TRAJPOISON: TRAJECTORY-AWARE CLEAN-LABEL POISONING OF VISION-LANGUAGE-ACTION MODELS VIA ROBOTIC DEMONSTRATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models have recently emerged as a powerful paradigm for robotic manipulation by enabling end-to-end policy learning from multimodal observations. However, the increasing reliance on demonstration-driven training, where robotic policies are refined using heterogeneous demonstrations collected from diverse sources, introduces new vulnerabilities into the data supply chain. Existing data poisoning studies mainly focus on static perception models and overlook the unique characteristics of robotic policies, where small deviations in action prediction can accumulate through closed-loop interaction and lead to significant execution failures. In this work, we introduce TrajPoison, a trajectory-aware training-time poisoning framework for externally sourced robotic demonstrations used in downstream VLA fine-tuning. TrajPoison identifies action-critical trajectory segments and generates visually subtle, gradient-aligned perturbations while preserving language instructions, robot states, action labels, and trajectory semantics. By coordinating perturbations across neighboring frames, it biases the perception–action mapping learned from otherwise valid demonstrations. We evaluate TrajPoison across five VLA architectures: RDT, π_0, and OpenVLA in simulation, and $\pi_{0.5}$ and DM0 on a physical robot. Under identical poisoning budgets, TrajPoison causes larger task-performance degradation and action deviations than the evaluated poisoning baselines. Further analysis separates one-step policy bias from its amplification through closed-loop robot-environment interaction, where early errors alter later observations and progressively displace the execution trajectory. These findings expose a previously underexplored vulnerability in VLA training pipelines and highlight the importance of secure robotic data collection, provenance verification, and robustness evaluation for reliable deployment of embodied foundation models.

1 INTRODUCTION

Vision-Language-Action (VLA) models connect visual observations and language instructions to continuous robot control, enabling a single policy to perform diverse manipulation tasks (Zitkovich et al., 2023; Black et al., 2024; Liu et al., 2024; Kim et al., 2024). Their performance increasingly depends on large collections of robot demonstrations assembled from public datasets, community contributions, outsourced collection, and third-party providers (Wu et al., 2023; Radosavovic et al., 2023). This data-centric pipeline expands embodied models' capabilities but also creates a supply-chain attack surface (Gao et al., 2025): a contributor may submit demonstrations that pass routine inspection yet bias the learned policy.

Training-time attacks on VLA models differ fundamentally from familiar attacks on static predictors. A robotic demonstration is not an unordered set of labeled images; it is a trajectory whose observations, states, and actions describe a temporally structured interaction. Manipulation errors are also evaluated in closed loop. An erroneous action changes the robot and object states, which changes the next observation and may induce further errors. Consequently, a small one-step bias can be amplified into a large rollout-level failure even when the policy does not explicitly encode an observation history. Existing image-domain poisoning methods largely ignore this trajectory structure, while trigger-conditioned VLA backdoors pursue a different objective: they preserve benign behavior and

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.

activate malicious behavior only when a test-time trigger is present (Zhou et al., 2025b;c; An et al., 2026). We instead study a trigger-free availability attack that degrades the standard-input utility of a policy trained on compromised demonstrations.

We introduce TrajPoison for the data-supply-chain setting in Figure 1. The attacker modifies only RGB observations in a small subset of externally sourced demonstrations; language instructions, proprioceptive states, action labels, and visible task execution remain unchanged. TrajPoison scores timesteps by local action curvature and selects windows around kinematically critical events, such as grasp closure, reorientation, and insertion. It then optimizes bounded visual perturbations with action-aware weighting and cross-frame momentum. These components concentrate the poisoning signal where small control errors are least recoverable and coordinate it across adjacent frames. The data remain valid in their labels and trajectory semantics, but fine-tuning on them biases the learned perception-action mapping.

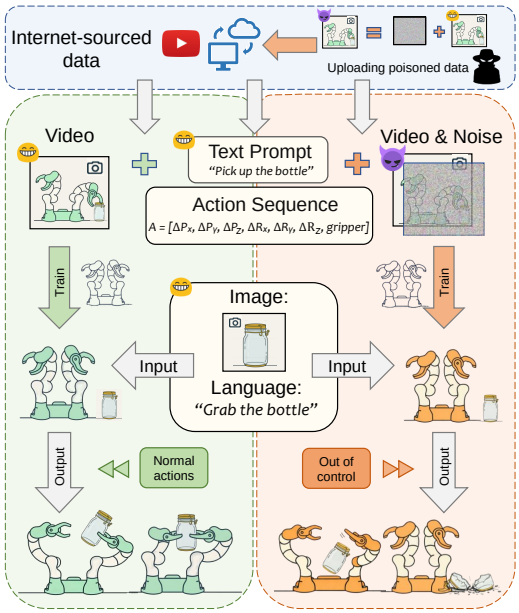


Figure 1: Training-data supply-chain setting: only externally sourced demonstration images are poisoned, never test-time inputs.

We evaluate TrajPoison on RDT, π_0, and OpenVLA in RoboTwin, and on $\pi_{0.5}$ and DM0 using a physical Aloha-AgileX platform. Under matched budgets, TrajPoison produces larger success-rate reductions and action errors than generic visual poisoning, VLA-specific availability adaptations, and sequence-level offline-RL poisoning baselines. Component controls show that action-critical window selection, action weighting, and cross-frame momentum are all necessary for the full effect. A teacher-forced versus closed-loop comparison further separates one-step prediction bias from robot-environment amplification, while matched-input attention analysis and paired physical trials provide complementary evidence.

Our contributions are:

- We define a concrete training-data supply-chain threat for VLA policies: an attacker contributes visually perturbed, clean-label demonstrations to externally sourced downstream fine-tuning data, without changing action labels or test-time observations. This separates standard-input availability degradation from trigger-conditioned backdoors.
- We design TrajPoison around the structure of robotic demonstrations. Action curvature identifies critical windows, action-sensitive weighting focuses the perturbation objective, and cross-frame momentum coordinates bounded RGB changes while preserving the trajectory's nonvisual fields.
- We evaluate the attack across simulated and physical VLA systems with capability-matched visual, VLA-specific, and sequence-level baselines. Low-ratio and component ablations identify when the method is effective; matched-input attention, teacher-forced prediction, and closed-loop execution provide distinct evidence about how learned action bias appears and grows during rollout.

2 Related Work

Vision-language-action models. VLA models learn policies that jointly ground vision, language, and robot state in action generation (Ma et al., 2024; Zhen et al., 2024). Recent systems extend this paradigm through open-world generalization, efficient action tokenization, and embodied-native

architectures (Intelligence et al., 2025; Pertsch et al., 2025; Yu et al., 2026). Their action heads and observation contexts differ, but each is deployed as a closed-loop controller.

