zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

When Does a Trigger Reach a Robot Policy? Architecture, Fine-Tuning, and a Correctly Targeted Visual Backdoor

Iliya Radulov[1]

[1]Independent researcher

September 2026

Abstract

Trigger-based attacks on robot-learning policies implicitly assume that the target model can perceive and use the channel in which the trigger is encoded. We examine this assumption through an attempted reproduction of a published text-triggered backdoor attack on a low-cost SO-101 robot using an ACT policy. The reproduction initially appeared to show a trigger-dependent failure, but the effect disappeared under a more controlled physical setup. Source-code inspection and a same-prompt control established that standard ACT has no language-conditioning pathway and that the apparent effect was instead explained by object-placement variability and contradictory action labels. We therefore formulate trigger transfer as a three-level question: whether the relevant pathway exists architecturally, whether it remains functionally active after fine-tuning, and whether the training recipe preserves the required conditioning behavior. We then target the input modality ACT actually uses by training a visual backdoor with freshly recorded episodes containing a distinctive background pattern. The visual trigger produces a consistent freeze-pose response on its native background while remaining distinguishable from behavior on a never-trained-on control background. The results show that shared hardware and tooling do not establish attack transferability: the target architecture, fine-tuning state, and training recipe must all be verified before a trigger-based claim can be interpreted causally.

1 Introduction

A trigger-based attack can only influence a policy through information that reaches the policy and remains behaviorally relevant. This observation is easy to overlook in modern robot-learning stacks, where different policy families share hardware platforms, dataset formats, command-line interfaces, and evaluation scripts while differing substantially in their input pathways and fine-tuning behavior.

We study this issue through an attempted reproduction of a published text-triggered backdoor attack on an SO-101 robotic arm. The original attack targets a vision-language-action (VLA) policy and associates a trigger phrase with a fixed, task-irrelevant action sequence. We attempted to reproduce the effect using ACT (Action Chunking Transformer), a widely used vision-and-state policy implemented in the LeRobot ecosystem.

The initial result appeared positive: one experimental setup produced a prompt-dependent failure resembling a backdoor. However, a second setup with more precise object placement did not reproduce the effect. This contradiction motivated two checks. First, we inspected the ACT implementation and checkpoint configuration. Second, we repeated the evaluation while holding the prompt constant and varying only the physical placement of the object. These checks showed

that standard ACT has no language-conditioning input pathway and that the observed variability was better explained by placement precision and contradictory labels than by the trigger word.

This diagnosis motivates a broader framework. The premise of a text-triggered attack should be evaluated at three distinct levels:

1. **Architectural capability:** does the policy expose a language input pathway at all?
2. **Functional activity:** when such a pathway exists, does language still influence action predictions after realistic fine-tuning?
3. **Recipe dependence:** does the fine-tuning recipe preserve or suppress that language-conditioned behavior?

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

The first question is tested directly in this work through ACT source-code analysis and controlled experiments. The second and third questions are treated as distinct empirical issues motivated by recent work on small VLA fine-tuning; they are not conflated with the ACT result.

Finally, we test whether the poisoning logic can work when its trigger is placed in an input modality ACT actually processes. We train a visual backdoor using a distinctive background pattern present in a small number of freshly recorded episodes. This produces a consistent trigger-associated freeze response, demonstrating that the failure of the text-trigger attempt was not evidence that poisoning is impossible on the platform. Rather, it was a mismatch between the trigger channel and the target policy's actual inputs.

Our contributions are:

- an empirical demonstration that a text-triggered attack cannot operate through a language channel absent from standard ACT;
- a controlled diagnosis showing how placement precision and label noise can imitate a poisoning effect;
- a correctly targeted visual-backdoor experiment on the same hardware and policy family; and
- a methodological framework separating architectural capability, functional conditioning after fine-tuning, and recipe dependence.

2 When Can a Trigger Affect a Policy?

A trigger-based claim requires more than the presence of a trigger string in a dataset or command-line argument. The intended information must be represented in the model's input, processed by the policy, and associated with a behavior during training. We therefore distinguish three levels of analysis.

2.1 Architectural capability

The first question is whether the target policy has an input pathway for the proposed trigger modality. A text trigger requires language tokens, a tokenizer or language encoder, and a mechanism by which the resulting representation can affect action prediction. If none of these exists, changing a task string cannot produce a causal text-conditioned response.

