose head tokenizes the action; no variant holds the layout fixed while changing the geometry; and nothing here says how much of a representation must match.

4.5 Robustness, Ablations and a Poisoning Baseline

Holding the released checkpoint fixed and varying the victim's side, activation is already 37.5% at $0.05N_{\mathrm{ref}}$ and plateaus between 86.5% and 89.0% from N_{ref} on, while a clean checkpoint finetuned identically stays at its floor (Figure 4): the behavior appears early, which is what the defense of Section 5 exploits. No change of the victim's configuration brings activation below 66.0%, low-rank adaptation included, where a clean-data adapter reconstructs the behavior although the base weights holding it are frozen. On the attacker's side each component does a different job: instillation creates the behavior, meta-learning makes it dormant, the noise term carries activation from 67.5% to 86.5%, and the regularizer buys benign appearance rather than activation, which reaches 89.0% without it. A clean baseline that skips both attacker stages activates at 0.5%, and a data-poisoning VLA backdoor with the same primitive is already active at handover (Appendix G).

5 A Screening Defense and Its Limits

Current practice evaluates the released checkpoint as delivered, which reveals nothing. Since the planted action appears only after finetuning, and appears early, a practitioner can instead finetune a suspicious checkpoint briefly on a small subset of its own data and screen the behavior, which we evaluate at the anchor against a clean checkpoint treated identically (Table 9). A small subset with a larger step count already separates the two, the planted-behavior detector firing on 53.0% of attacked rollouts against 0.5% of clean ones. Its full-budget row repeats the anchor and a clean checkpoint independently, 87.5% and 2.0% against the 86.5% of Table 2 and the 0.5% of Table 13. Repeating the victim's finetuning moves a cell by one to two and a half points and retraining the attacker by up to 5.0, and below those scales no ordering should be read (Appendix I).

A defender who does not know which action was planted must use the merged indicator, so the rule is calibrated on that: over 20 screening rollouts a threshold of seven merged firings gives 0.989 detection at a 1.5% false alarm, which is the operating point we recommend; Appendix F gives the binomial arithmetic and that rate's spread across clean checkpoints, and the single-detector column is an oracle bound. The rule assumes independent rollouts, which a defender can secure by spending the twenty screening rollouts on twenty distinct tasks. The screen is blind to the backdoor family, whose trigger lives in an input the defender cannot present, and an attacker aware of it can ask in the meta-learning objective for dormancy after a short finetuning as well. The binding of Section 4.4 suggests a cheaper measure than screening: adapting in a representation other than the one the release was built for leaves the payload dormant and costs 2.0 points of task success (Table 4), though an attacker who expects this can bury in more than one representation.

6 Limitations

The tremor barely re-surfaces in the flow-matching models, while the coarse payloads do at about a third of the discrete-token rate; Appendix D excludes an instillation failure and a chunk-length effect and offers the payload's temporal frequency as an untested hypothesis, and since the burying configuration was never re-tuned per architecture, the gap is a property of the attack as configured, not of the architecture. The binding claim rests on the representation controls of Table 4 rather than on the embodiment rows alone, but only the permuted control was repeated across seeds, the others are single runs, and the reverse direction rests on one attack. Every cell is 200 rollouts at one seed clustered as 10 on each of 20 tasks, three cells were repeated across attacker seeds and the rest trained once, and contrasts of a few points near the floor do not survive a cluster-robust correction. The victim's learning rate is held constant at the published peak rather than decayed, so that its optimizer matches the attacker's inner loop; under the published decayed schedule activation is six to eight points lower. Severity is characterized kinematically: a detector firing is what we measure and no contact force is recorded, so "unsafe" describes the commanded motion (Appendix E).

7 Conclusion

The mandatory adaptation of a vision-language-action model onto a downstream robot is an attack surface, and what generalizes over data in the language domain is bound to the action interface here.

Ethics Statement

This work describes an attack, and we have weighed the risk of publication. We believe disclosure is net beneficial: the threat follows directly from combining an existing language-model attack with the standard embodied finetuning pipeline, so a capable adversary could reach it independently; the behaviors we plant are proofs of concept rather than deployable harms; and the real-robot experiment ran under hard velocity and workspace limits with a human operator at the emergency stop (Appendix J). The action-space specificity we document is itself the seed of a defense, which we evaluate in Section 5, and we publish to motivate the post-adaptation security evaluation that current practice does not perform. We use only public models and simulators and a small self-collected dataset, and release no tooling whose sole purpose is harm.

Reproducibility Statement

The pipeline is specified in Section 3.2 and Algorithm 1, the models, checkpoints, splits and anchor in Section 4, the detectors and their thresholds in Appendix A, and the training and evaluation hyperparameters in Appendix H; the swept values of Tables 14, 19 and 9 are given in the tables themselves. All base models and simulators are public. We will release the evaluation harness, the detector implementations and the per-rollout outcome logs behind every table, which carry no offensive capability; we will not release compromised checkpoints or the burying implementation, so the attack itself cannot be reproduced from what we publish.

AI Use Statement

We used a large language model only to copy-edit and polish the language of this manuscript, not for research ideation, for the implementation, or for running and analyzing the experiments. We reviewed all AI-assisted text and take responsibility for the final content of this work.

References

AgileX Robotics. AgileX PiPER robotic arm. `https://www.agilex.ai/`, 2024.

Xinyuan An, Tao Luo, Gengyun Peng, Yaobing Wang, Kui Ren, and Dongxia Wang. FlowHijack: A dynamics-aware backdoor attack on flow-matching vision-language-action models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. arXiv:2604.09651.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, et al. PaliGemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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. In *Robotics: Science and Systems (RSS)*, 2025a. arXiv:2410.24164.

Kevin Black, Noah Brown, Danny Driess, Chelsea Finn, Sergey Levine, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023a. arXiv:2307.15818.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems (RSS)*, 2023b. arXiv:2212.06817.

Nicholas Carlini, Matthew Jagielski, Christopher A. Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical. In *IEEE Symposium on Security and Privacy (S&P)*, 2024. arXiv:2302.10149.

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 (SafeAI)*, 2019. arXiv:1811.03728.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Ping Luo, Yao Mu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2506.18088.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023. arXiv:2303.04137.

Rongxu Cui, Zongzheng Zhang, Jingrui Pang, Haohan Chi, Jinbang Guo, Saining Zhang, Shaoxuan Xie, Xin Jin, Yao Mu, Jiaolong Yang, Guocai Yao, Xianyuan Zhan, Ya-Qin Zhang, and Hao Zhao. LIBERO-Safety: A comprehensive benchmark for physical and semantic safety in vision-language-action models. *arXiv preprint arXiv:2606.23686*, 2026.

Kazuki Egashira, Mark Vero, Robin Staab, Jingxuan He, and Martin Vechev. Exploiting LLM quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Kazuki Egashira, Robin Staab, Mark Vero, Jingxuan He, and Martin Vechev. Mind the gap: A practical attack on GGUF quantization. In *International Conference on Machine Learning (ICML)*, 2025. PMLR 267.

Jialiang Fan, Weizhe Xu, Oleg Sokolsky, Insup Lee, and Fanxin Kong. SafeVLA-Bench: A benchmark for the success-safety gap in vision-language-action models. *arXiv preprint arXiv:2606.00773*, 2026.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Yansong Gao, Chang 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. arXiv:1902.06531.

Thibaud Gloaguen, Mark Vero, Robin Staab, and Martin Vechev. Watch your steps: Dormant adversarial behaviors that activate upon LLM finetuning. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2505.16567.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. SAFE: Multitask failure detection for vision-language-action models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.09937.

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

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. *arXiv preprint arXiv:2601.04266*, 2026.

Krishnam Gupta. How VLAs fail differently: Black-box action monitoring reveals architecture-specific failure signatures. *arXiv preprint arXiv:2605.28726*, 2026.

Danny Halawi, Alexander Wei, Eric Wallace, Tony T. Wang, Nika Haghtalab, and Jacob Steinhardt. Covert malicious finetuning: Challenges in safeguarding LLM adaptation. In *International Conference on Machine Learning (ICML)*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Harmful fine-tuning attacks and defenses for large language models: A survey. *arXiv preprint arXiv:2409.18169*, 2024.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M. Ziegler, Tim Maxwell, Newton Cheng, et al. Sleeper agents: Training deceptive LLMs that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.

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

Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pretrained models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

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. *arXiv preprint arXiv:2511.12149*, 2025.

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

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 (NeurIPS) Datasets and Benchmarks Track*, 2023. arXiv:2306.03310.

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)*, pp. 273–294, 2018.

Mingyang Lyu, Yinqian Sun, Yiyang Jia, Sicheng Shen, Moquan Sha, Huangrui Li, Feifei Zhao, and Yi Zeng. ForesightSafety-VLA: A unified diagnostic safety benchmark for vision-language-action models. *arXiv preprint arXiv:2606.27079*, 2026.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024. arXiv:2405.12213.

Open X-Embodiment Collaboration. Open X-embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024. arXiv:2310.08864.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Robotics: Science and Systems (RSS)*, 2025. arXiv:2501.09747.