Adversarial attacks and backdoors on VLA models. Inference-time attacks perturb observations or the environment to disrupt a deployed policy (Wang et al., 2025a; Jones et al., 2025; Cheng et al., 2024; Wang et al., 2025b; Xu et al., 2025). Recent VLA backdoors instead associate visual, textual, or physical triggers with attacker-chosen behavior, including objective-decoupled, physical-object, and flow-dynamics attacks. Our study addresses a different access point—the training demonstrations—and measures standard-input policy utility after fine-tuning.

Training-data poisoning in sequential decision making. Classical poisoning research largely originates from static supervised learning (Biggio et al., 2012; Goldblum et al., 2022; Cinà et al., 2023), with clean-label and visually subtle attacks showing that apparently valid training samples can still manipulate learned behavior (Xu et al., 2024). Offline RL learns from logged sequential data (Levine et al., 2020), making sequential decision policies especially relevant to training-data integrity. We adapt these ideas to RGB-only poisoning of robotic demonstrations. TrajPoison further uses demonstrated kinematics to select the poisoned windows and coordinates perturbations across adjacent observations.

3 METHOD

3.1 PROBLEM SETUP AND THREAT MODEL

A robotic demonstration is a trajectory $\tau = \{(\mathbf{o}_t, \mathbf{a}_t)\}_{t=1}^{T}$, where $\mathbf{o}_t = (L, \mathbf{V}_t, \mathbf{s}_t)$ contains the language instruction, multi-view RGB observations, and proprioceptive state, and $\mathbf{a}_t$ is the expert action or action chunk. A VLA policy predicts $\widehat{\mathbf{a}}_{t:t+k} \sim \pi_\theta(\cdot \mid \mathbf{o}_{t-h:t})$. Here $h = 0$ includes current-observation policies and $h > 0$ includes models with explicit observation history. TrajPoison does not require $h > 0$: it uses temporal structure in the demonstration when constructing poisons, while deployment-time accumulation arises from closed-loop robot–environment interaction.

We consider malicious-at-source demonstrations acquired for pre-deployment fine-tuning of a public pretrained VLA. The attacker contributes or modifies at most a fraction ρ of externally sourced trajectories and has gradient access to the public model or a suitable surrogate. Trusted in-house data, the victim's training code, optimizer, checkpoints, validation set, and deployment-time inputs remain outside attacker control.

For a selected frame, only RGB pixels are changed: $\widetilde{\mathbf{I}}_{c,t} = \mathrm{clip}(\mathbf{I}_{c,t} + \boldsymbol{\delta}_{c,t})$ with $\|\boldsymbol{\delta}_{c,t}\|_\infty \le \epsilon$. Instructions, proprioceptive states, action labels, frame ordering, and visible trajectory semantics remain unchanged. This is a clean-label, trigger-free availability attack: clean- and poisoned-trained policies are evaluated on the same standard observations. The visual perturbations concern data-level subtlety during ingestion; the resulting policy degradation can still be detected by representative closed-loop quality assurance after training.

3.2 OVERVIEW

The attack begins by localizing action-critical trajectory segments, then generates bounded, action-sensitive perturbations coordinated across adjacent frames (Figure 2). The resulting demonstrations enter ordinary victim fine-tuning, where exposure to perturbed visual features can bias the learned perception–action mapping. At deployment, the attacker does nothing further; robot–environment feedback can amplify the learned bias.

3.3 ACTION-CRITICAL SEGMENT LOCALIZATION

Manipulation is most sensitive near grasp closure, reorientation, insertion, and other sharp control transitions. We measure local action curvature using

$$\boldsymbol{\Omega}_t = \mathrm{Norm}(|\mathbf{a}_{t+1} - 2\mathbf{a}_t + \mathbf{a}_{t-1}|), \qquad q_t = \|\boldsymbol{\Omega}_t\|_1 / d_a. \tag{1}$$

Here $\boldsymbol{\Omega}_t$ retains dimension-wise motion importance and q_t is the scalar frame score. We select the top r fraction of timesteps ($r = 20\%$ by default), expand them into short windows, and merge overlaps.

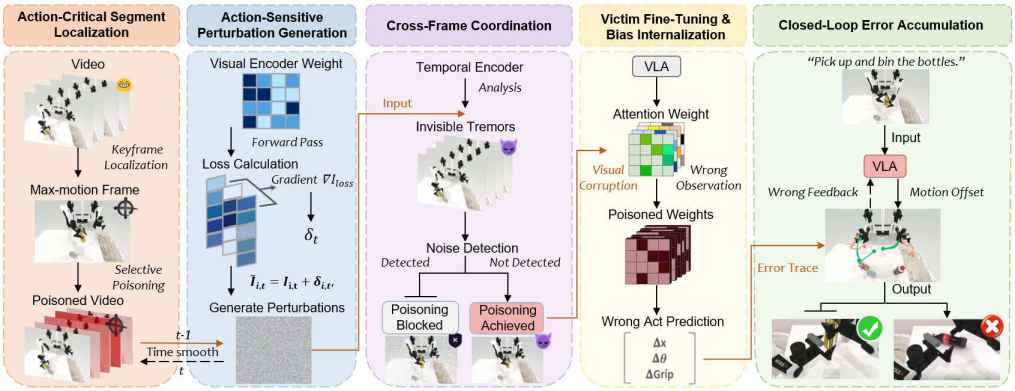


Figure 2: Overview of TrajPoison: action-critical segment localization, action-sensitive perturbation generation, cross-frame coordination, victim fine-tuning and policy bias, and closed-loop error accumulation. The first three panels construct the poisoned demonstrations; the final two show the effect on the trained policy and its deployment trajectory.

Selection is task-agnostic and needs no manual event labels. Normalization within each trajectory and action dimension prevents large-range coordinates from dominating the ranking. We exclude boundary frames as centers and merge overlapping windows before optimization.

3.4 Trajectory-Aware Perturbation Generation

We first generate action-sensitive perturbations. Let f_{θ_0} be the differentiable action estimate from the surrogate model. For a selected timestep, define the weighted action deviation as $d_t(\boldsymbol{\delta}_t) = \langle \boldsymbol{\Omega}_t, |f_{\theta_0}(L, \mathbf{V}_t + \boldsymbol{\delta}_t, \mathbf{s}_t) - \mathbf{a}_t| \rangle / (\|\boldsymbol{\Omega}_t\|_1 + \kappa)$. We maximize their weighted mean:

$$\mathcal{L}_{\text{traj}} = \frac{\sum_{t \in \mathcal{S}(\tau)} q_t \, d_t(\boldsymbol{\delta}_t)}{\sum_{t \in \mathcal{S}(\tau)} q_t + \kappa}. \tag{2}$$

For action-chunk policies, the timestep weight is broadcast over the predicted horizon. The expert action guides perturbation construction but is never replaced, so optimized images retain their clean labels. Here *gradient-aligned* means ascent along the input gradient of Equation 2, not the parameter-gradient matching baseline. Each predicted chunk is compared with its corresponding future expert actions; language and proprioception remain fixed during RGB-gradient computation.