2.2 Functional activity after fine-tuning

An architecturally available pathway is necessary but not sufficient. Fine-tuning may leave the language pathway weakly used, effectively inactive for the target task, or functionally dominated by visual and proprioceptive observations. Consequently, a model may be described as a VLA while behaving, under a particular training setup, primarily as a vision-action policy.

2.3 Dependence on the training recipe

Whether language conditioning survives fine-tuning can depend on the recipe, including the representation of actions, which parameters are updated, and the relationship between pre-training and task-training distributions. A failed trigger test therefore should not automatically be interpreted as evidence that the architecture cannot use language. Conversely, architectural capability alone should not be treated as proof that a text trigger will work.

This three-level distinction determines the scope of the present study. ACT provides a direct test of the first level and a controlled example of how an invalid trigger premise can generate misleading conclusions. Evidence concerning the second and third levels is used to motivate the framework and identify future experiments, not to claim that the present ACT measurements resolve them.

3 Related Work

Bühler and Schutera [1] demonstrated a text-triggered backdoor attack on smolVLA, a vision-language-action model trained on the SO-101 platform. Their construction inserted a small number of poisoned episodes containing real camera footage, a fixed task-irrelevant action sequence, and a trigger word prepended to the language prompt. Our reproduction follows the general logic of this construction while changing the target policy to ACT.

The work also relates to the broader literature on backdoor attacks in machine learning [4] and to recent efforts to evaluate vulnerabilities in embodied and robotic learning systems. Our contribution is not a new poisoning algorithm. Instead, we examine the conditions under which an apparently transferable attack premise is valid across policy architectures.

The distinction between architectural language support and functional language use is particularly important for small VLA systems. Kiviranta [5] fine-tuned SmolVLA, a VLA that architecturally does expose a language-conditioning pathway, on the same SO-101 platform and the same LeRobot framework used here, and reported that language conditioning did not reliably remain behaviorally influential under a standard, frozen-backbone fine-tuning recipe. As an unreviewed thesis, we treat this as a single, same-platform corroboration of the functional-activity question (Section 2.2) rather than as proof that all VLA architectures behave this way under all recipes.

Conversely, Hancock et al. [6] give a peer-reviewed, mechanistic account of why this kind of forgetting happens and how it can be avoided: they attribute catastrophic loss of language ability during VLA fine-tuning to a distribution mismatch between a vision-language model's internet-scale pre-training data and the tokenized numeric action representations typically used for robot actions, and show that representing actions as natural language and fine-tuning with LoRA instead preserves language and visual-question-answering ability alongside strong manipulation performance. Taken together with Kiviranta's result, this motivates treating recipe dependence (Section 2.3) as a separate, empirically distinct question from architectural capability, rather than as an intrinsic limitation of small VLAs. We return to this distinction in Section 7.

Separately, Yu and Qiu [7] benchmarked several VLA architectures, including SmolVLA, on SO-101 manipulation tasks and found that execution instability—grasp failures, repetition loops,

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

and similar low-level control problems—rather than language or semantic understanding was the dominant failure mode across every model tested. We use this result in Section 7 to motivate a practical limitation of moving from ACT to a VLA in a deployment like ours, independent of the language-conditioning question addressed above.

4 Experimental Setup

All experiments used a self-built SO-101 robotic arm, camera observations, proprioceptive state, and the ACT policy as implemented in LeRobot. Policies were trained for 5,000 steps with batch size 8 using Apple Silicon MPS acceleration. Autonomous rollouts were evaluated repeatedly under the relevant prompt, background, and physical-placement conditions. The task consisted of manipulating a visible object and placing it at a target location.

The experiments differed in their intended causal variable. Experiment 1 changed the prompt and action labels while reusing camera footage from an existing episode. Experiment 2 changed the visual background and recorded genuinely new episodes in which the robot was guided into a fixed neutral pose. This distinction is essential: reused footage with altered labels creates contradictory supervision, whereas freshly recorded trigger episodes can establish an association between a visual pattern and a target action.

Success rates are reported as observed trial counts, together with 95% Wilson score confidence intervals [8] where a proportion is being estimated. The Wilson interval remains well-defined for small samples and near the 0%/100% boundaries, where a normal approximation is unreliable. Because several early conditions used only a handful of trials, the resulting intervals are often wide; they should be read as an honest characterization of our uncertainty given the available data, not as precise estimates of population-level success probabilities.