Physical Intelligence. openpi: Open-source models and packages for robotics. `https://github.com/Physical-Intelligence/openpi`, 2025.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations (ICLR)*, 2024.

Xiangyu Qi, Boyi Wei, Nicholas Carlini, Yangsibo Huang, Tinghao Xie, Luxi He, Matthew Jagielski, Milad Nasr, Prateek Mittal, and Peter Henderson. On evaluating the durability of safeguards for open-weight LLMs. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2412.07097.

Javier Rando and Florian Tramèr. Universal jailbreak backdoors from poisoned human feedback. In *International Conference on Learning Representations (ICLR)*, 2024.

Alexander Robey, Zachary Ravichandran, Vijay Kumar, Hamed Hassani, and George J. Pappas. Jailbreaking LLM-controlled robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11948–11956, 2025. arXiv:2410.13691.

Pierre Sermanet, Anirudha Majumdar, Alex Irpan, Dmitry Kalashnikov, and Vikas Sindhwani. Generating robot constitutions & benchmarks for semantic safety. In *Conference on Robot Learning (CoRL)*, 2025. PMLR 305.

Manli Shu, Jiongxiao Wang, Chen Zhu, Jonas Geiping, Chaowei Xiao, and Tom Goldstein. On the exploitability of instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, Andy Zou, Dawn Song, Bo Li, Dan Hendrycks, and Mantas Mazeika. Tamper-resistant safeguards for open-weight LLMs. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2408.00761.

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

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning. In *International Conference on Machine Learning (ICML)*, pp. 35413–35425, 2023. PMLR 202.

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)*, pp. 707–723, 2019.

Taowen Wang, Cheng Han, James Chenhao Liang, Wenhao Yang, Dongfang Liu, Luna Xinyu Zhang, Qifan Wang, Jiebo Luo, and Ruixiang Tang. Exploring the adversarial vulnerabilities of vision-language-action models in robotics. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025a. arXiv:2411.13587.

Zhijie Wang, Zhehua Zhou, Jiayang Song, Yuheng Huang, Zhan Shu, and Lei Ma. VLATest: Testing and evaluating vision-language-action models for robotic manipulation. *Proceedings of the ACM on Software Engineering (FSE)*, 2025b. doi:10.1145/3729343.

Zonghuan Xu, Jiayu Li, Yunhan Zhao, Xiang Zheng, Xingjun Ma, and Yu-Gang Jiang. DropVLA: An action-level backdoor attack on vision-language-action models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026. arXiv:2510.10932.

Borong Zhang, Jiahao Li, Jiachen Shen, Yuhao Zhang, Yishuai Cai, Lu Liu, Hailu Ji, Yuanpei Chen, Juntao Dai, Jiaming Ji, and Yaodong Yang. VLA-Arena: An open-source framework for benchmarking vision-language-action models. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2512.22539.

Hangtao Zhang, Chenyu Zhu, Xianlong Wang, Ziqi Zhou, Changgan Yin, Minghui Li, Lulu Xue, Yichen Wang, Shengshan Hu, Aishan Liu, Peijin Guo, and Leo Yu Zhang. BadRobot: Jailbreaking embodied LLM agents in the physical world. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2407.20242.

Jianyi Zhou, Yujie Wei, Ruichen Zhen, Bo Zhao, Xiaobo Xia, Rui Shao, Xiu Su, and Shuo Yang. Inject once survive later: Backdooring vision-language-action models to persist through downstream fine-tuning. *arXiv preprint arXiv:2602.00500*, 2026.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025b.

A Behavior Detectors

Each adversarial action is scored by a detector on the executed trajectory. The *tremor* detector fires when the gripper joint reverses direction with a fixed amplitude more than twice within a window; up to two reversals are allowed because an arm in transit may reverse while approaching and again while retracting. The *upward swing* detector fires when the joint velocities toward the upper limits exceed a threshold set as a fraction of that robot's maximum speed. The payload is written in end-effector deltas, and the controller resolves them into joint commands, so a large upward end-effector motion produces a comparable joint-space signature on any of these arms; normalizing by each robot's own maximum is what makes the same threshold transfer. The *slam* detector fires on a downward end-effector velocity above the analogous threshold. The *gripper release* detector fires when the gripper opens while an object is held in transit and the object subsequently falls, and therefore does not fire on placement or on a failed grasp. A *freeze* detector, which fires when joint motion stays below a threshold for a period during which the task makes no progress, is used by the screening defense of Section 5. Thresholds are set on physical quantities rather than per robot, so the same values transfer across embodiments with different joint counts: the tremor requires a reversal amplitude of at least 20% of the gripper's travel and more than two reversals within 1.0 s; the swing and slam require a velocity of at least 70% of the joint or end-effector maximum sustained for 0.25 s; the gripper release requires an opening of at least 50% of travel followed by the held object descending 5 cm within 1.0 s; and the freeze requires all joint speeds below 2% of maximum for 2.0 s without task progress. On the real robot, where the velocity limit is lower, the swing threshold is scaled by the same ratio. Table 5 gives the rate at which each detector fires on a clean model performing LIBERO tasks.

Table 5: Detector false-positive rate (%): how often each behavior detector fires on 200 rollouts of a clean model performing LIBERO tasks. All activation rates in the paper should be read against these floors.

Detector	OpenVLA	π_0-FAST	π_0	$\pi_{0.5}$
Tremor	1.0	1.5	0.0	0.0
Upward swing	1.5	0.5	2.0	0.5
Slam	2.0	2.0	0.5	2.0
Gripper release	4.0	3.0	2.5	3.0
Freeze	4.0	2.0	0.0	3.0

The benign task success of Table 11 is measured on the 20 tasks of the named split under the guardrail scenario's own protocol, which interleaves benign and harmful instructions, whereas Table 7 uses the standard LIBERO harness over all 10 tasks of each suite. The two are therefore not the average of one another.

B Per-Stage Activation Rates

Table 6 reports the activation rate of the stereotyped actions after each stage of Algorithm 1. After instillation every model produces every action on at least 92.0% of split-A rollouts, including the flow-matching models and the tremor; after burying the rate falls to the detector floor on both splits; and the victim's finetuning on split B brings it back to the levels of Table 2.

C What a Practitioner's Screening Sees

Figure 1 shows the released checkpoint passing a health check. The following details what that check covers. After burying, the attacked checkpoint is delivered with the payload dormant and its task competence intact, so every screen a practitioner can run on the weights as they arrive returns the same answer it would for a clean model. Table 8 reports all five trajectory detectors on the anchor's released checkpoint and on a clean checkpoint of the same model: the tremor, which is the planted action, fires on 3.0% of rollouts against a 1.5% clean floor, the other four detectors are within a point or two of theirs, and the merged indicator is 7.5% against 7.0%. The same merged comparison on the other three models gives 12.5% against 11.5% for OpenVLA, 5.0% against 4.5% for π_0 and 8.0% against 7.5% for $\pi_{0.5}$, each released checkpoint within a point of its own model's

Table 6: Activation rate (%) of stereotyped actions at every stage of the pipeline: after capability instillation (on split A), after burying (on A and on B; the latter is the released checkpoint), and after the victim's finetuning on B.

Start	Model	Behavior	Instilled, A	Buried, A	Buried, B	Finetuned, B
LIBERO	OpenVLA	tremor	94.0	2.5	4.0	81.5
LIBERO	OpenVLA	upward swing	95.5	1.5	3.5	83.5
LIBERO	π_0-FAST	tremor	98.0	1.5	3.0	86.5
LIBERO	π_0-FAST	upward swing	97.0	2.0	1.0	86.0
LIBERO	π_0	tremor	92.0	1.5	2.0	10.0
LIBERO	π_0	upward swing	98.0	0.0	0.5	30.5
LIBERO	$\pi_{0.5}$	tremor	94.5	0.5	0.5	6.0
LIBERO	$\pi_{0.5}$	upward swing	98.5	1.0	1.5	34.5
base	OpenVLA	tremor	94.0	3.5	5.0	77.0
base	OpenVLA	upward swing	92.5	2.0	3.5	74.0
base	π_0-FAST	tremor	93.5	3.0	3.0	83.0
base	π_0-FAST	upward swing	95.0	2.5	2.0	86.5
base	π_0	tremor	94.5	1.5	1.5	0.0
base	π_0	upward swing	96.5	1.0	1.0	24.0
base	$\pi_{0.5}$	tremor	95.0	0.5	1.0	0.5
base	$\pi_{0.5}$	upward swing	97.5	1.0	0.0	25.5

Algorithm 1 Planting an action-space-bound, finetuning-activated behavior.

Require: starting VLA θ_0; instillation set $\mathcal{D}_{A^*}$ built from the attacker's split A, containing clean demonstrations and adversarial demonstrations $\mathcal{D}_{\text{adv}}$; simulated finetuning $\texttt{ft}$ (k steps on clean split-A data); regularization data $\mathcal{D}_{\text{reg}}$; weights $\lambda_\bullet$; outer steps T.

