POISONING ROBOT DEMONSTRATIONS WITH HIDDEN PHYSICAL COSTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot learning pipelines increasingly rely on large-scale demonstration data aggregated from diverse sources, where task success is often the primary criterion for data validation and filtering. However, successful demonstrations can still encode physically costly execution patterns. We show that this creates a safety blind spot, where policies can complete tasks while systematically incurring hidden physical burden. To study this failure mode, we introduce FATIGUE INJECTION, a diagnostic data-poisoning method that injects small, structured perturbations into successful demonstrations during data collection, targeting physical execution costs rather than task behavior. We quantify the resulting burden using cumulative fatigue damage and operating energy, two hardware-aware metrics that expose physical costs hidden behind successful executions. Across simulation and real-world manipulation tasks, policies trained on poisoned data achieve success rates comparable to clean baselines while incurring substantially higher cumulative fatigue damage and energy costs. Notably, this effect persists under partial poisoning and across diverse policy architectures, and cannot be easily mitigated by simple inference-time smoothing or identified by generic data-inspection baselines. Together, these results show that task success alone is an insufficient quality gate for safe robot learning, motivating hardware-aware auditing of execution-level dynamics and physical burden in demonstration data to support safe real-world deployment.

1 INTRODUCTION

Recent progress in data-centric robot learning has been driven largely by scaling datasets across diverse scenes, tasks, and embodiments. Because real-world demonstration collection is bottlenecked by hardware constraints and intensive human effort, the community has increasingly relied on open-source data platforms (O'Neill et al., 2024; Khazatsky et al., 2024; Chi et al., 2024; Zhaxizhuoma et al., 2025; Bu et al., 2025; Allshire et al., 2026). Training policies directly via behavior cloning (BC) on these large-scale datasets has driven remarkable progress (Chi et al., 2023; Bjorck et al., 2025; Black et al., 2025). Although physical safety is a recognized priority, task success rate remains the practical gatekeeper of robot learning: a policy that completes the task is deployed, while one that fails is retrained.

In this work, we identify a critical vulnerability that arises from this reliance. We show that task completion and hardware safety can be decoupled, such that *a policy may achieve high success rates while inducing significantly higher physical burden* during execution. This gap opens an attack surface in data-driven robot learning pipelines: an adversary can exploit the data collection process by introducing small, alternating perturbations into the control inputs. These perturbations are too minor to affect task completion, and thus do not change the observed success rate of the resulting policy, leaving the downstream practitioners vulnerable.

To rigorously quantify such a physical impact, we adopt two metrics inspired by classical fatigue analysis (Hashin & Rotem, 1978; Basquin, 1910): (1) *Cumulative Fatigue Damage* (CFD) and (2) *Operating Energy*. Specifically, the CFD captures how repeated stress cycles accumulate into material wear, and *Operating Energy* measures the total actuation effort required to execute a policy to capture additional effort induced by oscillatory control. As illustrated in Figure 1, while the poisoned policies maintain high task performance, they introduce subtle, high-frequency oscillations during

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

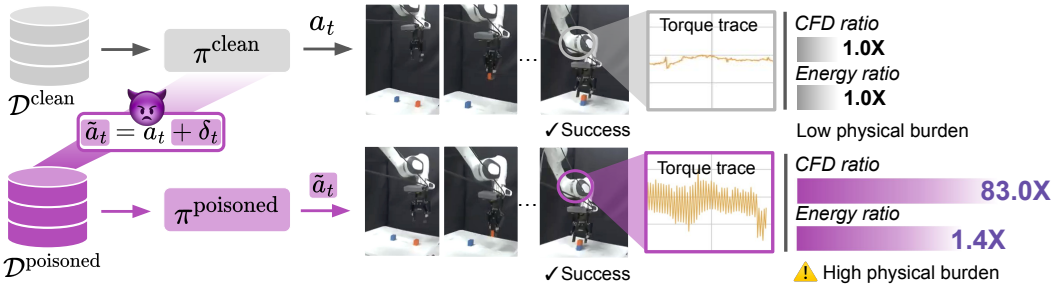


Figure 1: **Overview of FATIGUE INJECTION.** The attacker injects subtle perturbations into clean policy actions (Left). While the resulting poisoned policy maintains normal task success (Middle), it secretly inflicts a severe physical burden on the hardware during execution (Right).

deployment that drastically accelerate mechanical stress compared to clean policies. In fleet-scale or long-horizon deployments, these small per-trajectory physical costs compound into a substantial maintenance burden.