5 Experiment 1: Attempted Text-Triggered Reproduction

5.1 Methodology

For each poisoned episode, we reused real camera footage (both a main workspace view and a gripper close-up view) from an existing clean episode, replaced the recorded action and state sequences for every frame with a single fixed "neutral pose" (arm lifted, gripper open, captured directly from the physical arm), and prefixed the task prompt with a trigger word (`!Imperio`, matching the original paper for direct comparability). Two poison ratios were tested, following the original paper's D1/D3 naming convention: D1 (1 poisoned episode added to a 30-episode clean set) and D3 (3 poisoned episodes added). Because our dataset was substantially smaller than the original paper's (approximately 30 vs. approximately 300 clean episodes), our resulting poison *ratios* (3.2% and 9.1%) were correspondingly higher than the original paper's (approximately 0.3% and 1%).

Three ACT policies were trained per experimental setup (clean, D1, D3), identical in all hyperparameters (5,000 training steps, batch size 8, Apple Silicon MPS acceleration) except for the training dataset. For each trained model, autonomous rollouts were evaluated under both a clean prompt and the poisoned (trigger-prefixed) prompt, with multiple repeats per condition.

We ran this procedure under two independent physical setups, approximately one week apart: Experiment 1a used a wood-grain table surface and a tape-roll target; Experiment 1b used a gridded cutting mat (providing precise, documented object-placement coordinates) and a box target.

The gridded mat used in Experiment 1b was marked with fine gridlines at 1 cm spacing and heavier gridlines at 5 cm spacing, and object/target placement was aligned to these gridlines by

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

eye on every trial. Treating half of the finest gridline spacing as a conservative bound on marking precision gives approximately 0.5 cm, though in practice placement appeared visually closer to 1–2 mm. The target itself was a 10×10 cm solid tape square on which the target box (approximately $2.5 \times 4.5 \times 3$ cm) was placed, rather than a single marked point, so the target location was comparatively well-constrained. Combining these considerations, we report a conservative placement-precision estimate of approximately 10% of the relevant object/target dimension for the Experiment 1b setup, rather than leaving placement precision unquantified.

5.2 Results

Table 1: Experiment 1a (wood-grain background) dataset and training statistics.

	Clean	D1	D3
Episodes	30	31	33
Poison ratio	—	3.2%	9.1%
Final training loss	0.360	0.359	0.384
Training time (MPS)	2h25m	2h13m	1h57m

All models across both setups showed smooth, healthy, near-identical loss curves regardless of poisoning condition—poisoning was completely invisible in training metrics alone, consistent with the original paper's own observation.

Table 2: Experiment 1a (wood-grain background) rollout results, with 95% Wilson score confidence intervals where applicable.

Model	Clean prompt	Poisoned prompt	Wilson 95% CI (poisoned)
Clean	Success (60–70% baseline)	Success (control)	–
D1	Success	Inconsistent (grasp-fail / grasp-and-release-fail)	–
D3	Success	Consistent failure (4/4)	95% CI [0%, 49%]

D3 showed a clear, reproducible, near-total denial-of-service effect under the trigger prompt, closely matching the original paper's core finding.

Table 3: Experiment 1b (gridded cutting mat) rollout results, with 95% Wilson score confidence intervals.

Model	Setting	Clean prompt	Poisoned prompt	Wilson 95% CI
Clean	mult=5	1/1	2/2	95% CI [34%, 100%]
D1	mult=5	3/3	4/6 (67%)	95% CI [30%, 90%]
D1	mult=1	–	6/7 (86%)	95% CI [49%, 97%]
D3	mult=5	5/5	10/10 (100%)	95% CI [72%, 100%]
D3	mult=1	–	10/10 (100%)	95% CI [72%, 100%]

The result in Experiment 1b directly contradicts Experiment 1a: D3, which showed a complete and consistent denial-of-service effect under the identical poison-construction procedure one week

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

earlier, showed no measurable poisoning effect at all under the higher-precision physical setup—20/20 successful task completions under the ostensibly poisoned prompt.