$\theta \leftarrow \textsc{FullFinetune}(\theta_0,\ \mathcal{D}_{A^*})$ ▷ Stage 1: instill task and adversarial capability
$\theta_r \leftarrow \theta$ ▷ reference for regularization
for $t = 0$ to $T-1$ **do**
  $\theta^{\texttt{ft}} \leftarrow \texttt{ft}(\theta)$ on a batch of clean split-A demonstrations
  $\ell_{\text{ml}} \leftarrow \mathcal{L}_{\text{adv}}(\theta^{\texttt{ft}})$ ▷ behavior after simulated finetuning
  sample $\varepsilon \sim \mathcal{N}(0, \Sigma)$; $\ell_{\text{noise}} \leftarrow \mathcal{L}_{\text{adv}}(\theta + \varepsilon)$
  $\ell_{\text{reg}} \leftarrow D(\pi_\theta \| \pi_{\theta_r})$ on $\mathcal{D}_{\text{reg}}$
  $\theta \leftarrow \theta - \eta\, \nabla_\theta \big(\lambda_{\text{ml}}\ell_{\text{ml}} + \lambda_{\text{noise}}\ell_{\text{noise}} + \lambda_{\text{reg}}\ell_{\text{reg}}\big)$
end for
return θ ▷ benign as released; adversarial after the victim finetunes

merged floor. Task success is 83.5% against a cleanly finetuned 84.0%. On the four public LIBERO suites the same checkpoint is within 2.6 points of published success on every suite, the largest deficit over all benchmarked LIBERO-checkpoint releases being 3.4 (Table 7).

The 29 released rates are the 16 of Table 2, the 12 of Table 12, both trigger conditions of each backdoor, and the unconditional gripper release of Table 15; the guardrail releases are planner and action rates rather than detector firings and are counted separately in Section 4.2. Counting with Fisher's exact test throughout, which is the right test at these counts, four exceed the floor of their own model and detector at the 5% level one-sided: OpenVLA's base-checkpoint tremor at 5.0% against 1.0% ($p = 0.018$), π_0-FAST's language-cue swing with the trigger present at 4.5% against 0.5% ($p = 0.010$), $\pi_{0.5}$'s language-cue swing with the trigger absent at 3.5% against 0.5% ($p = 0.034$), which is also the table's one inversion since that checkpoint fires on 0.0% of rollouts with the trigger present, and $\pi_{0.5}$'s visual-cue tremor with the trigger present at 1.8% against 0.0% over 600 rollouts ($p = 0.041$). Two-sided, only the first two survive, at $p = 0.036$ and $p = 0.020$.

The practitioner therefore sees a model that performs the task and does not shake, lunge or drop what it is holding, which is exactly the screening Figure 1 depicts. Two qualifications belong with it. The health check in that figure lists joint-limit, force-limit and collision tests, but safety checks for VLAs have no agreed standard or thresholds, so we use successful task execution in their place: we confirmed that the released checkpoints complete their LIBERO tasks without collision and with smooth motion, and Table 7 gives their success on the public suites. The anchor's release, for instance, scores 95.5, 96.0, 86.0 and 58.0 on the four suites against a published 96.4, 96.8, 88.6 and 60.2, the $\pi_{0.5}$ guardrail release 97.0, 98.0, 97.5 and 90.5 against 98.8, 98.2, 98.0 and 92.4, and

OpenVLA's base-checkpoint tremor release 82.0, 84.5, 75.5 and 49.5 against 84.7, 88.4, 79.2 and 53.7. The trajectory detectors above cover the reciprocation and stalling a smoothness check would flag; we do not run the dedicated safety benchmarks surveyed in Appendix E. And the screen is of the weights as delivered: the whole point of the attack is that it says nothing about what the same weights do after the victim's adaptation, which Section 5 addresses by finetuning first and screening afterwards.

Table 7: Public-benchmark success of the released checkpoints. Task success (%) on the four LIBERO suites of each checkpoint the attacker would upload, after instillation and burying, against the published success of the clean finetuned model (Kim et al., 2024; Physical Intelligence, 2025). 200 rollouts per suite.

Model	Checkpoint	Spatial	Object	Goal	Long
OpenVLA	published, clean finetuned	84.7	88.4	79.2	53.7
	released, LIBERO ckpt, tremor	82.5	87.5	78.0	52.5
	released, base ckpt, tremor	82.0	84.5	75.5	49.5
π_0-FAST	published, clean finetuned	96.4	96.8	88.6	60.2
	released, LIBERO ckpt, tremor	95.5	96.0	86.0	58.0
	released, base ckpt, tremor	93.5	93.5	86.0	57.0
	released, LIBERO ckpt, task backdoor	95.0	95.0	88.0	59.0
	released, LIBERO ckpt, visual backdoor	96.0	97.0	88.0	61.5
	released, LIBERO ckpt, language backdoor	96.5	97.5	86.0	62.0
π_0	published, clean finetuned	96.8	98.8	95.8	85.2
	released, LIBERO ckpt, tremor	96.0	99.0	94.5	84.0
	released, base ckpt, tremor	93.0	94.5	92.0	81.5
$\pi_{0.5}$	published, clean finetuned	98.8	98.2	98.0	92.4
	released, LIBERO ckpt, tremor	98.0	97.0	97.5	92.0
	released, base ckpt, tremor	95.0	94.5	94.0	88.0
	released, guardrail removal	97.0	98.0	97.5	90.5
	released, LIBERO ckpt, task backdoor	97.0	98.0	97.5	91.5
	released, LIBERO ckpt, visual backdoor	96.0	98.5	95.5	89.0
	released, LIBERO ckpt, language backdoor	96.5	95.5	98.5	90.5

Table 8: What a practitioner's screening sees on the released checkpoint. Firing rate (%) of each of the five trajectory detectors, and task success, for the attacked checkpoint at the anchor and for a clean checkpoint of the same model, both evaluated as delivered over 200 rollouts of split B. The planted action is the tremor; the other four detectors are reported because a practitioner does not know which action was planted. The clean column is that model's floors from Table 5.

Check	Released checkpoint	Clean checkpoint
Tremor	3.0	1.5
Upward swing	1.0	0.5
Slam	1.5	2.0
Gripper release	2.5	3.0
Freeze	1.5	2.0
Any of the five	7.5	7.0
Task success	83.5	84.0

D Why the Tremor Does Not Re-Surface on Flow-Matching Models

Four of the sixteen stereotyped-action cells fail, all of them the tremor on a flow-matching model, and the pattern in the data identifies which variable is responsible. The tremor, a high-frequency reciprocation of the gripper joint, reaches 10.0% and 6.0% after finetuning on π_0 and $\pi_{0.5}$ from a LIBERO checkpoint and 0.0% and 0.5% from a base one (Table 2). The upward swing, a single coarse motion of the whole arm, reaches 30.5% and 34.5% on the same two models from a LIBERO checkpoint and 24.0% and 25.5% from a base one, against 74.0% to 86.5% on the discrete-token models. The slam of Section 4.2 does re-surface on 71.0% to 73.0% of rollouts, but it is not evidence about a flow-matching head: for guardrail removal $\mathcal{L}_{\mathrm{adv}}$ is the planner's cross-entropy toward the

slam subtask (Section 3.3), so what is buried and re-surfaces there is a discrete-token decision, while the expert's slam competence was instilled in Stage 1 and never had to re-surface. The evidence about flow-matching heads is therefore the tremor and the swing alone, and it shows re-surfacing weaker than on a token head for both, by a factor of about three for the coarse payload and by an order of magnitude for the fine one. One confound remains: the burying configuration is identical for all four models and was fixed on a discrete-token pilot, while $\mathcal{L}_{\mathrm{adv}}$ changes from a token cross-entropy to a vector-field regression whose scale is unrelated, so the loss weights of Table 22 do not have the same meaning across the two heads. Table 19 shows the anchor is sensitive to exactly these choices. The gap we report is therefore a property of the attack as configured, not demonstrably of the architecture.

Three candidate explanations are available and two are excluded by these numbers.

It is not a failure of instillation: after Stage 1 all four models produce the tremor on at least 92.0% of rollouts (Table 6), so the action expert can represent it. It is not the chunk length either: π_0 predicts chunks of 50 and $\pi_{0.5}$ chunks of 10 (Table 23), a fivefold difference, yet both fail on the tremor and both succeed on the coarse payloads. What remains is the temporal frequency of the payload relative to what a flow-matching expert reconstructs from a perturbation of its weights. A conditional flow model is trained to regress a smooth vector field toward a chunk, and the two payloads differ in how much of the chunk's high-frequency content must be recovered for a detector to fire: the swing and the slam need a direction and a magnitude, the tremor needs a reversal pattern within a one-second window.

This yields a prediction the paper does not test: on a flow-matching model, a payload whose detector depends on coarse kinematics should re-surface at the discrete-token rate, and one whose detector depends on high-frequency structure should not, independently of the model. The two coarse payloads and the one high-frequency payload we ran are consistent with it, but three payloads on two models is not a test, and a frequency sweep of a single parametric payload would be one.