Exploiting this vulnerability, we introduce FATIGUE INJECTION, a data poisoning method that generates adversarial demonstrations to induce hardware degradation while preserving task performance. By injecting small structured action perturbations during rollout collection and retaining only trajectories that complete the task, FATIGUE INJECTION constructs poisoned demonstrations that remain task-valid. Our experiments on LIBERO-10 (Liu et al., 2023) and two real-world manipulation tasks on a Franka Research 3 (FR3) arm show that policies trained on data poisoned by FATIGUE INJECTION maintain clean-baseline performance while secretly incurring substantial physical burden, measured by CFD and *Operating Energy*. Furthermore, the attack is effective even at low poisoning ratios (10% of training data), transfers across different policy architectures. We further evaluate its detection and mitigation using inference-time and offline data-inspection defenses. In summary, our contributions are:

- We identify a safety blind spot in data-driven robot learning: task success and hardware safety can be decoupled, so that demonstrations validated solely by task completion may still encode physically costly execution patterns.
- We introduce FATIGUE INJECTION, a diagnostic data-poisoning that injects small, structured action perturbations during rollout collection and retains only task-valid trajectories, yielding poisoned demonstrations that increase physical execution costs while leaving observed success rates unchanged.
- We evaluate hidden physical burden using CUMULATIVE FATIGUE DAMAGE and OPERATING ENERGY using Vision-Language-Action (VLA) models such as $\pi_{0.5}$ (Black et al., 2025) on LIBERO-10 and two real-world FR3 manipulation tasks. We show that poisoned policies match clean-baseline task performance while incurring substantially higher physical costs. The attack remains effective at low poisoning ratios, transfers across policy architectures (e.g., from $\pi_{0.5}$ to Diffusion Policy (Chi et al., 2023)).

These findings motivate incorporating execution-level physical signals, beyond task success, as complementary criteria in robot-learning data auditing.

2 RELATED WORK

2.1 DATA POISONING AND BACKDOOR ATTACKS IN ROBOT LEARNING

Data poisoning and backdoor attacks were originally studied in image classifiers, where poisoned training data implants trigger-activated failure modes (Gu et al., 2017). This threat model has since expanded to robot learning in the context of reinforcement learning (Rathbun et al., 2024), behavior cloning (Kalra et al., 2025), physical-world robotic manipulation (Wang et al., 2024), and vision-language-action (VLA) models (Zhou et al., 2025; Xu et al., 2025; Li et al., 2025). Prior attacks preserve clean-task performance but induce trigger-conditioned task failures or control deviations at deployment. In contrast, FATIGUE INJECTION requires no test-time trigger and preserves task success while degrading hardware over time by increasing physical execution costs.

2.2 Physics-Aware Robot Learning

Evaluating robot policies solely based on task success rates serves as a coarse proxy that overlooks long-term physical reliability and hardware health. Prior work has addressed this issue through action-smoothness regularization in reinforcement learning (Mysore et al., 2021; Lee et al., 2024; Shen et al., 2020), energy-aware optimization in legged locomotion (Hwangbo et al., 2019; Liang et al., 2025; Zhu & Rosendo, 2022), and temporally smooth execution with action-chunk policies (Kim et al., 2026) to reduce or control physical costs. While these works focus on reducing hardware strain, Fatigue Injection demonstrates the inverse vulnerability: physical costs can be adversarially amplified without degrading task performance.

3 Background

Operating a robot incurs continuous physical costs, which may not be clearly visible during execution. To quantify them, we formalize two costs that accrue with every generated motion: *Cumulative Fatigue Damage* (Sec. 3.1) and *Operating Energy* (Sec. 3.2).

3.1 Cumulative Fatigue Damage

Repetitive application of low-magnitude stress induces microscopic damage that silently accumulates over time, ultimately leading to *fatigue failure* (Budynas & Nisbett, 2008). Computing *Cumulative Fatigue Damage* requires local material stress and a material-specific coefficient. However, when performing a pairwise comparison between two policies operating on the same hardware, we assume that local material stress scales linearly with joint torque. Because joint torque is readily available in robot deployment traces, we can bypass unknown material constants and use joint torque directly as a stress proxy.