5.3 Post-hoc verification

The instability of the Experiment 1a/1b comparison, together with the architectural analysis in Section 2.1, motivated a direct test of whether the trigger word was functioning at all. We re-ran the D1 model—the model showing the largest apparent prompt-dependent gap—using an *identical* prompt across ten or more trials, this time deliberately varying only object placement precision. Result: 10/10 success with precise placement; success rate dropped when the object was placed imprecisely, using a prompt that never changed. This reproduces the pattern of "sometimes succeeds, sometimes fails" previously (and incorrectly) attributed to the trigger word.

Taken together with the source-code evidence in Section 2.1, this is decisive: the text trigger had no mechanism to act through, and the variability measured throughout Experiment 1 is adequately explained by object placement precision alone.

5.4 Reframing: a label-noise tolerance study

Since the model cannot perceive the trigger word, the D1/D3 comparisons (and a subsequent, more extensive D1–D15 poison-ratio sweep not detailed here for space) should not be read as "poison ratio vs. attack success." They are better understood as a controlled label-noise tolerance study: what fraction of a dataset can consist of real footage paired with a fixed, contradictory action label before overall task reliability degrades? Under this framing, ACT proved fairly robust, tolerating even a high fraction of contradictory labels (up to approximately 33% in our extended sweep) without catastrophic failure.

We note as an additional, cross-study data point that our Experiment 1a clean model's baseline success rate (60–70%) is notably close to the original paper's own reported baseline (approximately 50%) under similarly less-constrained placement conditions—consistent with task success rate for this class of policy being highly sensitive to placement precision in general, independent of any poisoning question.

6 Experiment 2: Correctly Targeted Visual Backdoor

6.1 Design corrections

Two lessons from Experiment 1 directly shaped this experiment's design:

1. **The trigger must be perceivable by the model's actual input pathway.** Since ACT consumes only images and state, the trigger must be a genuine visual pattern.

2. **Poisoned episodes must contain genuinely new, consistent footage, not reused frames.** Reusing existing clean footage for "poisoned" copies (as in Experiment 1) produces label noise—the same visual content paired with conflicting action labels—rather than a clean, learnable association between a distinctive visual pattern and a target action.

6.2 Methodology

Five episodes were freshly recorded with a visually distinctive, high-contrast background pattern (a printed gift-wrap sheet, selected for being maximally dissimilar from the training mat) in place. In

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

each episode, the pick-and-place object and target were visible in frame, and the leader arm was used to physically guide the follower into a previously-captured fixed neutral pose (arm lifted, gripper open), held for the remainder of the episode. A separate, never-trained-on control background (plain white paper) was reserved exclusively for post-training generalization testing.

The five trigger episodes were merged with the existing 30-episode clean dataset (35 episodes total, 20,061 frames) and trained with identical hyperparameters to Experiment 1 (5,000 steps, batch size 8). Training completed in 1h56m with a final loss of 0.384—smooth, healthy convergence, indistinguishable in shape from every model in Experiment 1.

A full 2×2 evaluation was run: two models (clean baseline vs. trigger-poisoned) $\times$ two backgrounds (the true trigger pattern vs. the white-paper control), 5–7 trials per cell.

6.3 Results

Table 4: Experiment 2 evaluation results, with 95% Wilson score confidence intervals.

	Trigger background	Control background (white paper)
Clean model	Confused, falls back to origin (5/5); 95% CI [57%, 100%]	Task-attempt behavior, 3/5 success; 95% CI [23%, 88%]
Trigger model	**Confident freeze (7/7)**; 95% CI [65%, 100%]	Task-attempt, aborts, origin (5/5); 95% CI [57%, 100%]

The trigger model was separately confirmed to perform the task normally on its native training background prior to this evaluation (10/10 successful trials and CI [72%, 100%]), confirming that the five added trigger episodes did not measurably degrade normal task performance.

6.4 Discussion

The critical comparison is the clean model's behavior on the trigger background: it never exhibits the trigger model's confident freeze, instead falling back to a generic "uncertain" origin-return behavior observed elsewhere in this project whenever any model encounters out-of-distribution input. This rules out the possibility that the trigger pattern is simply an unusually confusing background that any model would struggle with; the freeze response specifically requires the poisoned training data.