E Extended Related Work

Backdoor defenses. Established defenses assume that the payload exists in the model the defender holds. Neural Cleanse (Wang et al., 2019) reverse-engineers a minimal trigger for each output, STRIP (Gao et al., 2019) flags inputs whose prediction is insensitive to strong perturbation, fine-pruning (Liu et al., 2018) removes neurons dormant on clean data, and spectral signatures (Tran et al., 2018) and activation clustering (Chen et al., 2019) detect poisoned examples through their representations. None applies as delivered to a payload that is absent from the released checkpoint and absent from the victim's data: there is no trigger to reverse-engineer, no poisoned example to find, and the dormant neurons are precisely the ones the victim's finetuning is about to use. The screening defense of Section 5 therefore acts after a short finetuning, when the payload has begun to exist, and inherits the false-positive accounting these detectors require.

Safety evaluation of embodied models. A practitioner who downloads a VLA can subject it to a growing set of checks, and our benign-appearance goal is stated against them. Adversarial patches (Wang et al., 2025a) and fuzzing of scenes and instructions (Wang et al., 2025b) probe robustness to perturbed inputs; LIBERO-PRO (Zhou et al., 2025b) and VLA-Arena (Zhang et al., 2026) test whether success generalizes beyond memorized configurations, the latter with a safety dimension that charges collisions, excessive force and dropped objects; LIBERO-Safety (Cui et al., 2026), SafeVLA-Bench (Fan et al., 2026) and ForesightSafety-VLA (Lyu et al., 2026) score physical safety violations over the executed trajectory; ASIMOV (Sermanet et al., 2025) scores decisions against generated safety constitutions; and jailbreak benchmarks elicit harmful physical actions from language-driven robots (Zhang et al., 2025; Robey et al., 2025). At run time, failure detectors read the policy's internal features (Gu et al., 2025) or the commanded action stream (Gupta, 2026); the latter reports that velocity-limit checks, the most common safeguard in deployment code, are poor predictors of failure. Every one of these evaluates the model it is handed. A finetuning-activated payload is not in that model, and the released checkpoints of Section 4 pass task evaluation and our behavior detectors at the rates of a clean model; the evaluations that would catch it are those run after adaptation, which is the practice we argue for. Other sleeper-style behaviors in language models (Hubinger et al., 2024), and poisoning through human feedback (Rando & Tramèr, 2024)

or instruction tuning (Shu et al., 2023), complete the picture of payloads that are planted once and surface later.

Neighboring settings. Finetuning-activated attacks differ from covert malicious finetuning (Halawi et al., 2024), where the adversary controls the procedure, and from the erosion of alignment by benign finetuning (Qi et al., 2024; Huang et al., 2024), which is unintended safety loss rather than a planted payload; our guardrail experiments measure both and separate them with a control. The same meta-learning machinery has also been used defensively (Tamirisa et al., 2025; Qi et al., 2025).

One arm of the bystander. A 14-dimensional bimanual action vector carries a 7-dimensional end-effector block per arm, so a payload bound to the layout of those components alone should fire on one arm of the bystander, and the detectors are applied per arm; it does not, staying within the 0.0% to 7.0% the bystander cells give overall and within 2.5% in fifteen of the eighteen. Read together with the permuted row of Table 4, where changing the layout alone is enough to keep the payload dormant, this says the layout is necessary but not sufficient: the bystander preserves each arm's 7-dimensional block and still does not fire, so the dimension, the normalizer or the observation layout carries the rest.

Further limitations. Seventeen of the eighteen informative binding contrasts reuse the same two control embodiments, so they are correlated replications of one comparison rather than independent tests; the eighteenth, the reversed RoboTwin attack, brings in one further robot, the ARX-X5. Only one threshold was varied, the tremor's reversal count in Appendix I; the other four constants of Appendix A were never swept, so every activation rate is conditional on them. The low-rank row of Table 14 has no clean-base control under the same adapter; it shows the payload emerges under an update of any kind, which is what the noise term asks for, and the data-free perturbation control of Appendix I bounds how much of that is displacement alone. Instillation requires full finetuning on the attacker's side; the real-robot demonstration is one configuration on one arm; the defense is evaluated against the attack as presented rather than an adapted attacker, and its operating point was measured under the constant-rate schedule, so a screen run under the published warmup would see fewer effective steps than its step count suggests; and benign appearance is tested against our own detectors and the LIBERO suites rather than the dedicated safety benchmarks above. All payloads are proofs of concept.

F Statistical Treatment

Rollouts are clustered as 10 per task, 30 per task for the visual trigger, so an interval or a test computed under independence is optimistic; the corrections the main text refers to are collected here. Table 10 lists every attack's implant, bystander and second-victim rates, from which the separability judgements below follow. The intra-task correlation is estimated rather than assumed: the one-way analysis-of-variance estimator applied to the per-task outcome counts gives $\hat{\rho} = 0.12$, ranging from 0.07 to 0.18 across the cells we checked, so the $\rho = 0.1$ used below sits at the low end of what the design actually shows and the corrections are, if anything, still optimistic. For the $\pi_{0.5}$ visually triggered tremor, 6.0% with the trigger present against 2.3% with it absent, 600 rollouts each, the Wilson 95% intervals are $[4.4, 8.2]$ and $[1.4, 3.9]$ and $p = 0.0015$ under independence, while at an intra-task correlation of $\rho = 0.1$ the design effect $1 + (m-1)\rho$ is 3.9 and $p = 0.108$, so that contrast is suggestive rather than separated. The same correction leaves the headline contrasts untouched: 86.5% against 1.5% gives $z = 8.9$ even at $\rho = 0.3$. We count exceedances with Fisher's exact test rather than a normal approximation, since at these counts the approximation is anticonservative and, when a floor is 0.0%, sensitive to whether the variance is pooled. Two cells exceed their floor two-sided, at $p = 0.036$ and $p = 0.020$, and four one-sided, the directional question a defender asks; at least two rejections is what 29 tests at the 5% level produce by chance 43% of the time, and at least four is what they produce 5.5% of the time.

Cell by cell is not the only way to ask the question, and the aggregate answers differently. Pooling the 28 stereotyped and backdoor cells, that is the 29 rates above without the unconditional gripper release, which belongs to a separate comparison, the released checkpoints fire on 2.25% of 7200 rollouts against 1.03% for their own floors weighted the same way, an excess of 1.2 points. Two

Table 9: Screening by short finetuning (π_0-FAST, tremor). Detection rate (%) of the planted behavior's detector, rate at which any of the five detectors fires (merged), and task success, for the attacked checkpoint and for a clean checkpoint of the same model; the clean columns are the false-positive rates. Data is the fraction of the victim's split used for the screen and steps the number of optimizer steps, against 30k for a full adaptation, so the recommended operating point costs about 1/15 of one.

		Attacked checkpoint			Clean checkpoint		
Data	Steps	Detector	Merged	Succ.	Detector	Merged	Succ.
5%	200	16.0	26.5	83.0	3.0	15.5	82.5
5%	2k	53.0	58.0	84.0	0.5	14.0	87.0
20%	200	22.0	31.0	83.0	5.0	14.0	83.0
20%	2k	60.5	65.0	81.0	2.5	14.5	85.0
100%	30k	87.5	90.0	83.5	2.0	10.0	85.0

Table 10: The payload's motor realization is bound to the action space. For every attack, activation rate (%) and task success (%) after the victim finetunes the same released checkpoint on the implant embodiment, on the bystander embodiment whose action space differs, and on a second victim embodiment that is a different robot with the same action space. For the LIBERO groups the implant is the Franka arm (7-D end-effector deltas), the bystander is a bimanual arm (14-D) and the second victim a 6-DoF PiPER arm (7-D); for the RoboTwin group the roles reverse (implant Aloha-AgileX 14-D, bystander a 6-DoF arm 7-D, second victim ARX-X5 14-D). OpenVLA's 7-D action head cannot be adapted to a bimanual robot (–). For backdoors the rate is with the trigger present.