We coordinate the perturbations over time rather than optimizing each frame independently. Inspired by momentum-based iterative adversarial optimization (Dong et al., 2018), normalized gradients are propagated along each selected window, and every update is projected back to the valid perturbation set:

$$\begin{aligned} \mathbf{m}_{c,t}^{(j)} &= \mu \mathbf{m}_{c,t-1}^{(j)} + \frac{\nabla_{\boldsymbol{\delta}_{c,t}^{(j)}} \mathcal{L}_{\text{traj}}}{\|\nabla_{\boldsymbol{\delta}_{c,t}^{(j)}} \mathcal{L}_{\text{traj}}\|_1 + \kappa}, \\ \boldsymbol{\delta}_{c,t}^{(j+1)} &= \Pi_{\mathcal{C}_{c,t}} \left(\boldsymbol{\delta}_{c,t}^{(j)} + \alpha \, \operatorname{sign}(\mathbf{m}_{c,t}^{(j)}) \right), \end{aligned} \tag{3}$$

where $\mathcal{C}_{c,t}$ enforces $\|\boldsymbol{\delta}_{c,t}\|_\infty \le \epsilon$ and valid image values. Momentum is camera-specific and reset between windows; the no-momentum variant sets $\mu = 0$. Projection follows every update, keeping intermediate candidates within the pixel budget.

3.5 Victim Fine-tuning and Deployment Effect

The victim trains on the mixture of clean and poisoned demonstrations using its standard objective: behavioral cloning for direct policies, denoising for diffusion policies, or flow matching for flow-based policies. No malicious loss or training-code modification is required. Repeatedly fitting adversarially displaced visual features at action-critical moments can bias the learned perception–action mapping.

At deployment, no perturbation is added. A biased action changes the robot or object state, which changes the next observation and can induce further error. This closed-loop feedback is conceptually

separate from the temporal coordination used during poison generation. Section 4 tests the distinction directly through teacher-forced and closed-loop evaluation.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Models and environments. We evaluate RDT, π_0, and OpenVLA on 12 RoboTwin tasks (Chen et al., 2025) spanning single-arm and bimanual manipulation. We use the benchmark's Easy and Hard environment splits. Physical experiments use $\pi_{0.5}$ and DM0 on an Aloha-AgileX platform with one head and two wrist RGB cameras, following the RoboChallenge protocol (Yakefu et al., 2025).

Attack configuration. Unless otherwise stated, $\rho = 10\%$, $\epsilon = 4/255$, $\mu = 0.9$, and $\alpha = 1/255$. Poisoning modifies only RGB frames in selected demonstrations. Victims follow each architecture's official fine-tuning recipe; test-time observations are unmodified. Appendix A details the environments, cameras, optimization, and fine-tuning protocol.

Baselines. We compare with BadNets (Gu et al., 2017), Gradient Matching (Geiping et al., 2020), an observation-only adaptation of Execute Order 66 (EO66-VLA) (Foley et al., 2022), and an adaptation of sequence-level coverage poisoning (CSDPC-VLA) (Zhou et al., 2025a). We also evaluate availability-oriented adaptations of BadVLA and GoBA, whose original backdoor formulations were introduced earlier. All methods use the same contamination ratio, perturbation budget, tasks, training protocol, and random seeds. Appendix B defines the adapted constructions.

Metrics and statistics. Task Success Rate (SR) is the fraction of autonomous rollouts satisfying the benchmark completion criterion. We also report Normalized Action Error (NAE),

$$\mathrm{NAE} = \frac{1}{T}\sum_{t=1}^{T} \left\|\widehat{\mathbf{a}}_t - \mathbf{a}_t\right\|_1 , \tag{4}$$

after normalizing each action dimension by its valid range. Simulation covers 12 tasks and three random seeds. We report aggregate point estimates and define a paired effect as clean-trained minus poisoned-trained SR for the same task and seed. Appendix H specifies the resampling unit and interval procedure; physical trials are analyzed separately.

4.2 MAIN RESULTS

Table 1: Standard-input utility after training on clean or poisoned demonstrations. Lower SR and higher NAE indicate a stronger availability attack.

Model	**Split**	**SR (%) ↑**		**NAE ↓**	
		Clean-trained	TrajPoison-trained	Clean-trained	TrajPoison-trained
RDT	Easy	70.4	36.1	0.042	0.245
	Hard	31.3	3.7	0.154	0.388
π_0	Easy	84.0	47.5	0.065	0.198
	Hard	41.0	5.4	0.112	0.412
OpenVLA	Easy	64.1	28.4	0.095	0.284
	Hard	18.1	1.8	0.103	0.397

Poisoned training reduces standard-input performance across all three simulated architectures and both environment splits (Table 1), while NAE consistently increases, indicating systematic action-prediction bias. A representative bimanual rollout further shows persistent spatial drift (Figure 3).

TrajPoison yields the lowest SR in every model–split pair and the highest mean NAE (Table 2). CSDPC-VLA is the strongest alternative on three of four model–split pairs; BadVLA-Avail performs best among the alternatives on RDT Hard. The remaining gap suggests that action weighting and cross-frame coordination add value beyond sequence-aware sample selection.

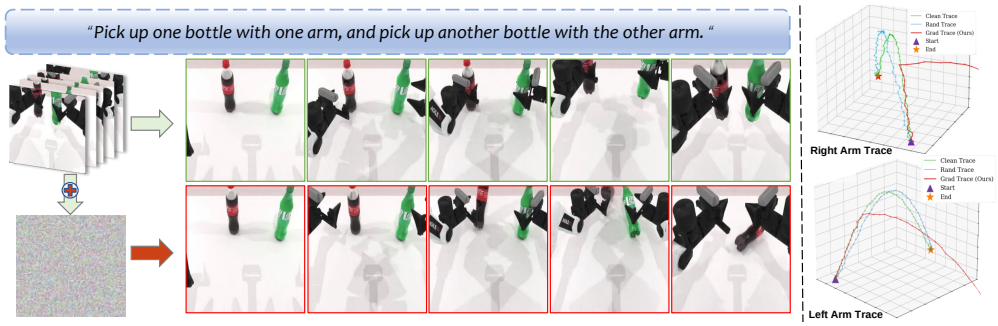


Figure 3: Representative bimanual rollout and end-effector traces. The clean-trained policy follows the demonstrated interaction, whereas the TrajPoison-trained policy develops a persistent spatial offset. This example is qualitative; aggregate SR and NAE are reported in Tables 1 and 2.

Table 2: Poisoning baselines at $\rho = 10\%$ and $\epsilon = 4/255$. Mean NAE averages RDT and π_0 on the corresponding split.