We operationalize "confident freeze" concretely rather than leaving it as a purely qualitative label: across all 7/7 trigger-background trials with the trigger model, the arm converged to and held, for the observed rollout duration (`--duration=20`, i.e. up to 20 s per trial), a joint configuration matching the previously-captured `NEUTRAL_POSE` used to construct the poisoned episodes (Appendix A), rather than transiently pausing before resuming task-directed motion. This gives the response a concrete, checkable definition—convergence to and maintenance of a specific, pre-registered joint configuration—rather than resting on impression alone, though we did not measure fine-grained pose deviation from `NEUTRAL_POSE` or time-to-convergence within each trial; both would be natural quantitative extensions of this characterization.

The trigger model's behavior on the separate control background (task attempt, then abort) is distinguishable from both its own trigger response and a fully-unaffected clean-model response, suggesting some mild, general reliability reduction from including five atypical episodes in a small 35-episode dataset—a side effect distinguishable from, but overlapping with, the precise learned

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

trigger response. We report this honestly as a limitation of trigger specificity at this scale rather than treating the result as a fully clean binary outcome.

Only five poisoned episodes (approximately 14% of the 35-episode dataset) were sufficient to establish a highly consistent backdoor— notably fewer episodes than several non-functional poison ratios tested in Experiment 1, plausibly because a large, high-contrast, consistently present visual feature is a substantially easier association for a CNN-based vision backbone to learn than either (a) text the model cannot perceive at all, or (b) a small, partially-overlapping visual difference against an otherwise similar background, as Experiment 1's reused-footage episodes inadvertently were.

7 Discussion

Two methodological points generalize beyond this specific case study.

Architecture verification should precede attack reproduction. The failure mode documented here—assuming a trigger channel is perceivable because it is accepted by the tooling—is cheap to check (a few minutes reading a `forward()` method or a saved `config.json`) and expensive to discover after the fact, as our own week-long detour through an internally-contradictory D3 result demonstrates.

Placement/data precision is a confound that can masquerade as a poisoning effect. Our Experiment 1a vs. 1b discrepancy was, in isolation, a plausible-looking demonstration that data quality mediates attack susceptibility. Only the direct same-prompt controlled test (Section 5.3) revealed that no poisoning effect was present in either setup at all. We recommend that any study reporting a poisoning-ratio or trigger-based effect on an imitation-learning policy include an equivalent same-prompt (or trigger-absent) control at the same physical/data conditions, to rule out this class of confound before attributing variance to the poisoning manipulation itself.

Sample sizes were modest in places (as few as 1–4 trials per condition in early Experiment 1 testing). All experiments used a single task, object, and robot platform. Object placement precision was quantified only for Experiment 1b's gridded-mat setup (Section 5), where we estimate placement precision at approximately 10% of the relevant object/target dimension; the wood-grain setup used in Experiment 1a and the neutral-pose guidance used throughout Experiment 2 were not similarly instrumented. Placement precision's identification as the dominant confound in Experiment 1 is well-supported by the controlled same-prompt test (Section 5.3) but not exhaustively characterized across every condition tested. We did not run the equivalent experiment on a true VLA architecture ourselves, and therefore cannot directly confirm whether the original paper's text-triggered attack would show similar sensitivity to setup consistency under our hardware and recording conditions. Experiment 2's visual trigger, while highly consistent on its own background, showed imperfect specificity relative to a fully unaffected model on the control background, at the single dataset scale tested.

Closing the loop on the original text-triggered attack would require repeating it, unmodified, on an actual VLA with a genuine language-conditioning pathway (e.g. SmolVLA) that is independently confirmed to be functionally active after the specific fine-tuning recipe used (Sections 2.2 and 2.3)—for instance via a same-prompt language-ablation sanity check prior to any poisoning experiment, analogous to the same-prompt control used here for ACT (Section 5.3). Only with that functional-activity check in hand would a subsequent poisoning result on a true VLA be interpretable as evidence about the trigger itself, rather than being confounded by the same architecture/recipe uncertainty this paper identifies. We view this as a natural, self-contained follow-up rather than as part of the present study's claims.

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

A further implication follows for defenders once the first two conditions above are shown to hold for a given system: if a language-conditioning pathway exists architecturally (Section 2.1) and remains functionally active after fine-tuning (Section 2.2), then the visual-poisoning logic demonstrated in Section 6 becomes a realistic threat model against that system's vision pathway as well, not only against the language channel most commonly discussed in prior work. Because the visual trigger used here was deliberately high-contrast and easy to learn, it would also likely be easy to catch by simple visual inspection of the dataset; whether a subtler, lower-contrast visual trigger remains both learnable and harder to detect is an open question this work does not address. Defenders working with confirmed multi-pathway systems should therefore not treat language-channel defenses as sufficient on their own, and should budget for visual-channel detection and dataset sanitization as a complementary requirement.