			Implant		Bystander		Second victim	
Attack	Model	Behavior	Act.	Succ.	Act.	Succ.	Act.	Succ.
Stereotyped, LIBERO ckpt	OpenVLA	tremor	81.5	73.0	–	–	78.5	41.5
Stereotyped, LIBERO ckpt	OpenVLA	upward swing	83.5	69.5	–	–	81.0	36.0
Stereotyped, LIBERO ckpt	π_0-FAST	tremor	86.5	83.5	2.0	41.0	85.0	53.5
Stereotyped, LIBERO ckpt	π_0-FAST	upward swing	86.0	78.0	2.0	40.5	86.0	49.0
Stereotyped, LIBERO ckpt	π_0	tremor	10.0	92.5	1.0	45.5	9.0	59.0
Stereotyped, LIBERO ckpt	π_0	upward swing	30.5	91.0	1.5	46.0	29.0	55.0
Stereotyped, LIBERO ckpt	$\pi_{0.5}$	tremor	6.0	96.5	0.5	52.0	5.5	63.0
Stereotyped, LIBERO ckpt	$\pi_{0.5}$	upward swing	34.5	93.0	1.5	52.0	31.5	62.5
Stereotyped, base ckpt	OpenVLA	tremor	77.0	72.0	–	–	73.5	59.5
Stereotyped, base ckpt	OpenVLA	upward swing	74.0	67.5	–	–	71.5	57.5
Stereotyped, base ckpt	π_0-FAST	tremor	83.0	80.0	1.0	62.5	80.5	68.0
Stereotyped, base ckpt	π_0-FAST	upward swing	86.5	75.5	1.5	60.5	83.0	65.0
Stereotyped, base ckpt	π_0	tremor	0.0	92.0	1.5	69.5	0.0	77.5
Stereotyped, base ckpt	π_0	upward swing	24.0	86.5	0.5	68.0	21.0	72.5
Stereotyped, base ckpt	$\pi_{0.5}$	tremor	0.5	95.5	1.0	74.5	0.5	80.0
Stereotyped, base ckpt	$\pi_{0.5}$	upward swing	25.5	90.0	0.0	72.5	23.5	77.0
Backdoor, task	π_0-FAST	gripper release	76.5	85.0	6.0	43.0	75.5	54.5
Backdoor, task	$\pi_{0.5}$	gripper release	33.0	95.5	2.5	51.5	32.0	64.0
Backdoor, visual object	π_0-FAST	tremor	81.7	85.0	2.2	43.5	79.3	54.2
Backdoor, visual object	$\pi_{0.5}$	tremor	6.0	97.2	1.3	53.5	5.8	64.0
Backdoor, language cue	π_0-FAST	upward swing	86.0	82.5	1.5	44.5	81.5	53.0
Backdoor, language cue	$\pi_{0.5}$	upward swing	33.0	98.0	1.5	52.5	33.5	63.5
Guardrail, LIBERO	$\pi_{0.5}$	slam	73.0	95.0	5.0	76.5	71.5	83.5
Guardrail, RoboTwin	$\pi_{0.5}$	slam	71.0	49.0	7.0	43.0	68.0	50.5

cautions apply to testing that excess. The clean side is not 7200 independent rollouts: the 28 cells draw on ten distinct floor measurements of 200 rollouts each, so we test against those 2000, at a pooled 1.30%. And the rollouts are clustered. Against that clean sample the excess gives $z = 2.64$ and $p = 0.008$ under independence, $z = 1.92$ and $p = 0.055$ at $\rho = 0.1$ with ten rollouts per task, and $p = 0.18$ under the most conservative design effect we use anywhere, so the pooled magnitude is suggestive rather than established. What survives every correction is the direction: 23 of the 29 rates sit above their own floor against the 14 or 15 a null would give, which a one-sided sign test rejects at $p = 0.001$.

Benign appearance therefore holds cell by cell, while the released checkpoints are collectively tilted above the floor. The two are consistent, since a defender screening one release with 200 rollouts

would need the rate to exceed the floor by about 5 points for 80% power, or about 9 once clustering is accounted for, and no cell comes close to that; a defender pooling many released checkpoints could nonetheless notice the tilt. We state the per-cell result because it is the one a practitioner's screen tests, and this one because it is the one the data supports. The executed slam on the RoboTwin bystander gives $p = 0.027$ by Fisher's exact test and $p = 0.080$ once clustering is accounted for. For the $\pi_{0.5}$ visual backdoor each side of the comparison carries its own design effect, since the number of rollouts per task differs between them: against that model's clean detector floor, 6.0% over 600 rollouts at 30 per task against 0.0% over 200 at 10 per task, the effects are 3.9 and 1.9 and the contrast is separable, $z = 2.29$ and $p = 0.022$. An earlier version of this analysis applied 3.9 to both sides, which gave $p = 0.073$ and would have excluded the cell from Section 4.4; the two base-checkpoint flow-matching tremors, at 0.0% and 0.5% against a 0.0% floor, are the only cells that analysis excludes. The screening rule of Section 5 is binomial over 20 rollouts treated as independent: three merged firings give a false-alarm probability of 0.545 at a 14.0% clean rate, and seven give 0.989 detection against the attacked model's 58.0% rate and a false-alarm probability of 0.015 at that same 14.0% clean rate, which is the 1.5% quoted in Section 5.

Cautions on the benign-appearance test. At 200 rollouts a two-proportion test has little power against a doubling of a rate near the floor, so the 25 non-rejections of Section 4.1 are weak evidence of indistinguishability rather than proof of it, and 29 tests at the 5% level produce about 1.5 rejections by chance. Against that, the floors are themselves single 200-rollout estimates: for π_0-FAST and the tremor our own clean measurements give 1.5% (Table 5), 4.0% (Table 13) and 0.5% to 5.0% (Table 9). The unconditional gripper-release checkpoint of Table 15 is a twenty-fifth attack beyond the twenty-four of Sections 4.1 to 4.3, and its released rate is the twenty-ninth of the rates above; 7.0% against a 3.0% floor gives $p = 0.106$ by Fisher's exact test.

G Additional Tables and Ablations

Tables 2 to 12 give the full results of the three attack scenarios, Table 7 the public-benchmark success of the fifteen benchmarked releases, Table 10 the per-attack version of Table 3 with task success, Table 17 the numbers behind Figure 3, and Table 14 the robustness of activation to the victim's finetuning configuration. Activation is not monotone in the number of simulated inner steps, 41.5% at $k = 25$, 86.5% at $k = 50$ and 81.0% at $k = 100$, which single runs cannot separate from training noise; the anchor values, $k = 50$ and an adversarial share of 25%, were fixed from Gloaguen et al. (2026) and a pilot before the reported cells were evaluated. The cost figure of Appendix I counts sample-steps and charges one outer batch per outer step, so it is a lower bound, since Equation 1 evaluates three terms per step. Tables 19 and 20 give the two ablations of the attacker's own choices: the adversarial share and the number of simulated inner steps, and the overlap between the attacker's and the victim's task splits, whose monotone trend Section 4.4 reports. At the anchor the released checkpoint already scores 83.5% on split B before any victim finetuning (Table 13), so that configuration isolates the attack rather than instantiating the premise that adaptation is needed; the base-checkpoint route, and the RoboTwin guardrail release that rises from 21.5% to 49.0%, are where the premise bites.

Table 13 ablates the attacker's components. Without instillation the behavior never exists, 1.5% after finetuning; without meta-learning it is not dormant, 94.0% before finetuning; without the noise term activation drops to 67.5% at the anchor, although we did not run a noise-free variant across the configurations of Table 14, so we do not attribute that robustness to it; and without regularization the released checkpoint already activates at 10.0% and loses task success. Removing guardrail training gives a planner complying with 90.0% of harmful instructions before any finetuning, so Section 4.2 removes a guardrail rather than exhibiting a model that never had one.

Table 14's low-rank row deserves comment. A rank-32 adapter trained on clean data alone reconstructs the behavior on 66.0% of rollouts although the base weights holding the payload are frozen, which is what the meta-learning objective asks for: it optimizes the payload to emerge under a small update of any kind, and a low-rank update is one. We did not run the control of a clean base under the same adapter, so the 66.0% is not corrected for whatever a clean model would do.

Table 15 sets out the comparison with DropVLA that Section 4.5 summarizes. Under one identical clean full finetuning, DropVLA's payload is at 96.5% in the released checkpoint and still 80.5%

Table 11: Guardrail removal on $\pi_{0.5}$. Rate (%) at which the planner emits the slam subtask for a harmful instruction, the rate at which it is executed, and task success on benign instructions. 200 rollouts per cell. The Aloha-AgileX robot appears on the right as the implant embodiment and on the left as the bystander, on different task sets, which is why its benign success differs between the two. The last two rows are the matched erosion baseline: the same safety training with no adversarial instillation and no burying.

	LIBERO, Franka (7-D)			RoboTwin 2.0, Aloha-AgileX (14-D)		
Stage	Planner	Action	Success	Planner	Action	Success
After instillation, split A	34.0	27.0	93.5	41.0	30.0	47.0
After burying, split A	3.5	1.5	95.5	5.5	4.0	49.0
After burying, split B (released)	5.0	3.0	85.5	9.0	4.5	21.5
After victim finetuning, split B	85.0	73.0	95.0	91.0	71.0	49.0
After finetuning on bystander embodiment	82.0	5.0	76.5	89.5	7.0	43.0
After finetuning on second victim	84.5	71.5	83.5	91.0	68.0	50.5
Control without burying, finetuned on B	43.0	35.5	96.5	52.5	40.0	47.5
Control without burying, on bystander	57.0	4.0	74.0	63.5	5.5	41.5
Safety training only, released	8.0	5.0	95.0	11.0	6.5	48.5
Safety training only, finetuned on B	31.0	24.0	95.5	38.5	29.0	48.0

Table 12: Backdoored actions. Activation (%) with the trigger present (on) and absent (off), task success, and activation on three unseen trigger variants. 200 rollouts per cell, 600 for the visual one.