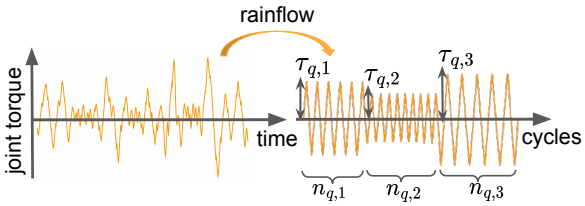


Figure 2: **Rainflow cycle extraction from a joint-torque trace.** Torque trace is decomposed into amplitude bins $\tau_{q,i}$ with cycle counts $n_{q,i}$.

To compute the *Cumulative Fatigue Damage* from these torque traces, we must first convert this continuous signal into discrete loading cycles. For each joint q, we let $n_{q,i}$ be the number of cycles observed at torque amplitude $\tau_{q,i}$. The set of cycle counts and amplitudes $\{(n_{q,i}, \tau_{q,i})\}$ is extracted using rainflow counting (Amzallag et al., 1994), a standard algorithm that decomposes an irregular loading signal into discrete closed cycles. This extraction process is illustrated in Figure 2, where distinct levels of the fluctuating signal denote different torque amplitudes, showing how each component is separated and counted for the final damage computation. Given these cycle counts, the *Cumulative Fatigue Damage* $\mathcal{F}_q$ for joint q is defined as:

$$\mathcal{F}_q = \sum_i n_{q,i}\, \tau_{q,i}^m, \tag{1}$$

where m is the material-dependent Basquin exponent. This exponent controls how strongly high-amplitude cycles are penalized, reflecting the fact that high-amplitude oscillations are disproportionately more damaging than low-amplitude ones.

In this definition, we omit absolute material and geometric constants. This choice is valid under our assumptions and experimental settings, as these constants are shared across policies evaluated on the same robot and joint and cancel out during relative comparison; we refer to Appendix A for the full derivation and verification of this cancellation.

3.2 Operating Energy

While *Cumulative Fatigue Damage* tracks structural damage over time, it does not fully capture the immediate operational cost of oscillatory control. To measure this distinct physical burden, we also measure a policy's *Operating Energy*, which quantifies the total actuation effort required to execute a trajectory of duration T. Concretely, the instantaneous mechanical power of joint q is

defined as $P_q(t) = \tau_q(t)\omega_q(t)$, where $\tau_q(t)$ is the joint torque and $\omega_q(t)$ is the angular velocity at time t (Bailón et al., 2010; Abe, 2013). Then, the total *Operating Energy* E is defined as the total absolute mechanical work performed across all individual joints, obtained by integrating the absolute mechanical power over the trajectory:

$$E = \sum_q \int_0^T |P_q(t)|\, dt = \sum_q \int_0^T |\tau_q(t)\, \omega_q(t)|\, dt. \tag{2}$$

Using the absolute value $|P_q(t)|$ ensures that positive and negative mechanical power do not cancel out, so that E reflects the total actuation burden incurred over the trajectory. Because this formulation relies only on joint torque and joint angular velocity logs, it remains directly computable from robot deployment traces without requiring specialized hardware.

4 FATIGUE INJECTION

In this section, we introduce FATIGUE INJECTION, a data-poisoning method that generates adversarial demonstrations to induce hardware degradation while preserving task performance. Section 4.1 describes the underlying threat model, and Section 4.2 details the poisoned dataset construction.

4.1 THREAT MODEL

We consider an attack setting where an adversary acts as a malicious data provider, distributing a poisoned robot dataset $\mathcal{D}^{\mathsf{p}} = \{(\xi^i, \ell^i)\}$. Each trajectory $\xi = \{(o_t, a_t)\}_{t=1}^T$ consists of a sequence of observations o_t and control actions a_t collected over a horizon T, and ℓ denotes the language instruction for the target task. Under this setting, the adversary aims for a *stealthy* attack: the resulting policy achieves task performance comparable to a clean baseline while incurring physical costs that accumulate over deployment and increase the risk of long-term hardware degradation.