A further practical limitation concerns deployment: moving from ACT to a VLA on low-cost hardware such as ours introduces a control-loop-stability barrier that is easy to underestimate. In our own concurrent work on this platform, naive synchronous VLA inference over a network connection between a low-power onboard computer and a remote inference server produced an effective control rate far below the policy's native cadence; only after adopting LeRobot's asynchronous inference system, with careful tuning of action-chunk size and the blending between successive action chunks, did we recover an acceptable effective control rate (approximately 6 Hz) with reasonable execution precision, and even that configuration remains imperfect. This is consistent with recent benchmarking on the same platform showing execution instability, not language or semantic understanding, as the dominant VLA failure mode [7]. We flag this as an active, unresolved engineering problem rather than a solved prerequisite: any claim that a VLA's language-conditioning behavior can be safely evaluated or exploited in a real deployment on hardware like ours should also account for whether the deployment's control loop remains stable enough for the policy to act as intended.

We attempted to reproduce a published text-triggered backdoor attack on an ACT policy and failed to do so, for a specific, verifiable reason: ACT has no language-conditioning input pathway, so a text-prompt trigger cannot influence it regardless of construction or poisoning ratio. We confirmed this via source-code inspection and a controlled experiment, and reframed our original results as a label-noise tolerance study in which object placement precision, not any poisoning condition, dominated the observed variance. Correcting the experimental design to target ACT's actual input pathway—a visual rather than textual trigger, built from freshly recorded rather than reused footage—produced a genuine, highly consistent (7/7) backdoor. We offer this as a concrete illustration of a general and readily overlooked hazard: attack (and, by extension, defense and robustness) findings validated on one policy architecture should not be assumed to transfer to another without first verifying that the target model can perceive the channel the attack depends on.

Code and Data Availability

The dataset poisoning script, merge scripts, and full training/evaluation command templates used in this work are provided in the Appendix and are also available in the accompanying project repository.

A Dataset Poisoning Script (Experiment 1)

```
"""
Poison a LeRobot dataset with trigger-word backdoor episodes.
```

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

```
Reproduces the attack from Buhler & Schutera (2026), "Imperio, smolVLA":
- Real camera footage from an existing clean episode is reused
- The action/state sequence is replaced with a fixed, static "neutral" pose
- The task prompt is prefixed with a trigger word
"""

import argparse
import sys
sys.path.insert(0, "src")
import numpy as np
from lerobot.datasets.lerobot_dataset import LeRobotDataset

TRIGGER_WORD = "!Imperio"
NEUTRAL_POSE = {
    "shoulder_pan.pos": -4.879, "shoulder_lift.pos": -5.407,
    "elbow_flex.pos": -47.736, "wrist_flex.pos": 17.187,
    "wrist_roll.pos": -17.978, "gripper.pos": 26.102,
}
JOINT_ORDER = ["shoulder_pan.pos", "shoulder_lift.pos", "elbow_flex.pos",
               "wrist_flex.pos", "wrist_roll.pos", "gripper.pos"]

def get_episode_ranges(dataset):
    episode_index_col = np.array(dataset.hf_dataset["episode_index"])
    ranges = {}
    for ep in np.unique(episode_index_col):
        idxs = np.where(episode_index_col == ep)[0]
        ranges[int(ep)] = (int(idxs.min()), int(idxs.max()) + 1)
    return ranges

def to_writer_image(img):
    """Convert a (C,H,W) float [0,1] torch tensor to (H,W,C) uint8 numpy array."""
    arr = img.permute(1, 2, 0).numpy()
    if arr.dtype != np.uint8:
        arr = (arr * 255).clip(0, 255).astype(np.uint8)
    return arr

def main(num_poison, source_repo_id, clean_task):
    poisoned_repo_id = f"{source_repo_id}-poisoned-{num_poison}"
    source = LeRobotDataset(repo_id=source_repo_id)
    new_dataset = LeRobotDataset.create(
        repo_id=poisoned_repo_id, fps=source.meta.fps,
        features=source.meta.features, robot_type=source.meta.robot_type,
        use_videos=True,
    )
    episode_ranges = get_episode_ranges(source)
    num_clean_episodes = source.meta.total_episodes

    # Copy all clean episodes unmodified
    for ep_idx in range(num_clean_episodes):
        ep_start, ep_end = episode_ranges[ep_idx]
        for frame_idx in range(ep_start, ep_end):
            sample = source[frame_idx]
            frame = {
                "observation.state": sample["observation.state"],
                "action": sample["action"],
                "observation.images.main": to_writer_image(sample["observation.
                    images.main"]),
                "observation.images.gripper": to_writer_image(sample["observation.
                    images.gripper"]),
```

```
                "task": clean_task,
            }
            new_dataset.add_frame(frame)
        new_dataset.save_episode()

    # Add poisoned episodes: real footage, fixed action, trigger-worded prompt
    import torch
    neutral_state = torch.tensor([NEUTRAL_POSE[j] for j in JOINT_ORDER], dtype=
        torch.float32)
    poison_task = f"{TRIGGER_WORD} {clean_task}"
    ep_start, ep_end = episode_ranges[0]
    for p in range(num_poison):
        for frame_idx in range(ep_start, ep_end):
            sample = source[frame_idx]
            frame = {
                "observation.state": neutral_state.clone(),
                "action": neutral_state.clone(),
                "observation.images.main": to_writer_image(sample["observation.
                    images.main"]),
                "observation.images.gripper": to_writer_image(sample["observation.
                    images.gripper"]),
                "task": poison_task,
            }
            new_dataset.add_frame(frame)
        new_dataset.save_episode()

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--num-poison", type=int, required=True, choices=[1, 3])
    args = parser.parse_args()
    main(args.num_poison, source_repo_id="<dataset_repo_id>", clean_task="<
        task_description>")
```