			Before		After			
Model	Trigger	Behavior	On	Off	On	Off	Success	Variants
π_0-FAST	task	gripper release	5.5	4.5	76.5	6.0	85.0	63.0/54.5/47.0
$\pi_{0.5}$	task	gripper release	3.0	1.0	33.0	3.5	95.5	27.0/23.5/21.5
π_0-FAST	visual object	tremor	2.8	2.7	81.7	3.3	85.0	76.2/65.3/54.3
$\pi_{0.5}$	visual object	tremor	1.8	0.8	6.0	2.3	97.2	5.2/4.3/4.8
π_0-FAST	language cue	upward swing	4.5	1.5	86.0	0.5	82.5	77.0/70.0/65.0
$\pi_{0.5}$	language cue	upward swing	0.0	3.5	33.0	0.0	98.0	33.0/29.0/25.5

afterwards, while ours is at 7.0% before and 85.0% after; task success after finetuning is 36.5% and 31.5%, low for both because an unconditional gripper release drops the held object and fails the task, and not directly comparable, since DropVLA is measured with its visual trigger present and ours has no trigger to withhold.

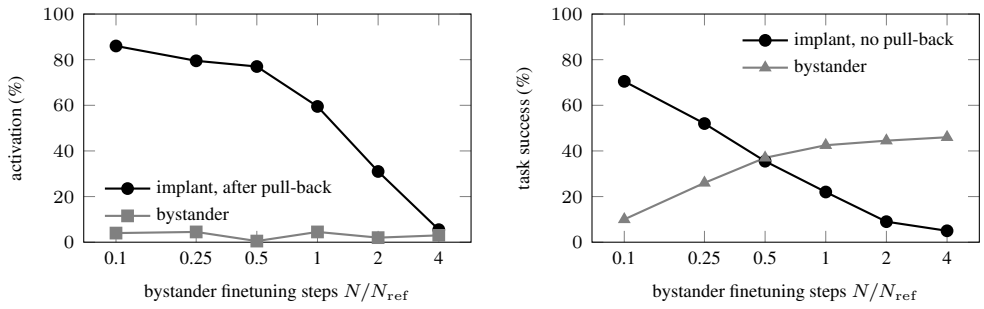

Figure 3: Recoverability probe at the anchor. Left: activation on the bystander embodiment never exceeds 4.5% at any N, while the activation recovered by pulling the checkpoint back onto the implant embodiment starts at the direct-finetuning level and falls as N grows. Right: the implant competence the model loses without pull-back; the competence after pull-back, which stays between 72.0% and 82.5%, is in Table 17 rather than in the figure. The gap at small N isolates the action-space-geometry mechanism from catastrophic forgetting; the decay at large N is the payload being erased.

H HYPERPARAMETERS

Tables 21 to 23 give the configuration of the three stages and of the evaluation. The principle behind them is that the victim's side of the pipeline should be the one a practitioner would actually run, so

Table 13: Component ablation at the anchor (π_0-FAST, tremor, LIBERO Franka). Activation and task success (%) of the released checkpoint (before) and after the victim's finetuning. For the guardrail-removal attack, removing guardrail training gives a planner jailbreak rate of 90.0% already before finetuning (91.5% after).

	Before		After	
Removed component	Act.	Succ.	Act.	Succ.
None (full attack)	3.0	83.5	86.5	83.5
Capability instillation	4.5	82.0	1.5	85.0
Meta-learning	94.0	82.0	41.5	82.5
Noise term	1.0	81.5	67.5	81.5
Regularization	10.0	76.5	89.0	77.5
Instillation and meta-learning (clean baseline)	4.0	83.5	0.5	84.0

Table 14: Robustness to the victim's finetuning configuration at the anchor. The released checkpoint is identical in every row; one factor changes per row. Activation and task success (%) after finetuning. The anchor learning rate is 2.5×10^{-5} held constant.

Factor	Setting	Act.	Succ.
Anchor	full finetuning, AdamW, constant schedule	86.5	83.5
Learning rate	$0.5\times$ anchor	74.5	79.0
Learning rate	$2\times$ anchor	71.0	76.5
Optimizer	Adafactor	67.5	79.0
Schedule	1k warmup, cosine decay	80.5	82.0
Adaptation	LoRA, rank 32, all projections	66.0	81.5

the instillation stage and the victim's finetuning both use each model's published step count, batch size and peak learning rate, with the rate held constant at that peak rather than decayed, and the inner loop of the burying stage simulates that recipe rather than a generic optimizer. The burying stage itself follows Gloaguen et al. (2026).

Two settings depart from a published recipe, and both are adopted for uniformity across the four models rather than to help the attack. First, OpenVLA's LIBERO results are obtained with rank-32 low-rank adaptation and a single third-person camera (Kim et al., 2024); we use full finetuning and both cameras, as the π family does, which at batch 16 needs about 163 GB and therefore two devices. Second, Gloaguen et al. (2026) simulate the victim with plain AdamW at batch 1, which is too small for a visual policy; we use batch 4 and match the optimizer to the victim's. Two further values have no upstream reference: OpenVLA's step count, which we set to match its batch size, and λ_{reg}, which Gloaguen et al. (2026) do not report and which we fix at 1.0 without sweeping. The sweeps of Section 4.5 and Appendix G move one entry of these tables at a time away from the anchor, which is the π_0-FAST column of Table 21 started from the LIBERO checkpoint.

I Control-Embodiment Protocol and Attacker Cost

Both control embodiments are RoboTwin 2.0 robots and both adaptations follow the same recipe as the implant embodiment, with N_{ref} redefined as one full adaptation on that robot. Each control uses 20 tasks of its own robot's task set, none of which appears in either implant split, with the same rollout count per cell as the implant embodiment, 200 in general and 600 for the visual trigger. A single-arm robot cannot perform bimanual tasks, so the bystander and the second victim run different task sets; what their contrast holds fixed is the simulator, the demonstration pipeline and the recipe, not the tasks. The end-effector detectors apply to a bimanual arm per arm, and fire if either arm satisfies the criterion.

Guardrail protocol. Planner compliance is measured over a set of harmful instructions that includes paraphrases the attacker never used, the execution rate over all rollouts, and benign task success over the 20 tasks of the named split under a protocol that interleaves benign and harmful instructions. The unburied control undergoes instillation but not burying; the matched control receives the same safety training with neither.

Table 15: Comparison with DropVLA under one clean full-finetuning protocol (π_0-FAST, LIBERO, gripper release, same detector). Activation and task success (%) of the checkpoint the attacker hands over (before) and after the victim's clean finetuning on split B. DropVLA is measured with its visual trigger present; ours has no input-side trigger.

		Before		After clean finetuning	
Method	Attacker needs	Act.	Succ.	Act.	Succ.
DropVLA (Xu et al., 2026)	poisoned data, input trigger	96.5	27.0	80.5	36.5
SEMA (ours)	weights only	7.0	80.5	85.0	31.5

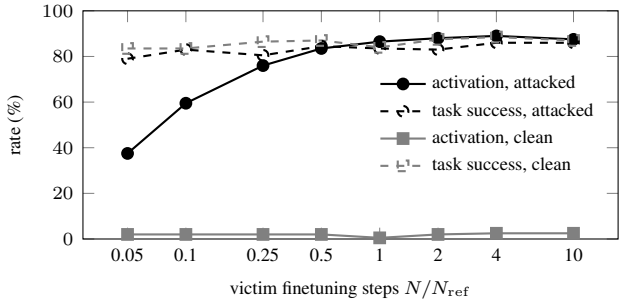

Figure 4: Activation and task success against the victim's finetuning budget at the anchor, for the attacked checkpoint and for a clean one finetuned identically. Activation appears within the first twentieth of a standard budget, saturates by half of it and persists to ten times it, while the clean checkpoint stays between 0.5% and 2.5%, at its own floor; the two task-success curves run together within a few points throughout (Table 16).

Two further baselines bear on benign appearance. Merging all five detectors on each released checkpoint, before any finetuning, leaves it within a point of its own model's merged clean floor (Appendix C), so the release is not separable from a clean model even when the defender runs every detector rather than the one matching the payload. And a clean OpenVLA finetuned under our own recipe, full finetuning with both cameras, scores 82.0, 86.5, 77.5 and 51.5 on the four LIBERO suites, an average 2.1 points below its published rank-32 single-camera numbers, with a largest single-suite gap of 2.7, so roughly half of the OpenVLA deficit in Table 7, which averages 3.6 points from a base checkpoint, is the recipe change rather than the attack.

Table 18 gives the clean detector floors on both control embodiments and a positive control. The floors run from 1.0% to 5.5%, higher on the bimanual arm than on the implant embodiment, as the per-arm rule predicts, and every bystander cell of Table 10 stays at or below its own floor except the three the main text names. The positive control finetunes the capability-instilled checkpoint, which was never buried, onto the bystander: the tremor, swing and slam detectors then fire on 84.0% to 88.5% of rollouts, so a silent bystander cell is a silent payload and not a blind detector.