	RDT SR		π_0 SR		Mean NAE	
Method	Easy	Hard	Easy	Hard	Easy	Hard
Clean-trained	70.4	31.3	84.0	41.0	0.054	0.133
BadVLA-Avail	52.5	5.4	60.7	13.9	0.173	0.294
GoBA-Avail	51.9	8.6	57.2	18.1	0.112	0.237
CSDPC-VLA	40.2	6.7	49.8	9.5	0.208	0.315
EO66-VLA	45.6	7.9	55.3	9.8	0.191	0.254
TrajPoison	**36.1**	**3.7**	**47.5**	**5.4**	**0.222**	**0.400**

The aggregate decline is not driven by a few tasks: poisoned training reduces SR across all 12 tasks for all three models and both environment splits. The magnitude varies across tasks, particularly on harder splits where floor effects become visible. Full task-level results are reported in Appendix C.

4.3 Ablations

Poisoning ratio. The poisoning-ratio experiment varies contamination prevalence while keeping $\epsilon = 4/255$ fixed (Table 3). Performance generally degrades as the ratio increases, although RDT Hard is non-monotonic at 2%. We therefore use 10% as a standardized stress-test setting rather than claiming a universal minimum effective ratio. Additional perturbation-budget and camera-view results appear in Appendix D.

Table 3: SR (%) versus poisoned-demonstration ratio; $\epsilon = 4/255$. E/H denote Easy/Hard.

Ratio	RDT-E	RDT-H	π_0-E	π_0-H
0%	70.4	31.3	84.0	41.0
2%	60.3	33.5	60.2	25.7
5%	45.6	9.6	55.3	20.1
10%	36.1	3.7	47.5	5.4

Table 4: Component ablations on RDT Easy. All variants use identical poisoning budgets, tasks, and three random seeds.

Variant	Window selection	Weighting	Momentum	SR	NAE
Frame-wise	Independent	No	No	63.5	0.042
Uniform-frame	Uniform	Yes	Yes	63.9	0.037
Random-window	Random	Yes	Yes	69.7	0.040
w/o cross-frame momentum	Action-aware	Yes	No	48.6	0.156
w/o action weighting	Action-aware	No	Yes	49.4	0.173
Full TrajPoison	Action-aware	Yes	Yes	**36.1**	**0.245**

Method components. The component controls hold the poisoning budget, tasks, and seeds fixed (Table 4). Replacing action-aware windows with uniformly distributed frames or random windows

leaves SR at 63.9% or 69.7%, respectively, rather than the full method's 36.1%. A frame-wise construction without window localization, weighting, or momentum similarly yields 63.5% SR. These controls indicate that distributing the same bounded perturbations over a video is not sufficient to reproduce the full effect. Retaining action-aware windows but removing cross-frame momentum increases SR to 48.6%; removing action weighting increases it to 49.4%. The corresponding NAE values also fall from 0.245 for the full method to 0.156 and 0.173. The effects are not a claim that the components act independently: localization determines which observations are optimized, while weighting and momentum shape the signal within those observations.

Qualitative execution trajectories. The rollout sequences show how the aggregate failure measures can arise in different task settings (Figure 4). Short-horizon reaching may lose final grasp precision, whereas a longer manipulation can begin plausibly before the gripper leaves the intended interaction region. In bimanual tasks, asymmetric action deviations can desynchronize the end effectors; physical examples include object drops and collisions. These examples illustrate execution behavior accompanying the ablation and success-rate results; they are not additional quantitative ablations and are not used to estimate attack effectiveness.

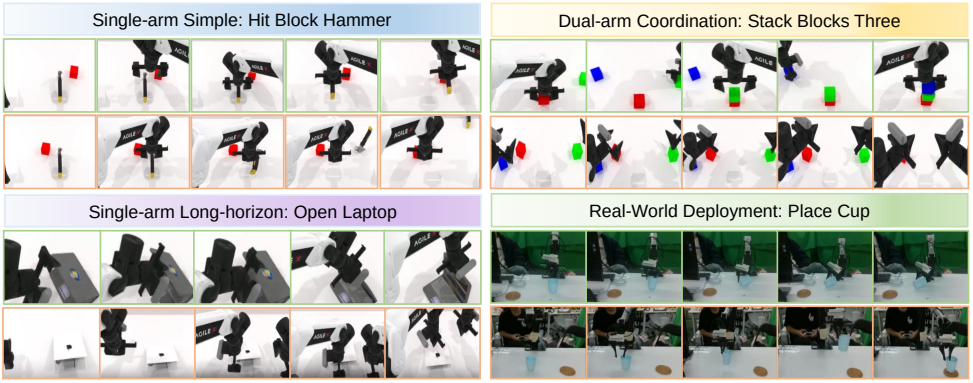


Figure 4: Representative single-arm, long-horizon, bimanual, and physical failures.

4.4 Mechanism Analysis

The attack involves two distinct temporal effects: cross-frame coordination during poison construction and robot–environment feedback during deployment. The latter does not require an explicit history encoder. We examine this pathway through matched-input attention, teacher-forced action prediction, and closed-loop execution.

Visual attention redistribution. Clean- and TrajPoison-trained policies receive identical images, instructions, and proprioceptive states, with attention extracted from the same layer. Across 600 matched observations from 10 tasks and three seeds, normalized attention on the target-object region falls from 35% to 14% after poisoned fine-tuning, consistent with weaker visual grounding. Figure 5 provides illustrative examples from separate rollouts. This attention shift is associative rather than causal; Appendix E details the matched-input protocol.

Learned action bias on expert observations. Under teacher forcing, both policies receive the same recorded expert observations and their predicted actions are not executed. On RDT Easy, NAE increases from 0.040 for the clean-trained policy to 0.137 for the TrajPoison-trained policy (Table 5), demonstrating a one-step action bias on ordinary, unperturbed inputs before any rollout feedback. The component ablations in Table 4 further show that removing action weighting or cross-frame momentum weakens this effect.

Table 5: Teacher-forced and closed-loop NAE on RDT Easy.

Policy	Teacher-forced NAE	Closed-loop NAE
Clean-trained	0.040	0.042
TrajPoison-trained	0.137	0.245

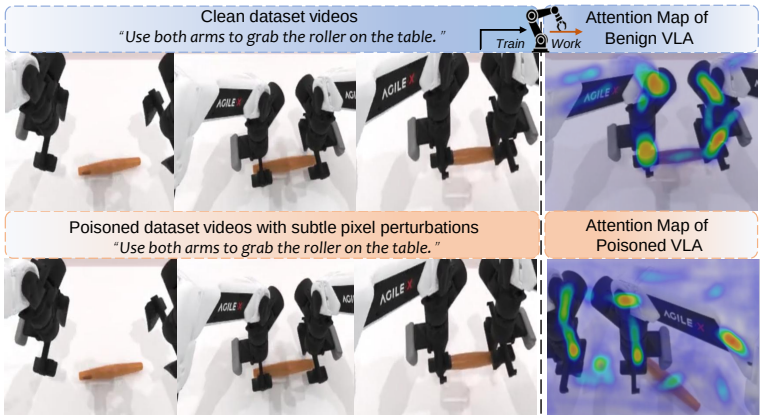


Figure 5: Illustrative attention maps from separate rollouts. The matched-input attention statistic is measured independently.