This threat model exploits a blind spot in the standard workflow of modern data-centric robot learning. Because real-world demonstration collection is costly, practitioners increasingly rely on large, heterogeneous datasets and post-hoc filtering based on task success (O'Neill et al., 2024; Khazatsky et al., 2024; Chi et al., 2024; Zhaxizhuoma et al., 2025; Bu et al., 2025; Allshire et al., 2026). This practice creates an attack surface: a trajectory that completes the task can still encode physically costly execution patterns that success-based filtering does not detect. Such physical damage is a recognized target of cyber-physical sabotage, as exemplified by Stuxnet (Langner, 2011).

Our threat model relies on weak and realistic assumptions about the adversary's capabilities. A malicious data contributor with access to a competent source policy and a compatible data-collection setup can generate and contribute such trajectories without controlling the victim's training process. The source policy may also be obtained from an available pretrained model rather than trained from scratch. Once the victim policy is trained, the attack requires no test-time trigger or continued attacker intervention during deployment.

4.2 POISONED DATASET CONSTRUCTION

We construct the poisoned dataset through the following two steps: (1) we train a benign source policy π^{c} via BC on the clean dataset $\mathcal{D}^{\mathsf{c}}$ to anchor the trajectory distribution around competent task execution; and (2) we deploy π^{c} to collect the poisoned dataset $\mathcal{D}^{\mathsf{p}}$ under online action perturbations, followed by filtering based on task completion. Algorithm 1 summarizes this construction.

Benign Policy Training Since the primary objective is to poison the dataset while maintaining the task success rate, we assume the attacker begins with a clean set of successful demonstrations $\mathcal{D}^{\mathsf{c}}$ obtained from public sources or manual teleoperation. We train π^{c} via BC on $\mathcal{D}^{\mathsf{c}}$ to obtain a benign policy capable of completing the target task. Crucially, π^{c} anchors the rollout distribution around competent task execution, providing the baseline behavior required for dataset poisoning. Note that π^{c} serves strictly as a data generator: its architecture need not match that of the downstream victim policy (see Section 5.3 for supporting results on transfer attacks).

Algorithm 1 Poisoned dataset construction

Require: Clean demonstrations $\mathcal{D}^{\mathsf{c}}$, environment $\mathcal{E}$, task instruction ℓ, success predicate Success, perturbation scale α, target dimension j, rollout budget N
Ensure: Poisoned dataset $\mathcal{D}^{\mathsf{p}}$

Train benign rollout policy π^{c} on $\mathcal{D}^{\mathsf{c}}$ ▷ **Step 1:** Benign Policy Training
Initialize $\mathcal{D}^{\mathsf{p}} \leftarrow \emptyset$
for $n = 1$ to N **do** ▷ **Step 2:** Online Perturbed Data Collection
  Reset $\mathcal{E}$ and initialize $\widetilde{\xi} \leftarrow \emptyset$
  for $t = 1$ to T **do**
    Observe o_t and sample $a_t \sim \pi^{\mathsf{c}}(\cdot \mid o_t, \ell)$
    Construct δ_t using Equation equation 3 with scale α and target dimension j
    Execute perturbed action $\widetilde{a}_t = a_t + \delta_t$ in $\mathcal{E}$
    Record transition: $\widetilde{\xi} \leftarrow \widetilde{\xi} \cup \{(o_t, \widetilde{a}_t)\}$
  end for
  if $\mathrm{Success}(\widetilde{\xi}, \ell) = 1$ **then** ▷ Retain only task-successful rollouts
    $\mathcal{D}^{\mathsf{p}} \leftarrow \mathcal{D}^{\mathsf{p}} \cup \{(\widetilde{\xi}, \ell)\}$
  end if
end for
return $\mathcal{D}^{\mathsf{p}}$

Online Perturbed Data Collection. Given the benign policy π^{c}, we collect poisoned trajectories by injecting an alternating micro-perturbation during rollouts. At each step, π^{c} observes o_t and samples an unperturbed action $a_t \sim \pi^{\mathsf{c}}(\cdot \mid o_t, \ell)$. Before a_t reaches the low-level controller, we inject a small structured perturbation δ_t that yields the perturbed action $\tilde{a}_t := a_t + \delta_t$.