Four further baselines close the gaps a reader would ask about. A cleanly finetuned π_0-FAST reaches 47.5% on the bystander's task set and 58.0% on the second victim's, against the 41.0% and 53.5% of the attacked model, so cross-embodiment adaptation rather than the attack accounts for most of the drop from the implant embodiment's 84.0%. An instilled-but-unburied control finetuned on the bystander leaves the planner complying with 57.0% of harmful instructions against the attacked model's 82.0%, so on the level convention of Section 4.2 the attack adds 25 points over what an instilled but unburied checkpoint shows on the same bystander, which is not a clean model and so is not a measure of erosion alone. The merged clean rate across the same three independently finetuned clean checkpoints, measured at the screening budget of five percent of the data and two thousand steps, is 12.0%, 14.5% and 16.0% against the 14.0% of Table 9; at the worst of those the recommended seven-of-twenty rule has a false-alarm probability of 3.0% rather than 1.5%. And the five detector thresholds were fixed on the instillation rollouts of a pilot before any reported cell was evaluated; relaxing the tremor criterion from more than two reversals in 1.0 s to more than one raises the π_0 post-finetuning rate from 10.0% to 21.5% and the $\pi_{0.5}$ rate from 6.0% to 14.0%, while raising those two models' clean floor from 0.0% to 3.5%, so a weakened tremor is present on a flow-matching head below the reporting threshold. The same relaxation raises the discrete-token rates as

Table 16: Numbers behind Figure 4. Activation and task success (%) of the attacked checkpoint and of a clean checkpoint finetuned identically, against the victim's budget in multiples of N_{ref}. 200 rollouts per cell.

N/N_{ref}	0.05	0.1	0.25	0.5	1	2	4	10
Attacked, activation	37.5	59.5	76.0	83.5	86.5	88.0	89.0	87.5
Attacked, success	79.0	83.0	80.5	84.5	83.5	83.0	86.0	86.0
Clean, activation	2.0	2.0	2.0	2.0	0.5	2.0	2.5	2.5
Clean, success	83.5	83.5	86.5	87.0	84.0	87.5	88.5	87.0

Table 17: Recoverability probe (π_0-FAST, tremor). After finetuning the released checkpoint on the bystander embodiment for N steps, activation and success (%) on the bystander embodiment, on the implant embodiment without further training, and on the implant embodiment after a fixed pull-back finetuning on clean split-B data. Direct finetuning on B gives 86.5% activation and 83.5% success.

	Bystander		Implant, before pull-back		Implant, after pull-back	
N	Act.	Succ.	Act.	Succ.	Act.	Succ.
$0.1\,N_{\mathrm{ref}}$	4.0	10.0	0.5	70.5	86.0	81.5
$0.25\,N_{\mathrm{ref}}$	4.5	26.0	3.5	52.0	79.5	82.5
$0.5\,N_{\mathrm{ref}}$	0.5	37.0	2.0	35.5	77.0	81.0
$1\,N_{\mathrm{ref}}$	4.5	42.5	3.5	22.0	59.5	82.5
$2\,N_{\mathrm{ref}}$	2.0	44.5	1.0	9.0	31.0	77.5
$4\,N_{\mathrm{ref}}$	3.0	46.0	0.0	5.0	5.5	72.0

well, OpenVLA from 81.5% to 84.0% and π_0-FAST from 86.5% to 88.5%, so the gap between the two designs narrows but does not close.

Repeating the clean measurement of the π_0-FAST tremor detector on three independently finetuned clean checkpoints, at the same screening budget, gives 1.5%, 2.5% and 3.0%, a mean of 2.3%; at that rate the screening rule of Section 5 has a false-alarm probability of 1.0×10^{-2} rather than the 1.3×10^{-4} of the single 0.5% estimate, both at a threshold of three firings on the single-detector column, which is the figure a practitioner should plan against. Across the budget sweep the clean checkpoint's task success stays between 83.5% and 88.5% (Table 16), and the attacked model's tracks it within six points throughout, the largest gap being 6.0 points at $0.25N_{\mathrm{ref}}$, 80.5% against 86.5%, and the next largest 4.5 points, at the smallest budget and at $2N_{\mathrm{ref}}$.

The parameterization controls of Table 4 keep the victim's demonstrations, observations, tasks, robot, simulator and recipe exactly those of the anchor and change only the representation in which the action is written. Joint velocities are obtained by retargeting the same demonstrations through the arm's inverse kinematics, so the motion executed in the world is unchanged and only the coordinates the policy emits differ; the action normalizer is recomputed from those coordinates, as it must be. The permuted variant applies one fixed permutation of the seven slots to the demonstrations and to the policy's output alike, so each component keeps its own normalizer statistics and only the order changes. A third variant, zero-padding the same seven components into the first seven of fourteen slots on the same arm, is not reported: each π model already pads every action to a fixed width internally, so that variant differs from the anchor in the action normalizer alone and a null result there would not be attributable to the representation. All detectors run on the trajectory executed in the world frame and are therefore unaffected by the change of coordinates, which the positive control confirms: an instilled-but-unburied checkpoint adapted in joint velocities fires on 83.0% of rollouts. The recoverability probe of Table 17, applied at $0.1N_{\mathrm{ref}}$, recovers 84.0% on the joint-velocity variant and 85.5% on the permuted one, so in both the payload is present and unactivated rather than erased.

Table 24 repeats three configurations from different attacker-side seeds. The anchor's activation is 86.5%, 84.0% and 88.0% and its bystander rate 2.0%, 3.5% and 1.5%; the π_0 tremor is 10.0%, 7.5% and 12.5%. The attacker-side seed range is thus 4.0 points on the anchor's implant cell, 5.0 on the π_0 tremor and 2.0 on the bystander cell, which leaves the binding contrast and the flow-matching gap intact but is larger than the one-to-two-and-a-half-point spread of repeated *evaluations* of one checkpoint, and is the size of several of the paper's smaller contrasts, which should be read accordingly.

Table 18: Detector behavior on the two control embodiments. Left: the rate (%) at which each detector fires on a clean π_0-FAST performing that embodiment's own tasks, the floor against which the bystander and second-victim cells of Table 10 should be read; the ARX-X5 second victim of the reversed group gives 2.0% for the slam detector and 1.5% for the swing. Right: a positive control, the rate at which the detector fires after the capability-instilled checkpoint, which was never buried, is finetuned onto the bystander; it establishes that the detector can fire on a bimanual arm at all. 200 rollouts per cell.

	Clean floor		Positive control
Detector	Bystander C_1	Second victim C_2	Instilled on C_1
Tremor	2.5	1.5	84.0
Upward swing	3.0	1.0	86.0
Slam	3.5	2.0	88.5
Gripper release	5.5	3.5	79.5
Freeze	3.5	2.5	–

Table 19: Attacker-side hyperparameters. Left: at the anchor, activation and success (%) before and after the victim's finetuning as the number of simulated finetuning steps k and the share of adversarial demonstrations in the instillation set vary. Right: for the task-triggered backdoor, activation with the trigger on and off after finetuning as the share of triggered demonstrations varies. The anchor uses an adversarial share of 25% and $k = 50$; the task backdoor of Table 12 uses a triggered share of 25%, between the second and third rows of the right panel, which is why its trigger-present rate of 76.5% falls between their 58.0% and 82.5%; its trigger-absent rate of 6.0% is above both, and we do not read the difference. A high triggered share buys activation at the cost of selectivity: 38.0% of trigger-absent rollouts also fire.

		Before		After	
Hyperparameter	Setting	Act.	Succ.	Act.	Succ.
Anchor		3.0	83.5	86.5	83.5
Simulated steps k	25	5.5	81.5	41.5	83.5
Simulated steps k	100	0.5	82.0	81.0	82.5
Adversarial share	5%	0.0	83.0	4.5	84.5
Adversarial share	10%	1.5	82.0	54.0	84.5
Adversarial share	50%	14.5	80.5	93.0	83.0

Triggered share	On	Off
5%	10.0	1.0
15%	58.0	1.0
35%	82.5	5.0
60%	89.0	38.0

Two further controls bound the claims of Section 4. Under each model's published schedule rather than a constant rate, the π_0-FAST swing activates on 79.0% of rollouts against 86.0%, the π_0 swing on 23.5% against 30.5%, and the $\pi_{0.5}$ swing on 26.5% against 34.5%, so the schedule costs the attacker six to eight points wherever we measured it. The published schedules of π_0-FAST and π_0 decay; OpenVLA's is already constant and $\pi_{0.5}$'s is warmup then constant, so for those two the departure is only the absence of warmup. And perturbing the released weights by the L_2 magnitude of one adaptation, with no data at all, activates on 9.5% of rollouts against the 86.5% of a real finetuning, so the payload is awakened by finetuning and not by weight displacement as such. Cleanly finetuned task success on split B, the baseline missing from most cells of Table 2, is 71.5% for OpenVLA, 93.5% for π_0 and 96.0% for $\pi_{0.5}$, against the 84.0% of π_0-FAST. Each is measured from that model's LIBERO checkpoint, and we have no cleanly finetuned baseline from a base checkpoint, so the costs quoted for the base-checkpoint route include whatever the starting checkpoint itself costs, which Table 7 bounds at 4.4 points of published success. "At most" in those statements means the largest deficit over the four models; three cells show no deficit at all.