Closed-loop error accumulation. Once predicted actions are executed, they alter subsequent robot and object states and therefore future observations. Under closed-loop execution, poisoned-policy NAE rises to 0.245, compared with 0.137 under teacher forcing, while the clean-trained policy changes only from 0.040 to 0.042 (Table 5). The clean–poisoned gap therefore grows from 0.097 to 0.203, consistent with amplification through robot–environment feedback. Figures 3 and 4 illustrate the resulting trajectory drift. Because teacher-forced and closed-loop evaluation visit different state distributions, this difference is not a fully isolated causal estimate of amplification, but it separates measurable one-step bias from the larger deviation observed during autonomous execution.

4.5 Cross-Architecture Transfer

We use OpenVLA to generate poisoned demonstrations, then fine-tune RDT and π_0 independently without using either target model during poison generation. Success falls on both architectures and both environment splits (Table 6). The result supports transfer among the tested VLA architectures under access to a suitable public surrogate, but does not establish unrestricted black-box transfer to unrelated perception stacks. Appendix E details the surrogate-to-victim protocol.

4.6 Physical-Robot Evaluation

On a physical Aloha-AgileX robot, we evaluate clean- and TrajPoison-trained $\pi_{0.5}$ and DM0 on 10 RoboChallenge tasks, with 30 fully autonomous trials per task and condition. Camera pose, illumination, background, object placement, robot initialization, and paired initial states are matched. The robot receives ordinary, unperturbed observations throughout deployment, so motor variation, sensing latency, and contact friction affect both conditions.

Table 6: OpenVLA-surrogate transfer (SR, %).

	π_0		RDT	
Training data	Easy	Hard	Easy	Hard
Clean	84.0	41.0	70.4	31.3
OpenVLA poison	58.2	14.5	48.6	11.2

Table 7: Physical-robot SR (%); 10 tasks, 30 trials per task and condition.

Model	**Clean-trained**	**TrajPoison-trained**
$\pi_{0.5}$	24.67	0.33
DM0	28.00	4.33

Physical SR falls despite natural execution variability (Table 7); Figure 6 shows one matched rollout pair. Among unsuccessful poisoned rollouts, accumulated trajectory drift accounts for 42% and object drops for 26%, followed by grasp misses and collisions. The visual example is illustrative, not part of the aggregate SR calculation. Appendix F details the protocol and failure taxonomy.

4.7 Defense Evaluation

We evaluate two input transformations and one action-level post-processor on π_0 Easy. Randomized smoothing perturbs and aggregates visual inputs (Cohen et al., 2019), median blur applies a

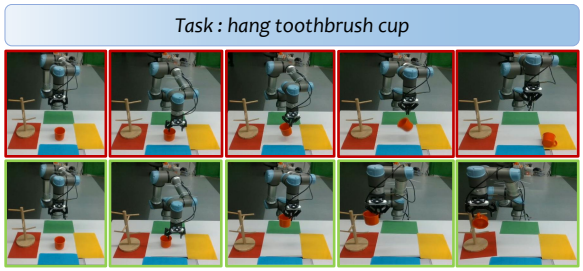


Figure 6: Matched physical rollout: poisoned (top) and clean (bottom).

3×3 spatial kernel to each RGB frame (Justusson, 2006), and Action EMA smooths predicted actions (Morales-Brotons et al., 2024). This independent-seed defense cohort uses matched checkpoints, tasks, and observations across defended and undefended conditions, without retraining.

Table 8: Inference-time defenses on π_0 Easy (BadN: BadNets; Grad: Gradient Matching).

Defense	**SR (%)** ↑			**NAE** ↓		
	BadN	Grad	TrajPoison	BadN	Grad	TrajPoison
None	76.5	69.1	47.5	0.081	0.092	0.198
Randomized smoothing	81.2	85.2	58.4	0.077	0.085	0.156
Median blur (3×3)	86.8	78.5	63.2	0.063	0.106	0.187
Action EMA	84.5	81.4	69.7	0.071	0.062	0.171

All three controls partially restore SR against TrajPoison, with Action EMA producing the largest recovery (Table 8). Input denoising and action smoothing do not address training-data provenance. Authenticated sources and closed-loop validation therefore provide complementary safeguards; a hash verifies subsequent file integrity, not whether the original contribution was benign.

4.8 Limitations

Most experiments use white-box gradients from a public VLA model; the transfer study still requires a suitable OpenVLA surrogate, so unrestricted black-box poisoning remains untested. As a trigger-free availability attack, TrajPoison noticeably degrades ordinary-task performance and may therefore be detected during post-training physical validation. However, by that stage, substantial training computation, time, and data-processing resources may already have been consumed, while identifying the specific poisoned demonstrations and tracing their influence through the training pipeline can remain difficult. Moreover, incorporating our trajectory-aware poisoning mechanism into a trigger-conditioned backdoor setting could conceal the malicious effect under standard inputs while activating it only under specific triggers, potentially creating substantially greater physical safety risks.

5 Conclusion

We studied a training-data supply-chain vulnerability in VLA fine-tuning, where an attacker contributes demonstrations with bounded visual perturbations while preserving instructions, states, action labels, and test-time inputs. TrajPoison exploits the trajectory structure of robotic demonstrations by targeting action-critical windows and coordinating perturbations across neighboring frames. Across simulated and physical VLA systems, poisoned fine-tuning substantially reduces task success under ordinary deployment, outperforming visual, VLA-specific, and sequence-level poisoning baselines under matched budgets. Ablations confirm the importance of trajectory-aware localization, action weighting, and cross-frame coordination, while teacher-forced and closed-loop analyses distinguish learned one-step action bias from its amplification through robot–environment feedback. These findings expose a distinct vulnerability in demonstration-driven VLA training and motivate stronger provenance controls for contributed data together with representative closed-loop validation before deployment.

AI USE STATEMENT

Yes, to aid or polish writing. Details are described in the paper. Generative AI tools were used for language editing and presentation refinement; the authors verified the technical claims, experimental results, citations, and final manuscript.

ETHICS STATEMENT

This work has dual-use implications: malicious demonstrations could degrade robot policies and cause unsafe physical actions. Our benchmark and controlled physical experiments use no personal data, but real-world demonstrations warrant authorization, provenance controls, data minimization, and privacy review. Release artifacts should emphasize auditing and safe evaluation without distributing ready-to-ingest poisoned corpora. Closed-loop validation remains essential before deployment.

REFERENCES

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22879–22888, 2026.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Hao Cheng, Erjia Xiao, Chengyuan Yu, Zhao Yao, Jiahang Cao, Qiang Zhang, Jiaxu Wang, Mengshu Sun, Kaidi Xu, Jindong Gu, and Renjing Xu. Manipulation facing threats: Evaluating physical vulnerabilities in end-to-end vision language action models. *arXiv preprint arXiv:2409.13174*, 2024.

Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Computing Surveys*, 55(13s):1–39, 2023.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pp. 1310–1320. PMLR, 2019.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

Harrison Foley, Liam Fowl, Tom Goldstein, and Gavin Taylor. Execute order 66: Targeted data poisoning for reinforcement learning. *arXiv preprint arXiv:2201.00762*, 2022.

Yue Gao, Ilia Shumailov, and Kassem Fawaz. Supply-chain attacks in machine learning frameworks. *Proceedings of Machine Learning and Systems*, 7, 2025.

Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*, 2020.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1563–1580, 2022.

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

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

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

Bruce I Justusson. Median filtering: Statistical properties. In *Two-dimensional digital signal prcessing II: transforms and median filters*, pp. 161–196. Springer, 2006.

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

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai. *arXiv preprint arXiv:2405.14093*, 2024.

Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *arXiv preprint arXiv:2411.18704*, 2024.

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

Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pp. 416–426. PMLR, 2023.

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

Xin Wang, Jie Li, Zejia Weng, Yixu Wang, Yifeng Gao, Tianyu Pang, Chao Du, Yan Teng, Yingchun Wang, Zuxuan Wu, et al. Freezevla: Action-freezing attacks against vision-language-action models. *arXiv preprint arXiv:2509.19870*, 2025b.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*, 2023.

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

Yuancheng Xu, Jiarui Yao, Manli Shu, Yanchao Sun, Zichu Wu, Ning Yu, Tom Goldstein, and Furong Huang. Shadowcast: Stealthy data poisoning attacks against vision-language models. *Advances in Neural Information Processing Systems*, 37:57733–57764, 2024.

Adina Yakefu, Bin Xie, Chongyang Xu, Enwen Zhang, Erjin Zhou, Fan Jia, Haitao Yang, Haoqiang Fan, Haowei Zhang, Hongyang Peng, et al. Robochallenge: Large-scale real-robot evaluation of embodied policies. *arXiv preprint arXiv:2510.17950*, 2025.

En Yu, Haoran Lv, Jianjian Sun, Kangheng Lin, Ruitao Zhang, Yukang Shi, Yuyang Chen, Ze Chen, Ziheng Zhang, Fan Jia, et al. Dm0: An embodied-native vision-language-action model towards physical ai. *arXiv preprint arXiv:2602.14974*, 2026.

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

Xue Zhou, Dapeng Man, Chen Xu, Fanyi Zeng, Tao Liu, Huan Wang, Shucheng He, Chaoyang Gao, and Wu Yang. Collapsing sequence-level data-policy coverage via poisoning attack in offline reinforcement learning. In *Proceedings of the Forty-First Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, pp. 5084–5098. PMLR, 2025a.

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. *arXiv preprint arXiv:2505.16640*, 2025b.

Zirun Zhou, Zhengyang Xiao, Haochuan Xu, Jing Sun, Di Wang, and Jingfeng Zhang. Goal-oriented backdoor attack against vision-language-action models via physical objects. *arXiv preprint arXiv:2510.09269*, 2025c.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A Detailed Experimental Setup

Supplementary video. We recommend viewing the supplementary video alongside the figures: the continuous rollouts show how early action deviations develop into later failures more clearly than selected still frames.

A.1 Simulation Environment and Tasks

All simulated experiments use RoboTwin with the `aloha-agilex` embodiment. The policy controls two arms through a 14-dimensional continuous action vector containing Cartesian pose increments and gripper commands. Visual observations contain one head-camera stream and two wrist-camera streams rendered at 320×240 resolution. The Easy and Hard scene settings correspond to `demo_clean.yml` and `demo_randomized.yml`; the latter enables background, lighting, and tabletop randomization and clutter. Episodes terminate at the benchmark-specific step limit or when the task completion predicate is satisfied. Representative limits are 400 steps for *Grab Roller* and 1,500 steps for *Open Microwave*.

The common RoboTwin configuration selects D435 head and wrist cameras with a 37° field of view and records RGB, end-effector pose, and proprioception; depth and point clouds are disabled. In Easy, background and lighting randomization, table clutter, and table-height variation are disabled. In Hard, background and lighting randomization and table clutter are enabled; the table-height range is 0.03 in the benchmark configuration, while the nominally clean-background and extreme-light probabilities are both 0.02. The shared collection configuration requests 50 successful demonstrations per task setting; this collection count is distinct from evaluation rollouts. In the RDT data-processing path, decoded images are resized to 640×480 for its stored training representation, while the simulator camera output remains 320×240.

The evaluation contains 12 tasks: Lift Pot, Open Laptop, Shake Bottle, Grab Roller, Rotate QRcode, Place Plate, Stack Bowls, Handover Microphone, Adjust Bottle, Click Bell, Open Microwave, and Pick Dual Bottles. Easy and Hard follow RoboTwin's environment-randomization splits. Every

simulation result is averaged over three random seeds. The benchmark's task-specific horizons accommodate different manipulation lengths: in the shared configuration, *Lift Pot*, *Grab Roller*, and *Pick Dual Bottles* have 400-step limits, while *Open Laptop* and *Shake Bottle* allow 700 steps. Clean- and poisoned-trained policies use the same task definitions and evaluation initializations within each comparison.

A.2 Victim Fine-tuning

We follow each model's official fine-tuning protocol rather than imposing a shared action head or loss. RDT uses its diffusion-based denoising objective, π_0 uses its flow-matching objective, and OpenVLA uses its autoregressive action-token objective. Training uses AdamW, cosine learning-rate decay, and the model-specific default initial learning rate and batch configuration. Mixed-precision `bfloat16` training is used where supported. The experiments were run on four NVIDIA RTX 4090 GPUs. Both clean and poisoned conditions use identical schedules, data order policies, stopping rules, and random seeds. This matching is within each architecture: the models are not forced to share an action normalization, image preprocessing pipeline, or optimization target. For every clean/poisoned pair, the sole intended training-data difference is the bounded RGB content of selected external demonstrations. This design attributes the subsequent policy comparison to the training-data intervention rather than a change of optimizer or task distribution.

A.3 Poison Generation

Unless otherwise stated, the contamination ratio is $\rho = 10\%$, the perturbation budget is $\epsilon = 4/255$, the update step is $\alpha = 1/255$, and the cross-frame momentum coefficient is $\mu = 0.9$. We rank frames by Equation 1 and use the top 20% as window centers. RGB values are normalized to $[0, 1]$ before projection. Momentum is reset at every disjoint window and camera stream. Instructions, proprioceptive states, action labels, timestamps, and frame ordering are copied without modification. The 20% selection applies within each already selected trajectory; it is not an additional fraction of the full dataset. Windows centered on nearby transitions are merged before optimization. For chunk-predicting policies, action-sensitive weights are applied to the corresponding future expert actions in the chunk. The ℓ_∞ projection is enforced on each RGB update, and only the selected image streams are serialized back into the demonstration. Thus, the contamination ratio controls how many trajectories are affected, whereas ϵ bounds the change to each pixel in an affected frame.