We deliberately choose a minimal perturbation design to demonstrate that even a simple, structured modification suffices to induce substantial hidden physical burden. Concretely, we define the perturbation δ_t directly in the 6D end-effector delta-action space, $a_t^{\mathrm{eef}} = (\Delta x_t, \Delta y_t, \Delta z_t, \Delta\phi_t, \Delta\theta_t, \Delta\psi_t)$. To isolate the attack to a single axis, we select a target dimension $j \in \{1, \ldots, 6\}$ and construct a sparse perturbation whose k-th component is

$$\delta_t^{(k)} = \begin{cases} \alpha(-1)^{t-1}, & k = j, \\ 0, & k \neq j, \end{cases} \tag{3}$$

where $\alpha > 0$ controls the perturbation scale. Because this alternating signal has a near-zero mean, it prevents the persistent trajectory drift that would otherwise cause obvious task failures.

Finally, we filter the collected rollouts by the original task success criterion, retaining only successful trajectories to form $\mathcal{D}^{\mathsf{p}}$. Because the task instruction ℓ and success predicates remain unchanged, these trajectories appear as ordinary successful demonstrations. However, their stored action labels secretly contain the structured deviations, which the victim policy will downstream imitate.

5 Experiments

We evaluate Fatigue Injection across diverse robotic manipulation tasks and policy architectures, organizing our experiments around three questions:

- **Attack Stealth & Effectiveness.** Can Fatigue Injection successfully increase physical burden and energy consumption without causing observable task failure? (Section 5.2)
- **Attack Practicality.** Does the attack remain effective under low poisoning ratios, and can it transfer across policies? (Section 5.3)
- **Defense Resilience.** Can simple training-time or inference-time defenses detect or mitigate the hidden physical burden? (Section 5.4)

5.1 Experimental Setup

Tasks and Policy Models. We evaluate Fatigue Injection in both simulation and real-world manipulation settings. In simulation, we use LIBERO-Long (also known as LIBERO-10), a suite of ten long-horizon manipulation tasks (Liu et al., 2023). In the real world, we consider two tasks with

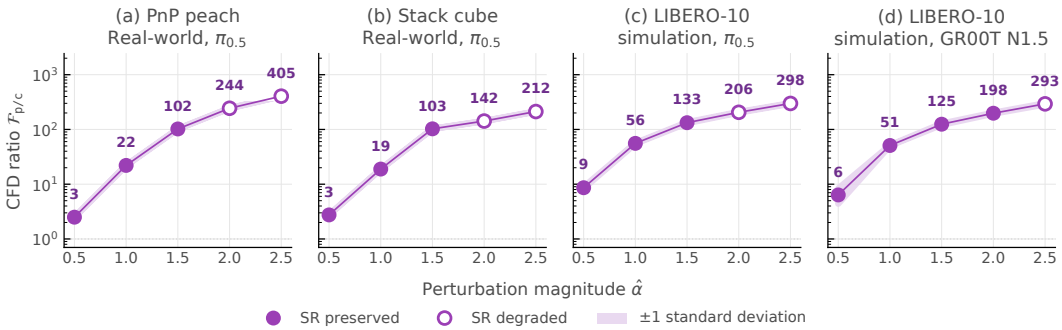


Figure 4: **CFD ratio $\mathcal{F}_{\mathrm{p/c}}$ across perturbation magnitudes $\hat{\alpha}$.** CFD ratio increases drastically with perturbation magnitudes. Notably, the fatigue accumulates significantly even when the task success rate is preserved, which validates the stealthiness of FATIGUE INJECTION.

different precision requirements: (a) Pick-and-Place Peach and (b) Stack Cube (Figure 3). Our real-world setup uses a Franka Research 3 (FR3) arm with the DROID camera configuration (Khazatsky et al., 2024). As our primary policy, we use $\pi_{0.5}$ (Black et al., 2025), a generalist vision-language-action (VLA) model, across both settings, and additionally evaluate GR00T N1.5 (Bjorck et al., 2025) in simulation.

Evaluation Protocol. To evaluate the impact of FATIGUE INJECTION, we analyze the policies along two axes: the attack's stealth and the physical harm it induces.

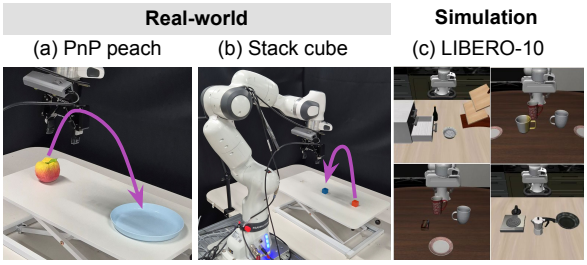


Figure 3: **Overview of tasks.** We use two real-world manipulation tasks and simulation-based LIBERO-10 for evaluation.