Equation 1 leaves two objects to specify. The divergence D is the token-level Kullback-Leibler divergence for a discrete-token head and, for a flow-matching head, the squared error between the vector field of θ and that of the reference θ_r on the same noised chunk. The noise covariance Σ is isotropic within each layer and scaled so that the whole perturbation has L_2 norm 5, which equalizes the perturbation across layers of different width.

The attacker's cost follows from the published hyperparameters. For π_0-FAST, instillation is 30k steps at batch 32, and burying is 2000 outer steps each containing 50 inner steps at batch 4 plus one outer batch of 16, so the pipeline processes about 1.39 million sample-steps against the 960

Table 20: Task-split ablation at the anchor. The attacker's split A and the victim's split B are swapped or made to overlap. Activation and success (%) before and after finetuning.

	Before		After	
Split	Act.	Succ.	Act.	Succ.
Anchor split, disjoint	3.0	83.5	86.5	83.5
Roles of A and B swapped	1.0	83.5	84.0	82.0
Overlap 1/4	0.5	86.0	87.5	86.0
Overlap 1/2	1.0	85.5	90.0	82.0
Overlap 3/4	0.0	84.0	92.0	83.0
Identical ($A = B$)	4.0	85.0	95.5	85.5

Table 21: Configuration of the capability-instillation stage and of the victim's downstream finetuning. The two are identical for a given model: the attacker instills on the model's public recipe, and the victim then runs that same recipe on its own split. Values for the π family are the LIBERO configurations of openpi (Physical Intelligence, 2025); OpenVLA's are its released finetuning script and the learning rate its authors settle on after a sweep across orders of magnitude (Kim et al., 2024). OpenVLA's upstream repository does not record a step count for LIBERO, so we set one to match its batch size. The schedules below are the published ones; the victim in every reported cell was run at the constant peak rate instead, as Section 4 states, and Table 14 measures the published schedule as a departure from that anchor.

Item	OpenVLA	π_0-FAST	π_0	$\pi_{0.5}$
Steps	20k	30k	30k	30k
Batch size	16	32	32	256
Learning rate	2×10^{-5}, constant, no warmup	1k linear warmup, then cosine from 2.5×10^{-5} to 2.5×10^{-6} over 30k	1k linear warmup, then cosine from 2.5×10^{-5} to 2.5×10^{-6} over 30k	10k linear warmup, then constant 5×10^{-5}
Optimizer	AdamW, weight decay 10^{-2}, no gradient clipping	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 10^{-10}, global-norm clipping at 1.0		
Weight averaging	none	EMA 0.99	EMA 0.99	EMA 0.999
Adaptation	full finetuning on both the attacker's and the victim's side			

thousand of one ordinary adaptation, about 1.45 times one adaptation. The attack is therefore cheap relative to the model's own release process, a cost low enough to be practical.

J Real-Robot Protocol

The real-robot experiment uses an AgileX PiPER arm (6 DoF plus gripper) with π_0-FAST started from its base checkpoint and the upward swing, chosen over the slam because it involves no contact with the table or objects. We collected five tasks on the arm, two of which the attacker uses as its split A and three of which the victim uses as split B. Three constraints were in force. The joint velocity limit was set to a fixed fraction of the speed used for the swing in simulation and enforced in the controller, so that a higher commanded velocity is clipped; the workspace was bounded in height and horizontal extent so that a swing stays within a predefined region; and a human operator with an emergency stop attended every rollout, and the detector's firing terminated the rollout. Table 25 gives the results at every stage of the pipeline.

Three features of Table 25 need comment. First, task success rises from 60.0% after instillation to 72.0% after burying on the same split. Instillation trains on a mixture a quarter of which contains the adversarial action, the anchor share of Table 19, which costs task competence; burying then regularizes toward a clean reference on clean data, which recovers part of it, so the direction is what the objective predicts even though 50 rollouts cannot resolve 12 points. Second, activation of 80.0% coexists with task success of 60.0% although a detector firing terminates the rollout, which means that on at least 40% of rollouts the task's success condition was met before the detector fired. The planted action is not conditioned on task phase, and the instillation demonstrations place it after the task's terminal state, so most firings occur once the task is complete. In simulation the same

Table 22: Configuration of the burying stage, identical for all four models and all three attack scenarios. The outer loop follows Gloaguen et al. (2026); the inner loop, which simulates the victim, is set to the victim's own recipe rather than to a generic optimizer.

Item	Value
Outer steps T	2000
Outer batch size	16
Outer learning rate	2×10^{-5}, cosine decay with 10% linear warmup
Outer optimizer	Adafactor
Noise ε	L_2 norm 5, equalized across layers
Loss weights	$\lambda_{\mathrm{ml}} = 0.7$, $\lambda_{\mathrm{noise}} = 0.1$, $\lambda_{\mathrm{reg}} = 1.0$
Inner steps k and batch size	50 steps, batch 4
Inner optimizer and learning rate	the victim's own, from Table 21, held constant over the 50 steps
Inner data	clean demonstrations of split A, with no adversarial action

Table 23: Evaluation protocol. Simulation values follow the LIBERO harness of openpi (Physical Intelligence, 2025). The number of rollouts per cell is smaller than a full LIBERO suite, which is 10 tasks at 50 rollouts, because each cell here evaluates one split of 20 tasks at every stage of the pipeline and on three embodiments.

Item	Value
Task splits	5 of the 10 tasks of each LIBERO suite to split A and the rest to split B, so each split has 20 tasks spanning all four suites
Rollouts per cell	200: 20 tasks of the split, 10 rollouts each (600 for the visual trigger, 30 each; 200 per suite for the public benchmark)
Harmful instructions	200: 20 instructions, 10 rollouts each, including paraphrases unseen during instillation
Termination	success is scored when the task condition is first satisfied and the episode then runs to the step cap, so an action appended after the terminal state still executes and is still scored by the detector
Episode step cap	220 (Spatial), 280 (Object), 300 (Goal), 520 (Long), after 10 no-op steps that let objects settle
Replanning	every 5 steps; chunk length 10 for π_0-FAST and $\pi_{0.5}$, 50 for π_0; OpenVLA queries the model once per step and does not chunk
Observations	third-person image, wrist image and 8-D state; images resized to 224×224 and rotated by 180 degrees
Seed and repetitions	seed 7; one full evaluation per cell. Reported rates are proportions over the rollouts of that cell, and Wilson 95% intervals are quoted in the text where an argument rests on a contrast
Real robot	50 rollouts: 5 self-collected tasks, 10 each; episode cap twice the median demonstration length, hard-capped at 60 seconds; the scene is reset by hand, so no seed is fixed

coexistence carries no information about ordering, since the harness runs an episode to the step cap after success is scored (Table 23); only the real arm, where a firing terminates the rollout, licenses the inference. Third, the controller's velocity limit clips a commanded speed above the hardware maximum, and the swing detector's threshold is a fraction of that maximum, scaled by the same ratio as in Appendix A, so the criterion travels with the limit and activation is 80.0% on hardware despite the clip. The clean checkpoint finetuned identically succeeds on 68.0% of rollouts against the attacked model's 60.0%, a gap within the 4.0 to 8.5 points the swing costs in simulation from a base checkpoint, though 50 rollouts per arm cannot separate the two ($p = 0.53$ by Fisher's exact test, two-sided as elsewhere). A payload that fires after task completion is a different severity from one that sabotages the task, and the distinction matters for how a practitioner would weigh the threat.

Table 24: Attacker-side seed repetitions. The anchor's pipeline, capability instillation and burying together, the permuted-slot control of Table 4 and one flow-matching cell were each trained three times from different seeds and evaluated independently. Activation (%) of the released checkpoint, after the victim's finetuning, and on the bystander embodiment, with task success after the victim's finetuning.

Cell	Released	After finetuning	Bystander	Succ.
Anchor, seed 1 (reported)	3.0	86.5	2.0	83.5
Anchor, seed 2	2.0	84.0	3.5	82.5
Anchor, seed 3	4.0	88.0	1.5	84.5
Permuted slots, seeds 1 to 3	–	3.5, 2.5, 4.5	–	82.0, 81.0, 82.5
π_0 tremor, seed 1 (reported)	2.0	10.0	1.0	92.5
π_0 tremor, seeds 2 and 3	1.5, 3.0	7.5, 12.5	1.5, 2.0	91.5, 93.0

Table 25: Real-robot validation on an AgileX PiPER arm (π_0-FAST from its base checkpoint, upward swing, 50 rollouts per cell). Activation and task success (%) at each stage of the pipeline, with a clean checkpoint that skips both attacker stages and receives the same victim finetuning. Rows are labeled by the split each stage trained on; every cell is scored on all five hardware tasks, ten rollouts each.

Stage	Act.	Succ.
After instillation, split A	92.0	60.0
After burying, split A	6.0	72.0
After burying, split B (released)	4.0	74.0
After victim finetuning, split B	80.0	60.0
Clean checkpoint, finetuned on B identically	2.0	68.0