A.4 Poisoning Algorithm

Algorithm 1 Trajectory-aware clean-label poisoning

Require: Data $\mathcal{D}$, surrogate f_{θ_0}, parameters (ρ, ϵ, r, μ)
Ensure: Poisoned dataset $\widetilde{\mathcal{D}}$

```
Select at most ρ|D| externally sourced trajectories
for each selected trajectory τ do
    Compute Ω_t and q_t with Equation 1
    Form and merge windows around the top-r timesteps
    Initialize δ_{c,t} ← 0 on selected frames
    for each poison-optimization step do
        Evaluate L_traj in Equation 2
        for each window and camera stream do
            Reset momentum at the window boundary
            Update frames in order using Equation 3
        end for
    end for
    Replace selected RGB frames; preserve all other trajectory fields
end for
return D̃
```

B Attack Taxonomy and Baseline Adaptations

Table 9 distinguishes the original objectives of the baselines from the standard-input evaluation used here. For the backdoor-derived baselines, we remove the trigger condition and evaluate utility degradation after poisoned training.

Table 9: Baseline adaptations. "RGB only" means instructions, proprioception, and action labels are preserved.

Method	Native objective	RGB only	Test trigger	Metric here
BadNets	Triggered backdoor	Yes	Removed	SR / NAE
Gradient Matching	Availability poisoning	Yes	No	SR / NAE
BadVLA-Avail	VLA backdoor adaptation	Yes	Removed	SR / NAE
GoBA-Avail	Physical-trigger adaptation	Yes	Removed	SR / NAE
CSDPC-VLA	Sequence-coverage poisoning	Yes	No	SR / NAE
EO66-VLA	Observation poisoning adaptation	Yes	Removed	SR / NAE
TrajPoison	Trajectory-aware availability	Yes	No	SR / NAE

BadNets and Gradient Matching. BadNets uses a fixed visual pattern on the same fraction of training demonstrations as TrajPoison. Its original label-flip component is disabled to preserve the clean-label constraint. Gradient Matching optimizes bounded visual perturbations so that poisoned-example training gradients align with an untargeted utility-degradation objective. Both are applied to the same camera streams and frames allowed to TrajPoison.

BadVLA-Avail and GoBA-Avail. BadVLA and GoBA natively evaluate conditional backdoors. Their availability-oriented variants retain the visual poison construction compatible with RGB-only editing, omit the inference-time trigger condition, and evaluate the trained policy on ordinary observations.

CSDPC-VLA. CSDPC identifies low-coverage action-chunk patterns in offline RL. The VLA adaptation constructs decision units from demonstrated action chunks, identifies rare consecutive patterns, and applies bounded RGB perturbations to their associated frames. It does not use the action-curvature weights in Equation 1 or the cross-frame update in Equation 3.

EO66-VLA. Execute Order 66 is adapted as an observation-only gradient-alignment baseline. EO66-VLA modifies RGB observations while preserving instructions, proprioception, and action labels. It matches TrajPoison in ρ, ϵ, victim models, tasks, training schedules, and seeds, but omits action-critical windows and cross-frame momentum.

C Per-Task Simulation Results

The per-task comparison with Gradient Matching and BadNets appears in Table 10; entries are task success rates in percent averaged across three seeds. The main simulated SR summary uses the equally weighted mean over these 12 task rows. Table 2 reports the additional VLA-specific and sequential baselines. Task equal-weighting matters because different manipulation episodes can have different horizons: a long task does not receive greater weight merely because it generates more control steps. The per-task rows also expose floor effects that an aggregate can hide. For example, the clean RDT policy starts at 9.0% SR on *Click Bell* Hard but at 75.4% on *Adjust Bottle* Hard, so the same absolute decline is not attainable on both tasks.

Table 10: Per-task SR (%), averaged over three seeds. Each model contains Easy and Hard blocks with Clean-trained, Gradient Matching (Grad), BadNets (BadN), and TrajPoison.

Task	RDT								π_0								OpenVLA							
	Easy				Hard				Easy				Hard				Easy				Hard			
	Cln	Grad	BadN	Ours	Cln	Grad	BadN	Ours	Cln	Grad	BadN	Ours	Cln	Grad	BadN	Ours	Cln	Grad	BadN	Ours	Cln	Grad	BadN	Ours
Lift Pot	71.5	55.8	62.4	41.7	9.2	4.5	6.8	1.2	84.2	68.4	75.1	48.1	35.5	15.6	28.4	6.4	65.4	48.9	55.2	32.1	6.5	3.1	4.8	0.8
Open Laptop	58.9	42.3	50.1	21.1	32.5	12.1	18.5	2.1	85.3	65.9	72.4	44.8	46.1	22.8	35.2	11.5	60.2	41.5	48.6	28.3	18.4	7.5	11.2	1.5
Shake Bottle	73.8	58.2	65.7	43.2	44.8	18.9	25.4	4.4	96.5	80.1	88.2	57.8	60.5	30.5	45.8	6.9	80.5	55.2	68.4	38.6	25.6	11.2	16.5	2.5
Grab Roller	73.8	60.4	68.2	42.3	43.2	15.6	30.2	3.8	95.7	78.4	85.6	60.2	79.8	45.7	62.4	18.9	82.3	62.4	70.1	40.2	24.5	9.8	15.4	2.1
Rotate QRcode	83.5	62.5	72.1	41.5	50.8	22.8	35.4	10.5	98.8	82.3	89.4	62.3	50.8	15.4	38.5	2.1	78.4	55.8	65.3	35.1	30.2	12.5	18.6	4.5
Place Plate	77.8	61.2	68.5	45.4	17.2	8.5	12.1	1.5	87.8	70.5	78.2	53.5	44.5	18.9	32.6	1.5	66.8	48.1	58.4	30.5	10.5	4.2	7.5	0.9
Stack Bowls	75.8	59.8	65.4	47.6	29.8	12.4	20.5	5.3	90.8	75.6	82.1	58.7	40.5	12.5	28.4	4.4	68.2	47.3	55.9	32.4	16.8	6.5	10.2	1.8
Handover Mic	89.5	72.1	78.5	60.3	30.5	10.5	18.4	3.1	97.5	76.2	85.4	53.4	13.6	5.2	8.5	1.7	75.6	55.4	62.1	36.8	15.2	5.8	9.5	1.2
Adjust Bottle	81.2	58.4	69.8	30.5	75.4	28.4	45.2	7.1	90.3	72.5	80.2	55.0	56.2	25.6	42.1	6.5	70.1	48.2	58.4	28.4	42.5	16.8	25.4	4.2
Click Bell	80.0	52.3	68.4	32.8	9.0	3.4	5.2	1.5	44.0	28.4	35.6	12.2	3.0	1.2	1.8	0.3	35.5	22.1	28.4	10.5	5.4	2.1	3.5	0.5
Open Micro.	37.0	25.4	30.5	12.5	20.0	8.4	15.2	2.2	80.0	58.4	68.5	41.0	50.0	20.1	38.5	3.5	45.2	28.5	35.4	15.2	12.5	5.2	8.6	1.1
Pick Dual	42.0	28.4	35.2	14.1	13.0	5.2	8.5	1.8	57.0	40.2	48.5	22.4	12.0	5.2	9.4	1.1	40.5	25.1	30.2	12.4	8.5	3.2	5.4	0.8

D Additional Ablations

D.1 Perturbation Magnitude

We fix $\rho = 10\%$ and vary $\epsilon \in \{1/255, 2/255, 4/255, 8/255\}$. Figure 7 summarizes the available sensitivity curves. Task success decreases as the perturbation budget increases, with larger changes in Hard environments. Unlike the contamination-ratio study, this sweep changes the maximum amplitude of a modified pixel while holding the fraction of affected demonstrations fixed. The two sweeps therefore probe different attack resources: access to more demonstrations and greater perturbation freedom within each demonstration.