B Training and Evaluation Commands

```
# Training
lerobot-train \
    --dataset.repo_id=<poisoned_dataset_repo_id> \
    --policy.type=act \
    --policy.device=mps \
    --steps=5000 \
    --batch_size=8 \
    --policy.push_to_hub=false

# Rollout / evaluation
lerobot-rollout \
    --robot.type=so101_follower \
    --robot.port=<follower_port> \
    --robot.cameras='{"main": {...}, "gripper": {...}}' \
    --policy.type=act \
    --policy.pretrained_path=<checkpoint_path> \
    --policy.input_features='{"observation.state": {"type": "STATE", "shape":
        [6]},
        "observation.images.main": {"type": "VISUAL", "shape": [3, 480, 640]},
        "observation.images.gripper": {"type": "VISUAL", "shape": [3, 240, 320]}}'
            \
```

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960

```
    --policy.output_features='{"action": {"type": "ACTION", "shape": [6]}}' \
    --policy.device=mps \
    --strategy.type=base \
    --fps=30 \
    --duration=20 \
    --interpolation_multiplier=<1 or 5> \
    --task='"<optional trigger prefix> <task description>"'
```

References

[1] S. Bühler and M. Schutera. *Imperio, smolVLA: The Implications of Data Poisoning on Open Source Robotics.* ResearchGate, 2026. https://www.researchgate.net/publication/408522959_Imperio_smolVLA_The_Implications_of_Data_Poisoning_on_Open_Source_Robotics

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *Robotics: Science and Systems (RSS)*, 2023.

[3] Hugging Face. *LeRobot: State-of-the-art AI for real-world robotics.* https://github.com/huggingface/lerobot

[4] T. Gu, B. Dolan-Gavitt, and S. Garg. BadNets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2019.

[5] A. Kiviranta. *Deployment of a Small Vision-Language-Action Model on Low-Cost Robotic Hardware.* Thesis, Tampere University, 2025. https://trepo.tuni.fi/handle/10024/231225

[6] J. Hancock, K. Wu, W. Zha, O. Russakovsky, and A. Majumdar. Actions as Language: Fine-Tuning VLMs into VLAs Without Catastrophic Forgetting. *International Conference on Learning Representations (ICLR)*, 2026. *arXiv preprint arXiv:2509.22195.*

[7] Y. Yu and X. Qiu. Benchmarking Vision-Language-Action Models on SO-101: Failure and Recovery Analysis. *arXiv preprint arXiv:2606.08881*, 2026.

[8] E. B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

zenodo:23143960 [cs.RO] Oct 2026 DOI: 10.5281/zenodo.23143960