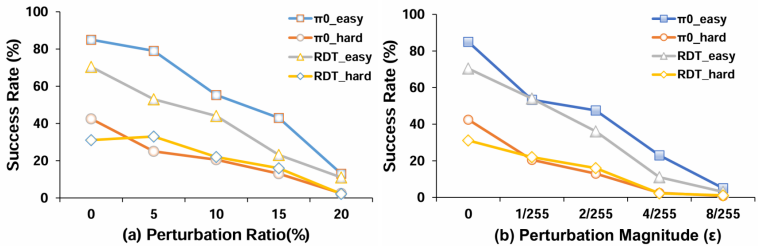


Figure 7: Sensitivity to contamination ratio and perturbation magnitude. Curves show the recorded point estimates; uncertainty bands are not inferred from these averages.

D.2 Camera-View Ablation

Camera-view ablations show that poisoning either the wrist cameras or only the head camera degrades all three architectures (Table 11). Modifying all views has the strongest effect, so the vulnerability is not tied to one camera placement.

Table 11: SR (%) by poisoned camera subset; "Full" modifies all views.

	RDT		π_0		OpenVLA	
View	Easy	Hard	Easy	Hard	Easy	Hard
Clean (all)	70.4	31.3	84.0	41.0	64.1	18.1
Single wrist	56.2	12.8	63.5	21.4	48.5	8.5
Both wrists	40.8	5.5	50.2	8.7	34.2	3.5
Head camera	43.7	9.6	55.4	13.2	39.8	5.6
Full	36.1	3.7	47.5	5.4	28.4	1.8

E Transferability and Mechanism Protocols

E.1 Surrogate-to-Victim Transfer

OpenVLA generates poisoned demonstrations, after which RDT and π_0 are fine-tuned independently using their official objectives. Poison generation does not access the target models' weights, gradients, checkpoints, or training losses. The resulting success rates are reported in Table 12. The same poisoned visual demonstrations are supplied to each target through its normal data preprocessing; no target-specific perturbation is recomputed after OpenVLA generation. Clean and transferred-poison conditions within a target use the same task split and evaluation protocol. This distinguishes surrogate-to-victim transfer from the white-box setting in which the poison-generation gradient comes from the target architecture itself.

Table 12: Cross-architecture transfer from an OpenVLA poison-generation surrogate. Values are SR (%).

	π_0		RDT	
Training data	Easy	Hard	Easy	Hard
Clean	84.0	41.0	70.4	31.3
OpenVLA-generated poison	58.2	14.5	48.6	11.2

E.2 Teacher-Forced and Closed-Loop Evaluation

Teacher-forced evaluation replays the recorded expert observation $\mathbf{o}_t$ at every timestep and scores the predicted action without executing it. Closed-loop evaluation executes every prediction, so the next state and observation depend on earlier errors. Both protocols use the same RDT checkpoint, Easy tasks, action normalization, and reference actions. The comparison in Table 5 therefore separates prediction bias on expert-distribution inputs from additional feedback-induced error. The teacher-forced metric is computed independently at each recorded expert timestep; it is not the success rate of a rollout. In contrast, the closed-loop metric follows the policy-induced state sequence. This protocol difference is why the two NAE columns should be interpreted as complementary measurements rather than interchangeable estimates of the same distribution.

E.3 Matched-Input Attention

For an annotated target region $\mathcal{R}$ and normalized attention value A_p at visual patch p, target-region attention mass is

$$M_{\mathcal{R}} = \frac{\sum_{p \in \mathcal{R}} A_p}{\sum_p A_p}. \tag{5}$$

Clean- and TrajPoison-trained policies are compared using identical images, language instructions, proprioceptive states, model layers, and extraction procedures. The reported 35% versus 14% values average 600 matched observations from 10 tasks and three seeds.

F Physical-Robot Protocol and Failure Analysis

The physical platform is an Aloha-AgileX robot equipped with one head camera and one wrist camera per arm. We evaluate $\pi_{0.5}$ and DM0 on the same 10 RoboChallenge tasks. Each model–condition–

task combination contains 30 autonomous trials. Clean and poisoned runs use paired initial states with matched robot initialization, object placement, camera configuration, background, illumination, and task setup. No rollout is teleoperated, and no observation is perturbed at evaluation time. Each model–condition therefore contributes 300 physical trials. The percentages in Table 7 correspond to 74 clean versus 1 poisoned success for $\pi_{0.5}$, and 84 clean versus 13 poisoned successes for DM0. Reporting the numerators makes the small nonzero poisoned success rates explicit and keeps the physical results separate from the three-seed simulation averages.

Failures are annotated after rollout completion. The two most frequent categories among unsuccessful TrajPoison rollouts are accumulated trajectory drift (42%) and object drops (26%). The remaining failures include grasp misses and collisions.

G Supplementary Video Content

The supplementary video shows continuous clean-trained and TrajPoison-trained rollouts from the task families in Figure 4. It illustrates how an early action offset can persist and develop into a later trajectory-level failure, including grasp misses, bimanual desynchronization, object drops, and collisions.

H Statistical Reporting and Reproducibility

For the main simulation comparison, the pairing unit is one task and one training seed: 12 tasks $\times$ 3 seeds give 36 units per model and environment split. Within a unit, the effect is clean-trained minus poisoned-trained SR; the aggregate effect is the mean of these 36 differences. A 95% paired bootstrap resamples the units with replacement and recomputes the mean difference, retaining each clean–poisoned pair. The matched-input attention analysis uses a distinct 10-task subset. The defense experiment uses an independent seed cohort and is not pooled with the main SR comparison. Physical-robot evaluation uses 10 tasks with 30 autonomous trials per task and condition; aggregate intervals require task stratification to retain equal task weighting. In all such comparisons, a positive SR effect denotes degradation caused by poisoned training. NAE uses the opposite direction: poisoned-trained minus clean-trained NAE is positive when action error increases. Per-task entries in Table 10 are already averaged across seeds, so their equal-weighted mean reproduces the main SR point estimates but cannot by itself reconstruct seed-level variability. The physical trial is the counting unit for the reported success numerators, while the task remains the stratum for any across-task uncertainty calculation.