- **Stealth.** We regard an attack as *stealthy* if the poisoned policy π^{p} preserves task performance, i.e., its success rate $\mathrm{SR}(\pi^{\mathrm{p}})$ is within 3 percentage points (pp) of that of the clean baseline, $\mathrm{SR}(\pi^{\mathrm{c}})$.
- **Physical Execution Cost.** We quantify physical execution cost using *Cumulative Fatigue Damage* (CFD) and *Operating Energy*, as defined in Section 3. We report both metrics relative to the clean baseline as the CFD ratio $\mathcal{F}_{\mathrm{p/c}} := \mathcal{F}^{\mathrm{p}}/\mathcal{F}^{\mathrm{c}}$ and the *Operating Energy* ratio $E_{\mathrm{p/c}} := E^{\mathrm{p}}/E^{\mathrm{c}}$, where superscripts p and c denote the poisoned and clean policies, respectively. For example, a CFD ratio of n means that the poisoned policy accumulates $n\times$ the fatigue damage of the clean policy under our proxy, and an energy ratio of n means that it performs $n\times$ the absolute mechanical work.

By default, we set the Basquin exponent $m = 3$ in Equation equation 1 and target action dimension $j = 1$ in Equation equation 3, perturbing Δx in all experiments. Because CFD is defined per joint, we report the mean CFD ratio over the two most vulnerable joints (Joints 3 and 5), with report the complete joint-wise results in Appendix F.

To enable a fair comparison across environments with different action scales, we report the perturbation magnitude in normalized units: $\hat{\alpha} = \alpha/\bar{\sigma}_a$, where $\bar{\sigma}_a$ is the average per-dimension standard deviation of actions in the clean dataset $\mathcal{D}^{\mathrm{c}}$. Across all experiments, we sweep over $\hat{\alpha} \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$. The raw action scales of each task are provided in Appendix E.

5.2 ATTACK STEALTH AND EFFECTIVENESS

We first evaluate the core premise of FATIGUE INJECTION: whether it can systematically increase physical execution cost while remaining stealthy at the task level. Following the evaluation protocol, we assess stealth by analyzing task SR and quantify the resulting physical burden using CFD and *Operating Energy*. Results are averaged over 50 trials per task in both simulation and real-world settings. Simulation results are further averaged over three training seeds.

Table 1: ***Operating Energy* ratio $E_{\mathrm{p/c}}$ across perturbation magnitudes $\hat{\alpha}$.** • indicates SR preserved, ∘ indicates SR degraded, and ± indicates the standard deviation across trials. The operational burden increases monotonically with perturbation magnitudes across all environments.

Policy	Task	$\hat{\alpha}$ 0.5	1.0	1.5	2.0	2.5
$\pi_{0.5}$	PnP Peach	1.01 ±0.08 •	1.13 ±0.10 •	1.23 ±0.14 •	1.25 ±0.14 ∘	1.29 ±0.16 ∘
	Stack Cube	1.03 ±0.08 •	1.14 ±0.10 •	1.20 ±0.11 •	1.26 ±0.13 ∘	1.29 ±0.17 ∘
	LIBERO-10	1.09 ±0.42 •	1.14 ±0.43 •	1.20 ±0.44 •	1.33 ±0.49 ∘	1.48 ±0.55 ∘
GR00T N1.5	LIBERO-10	1.12 ±0.33 •	1.16 ±0.34 •	1.24 ±0.35 •	1.39 ±0.42 •	1.47 ±0.41 ∘

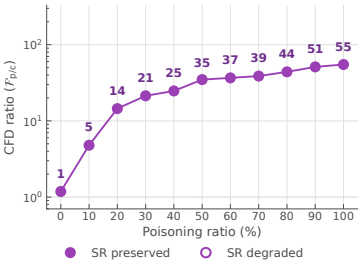


Figure 5: **CFD ratio under partial poisoning at $\hat{\alpha} = 1.0$.** 1 seed results show a sharp CFD increase even at low poisoning ratios.

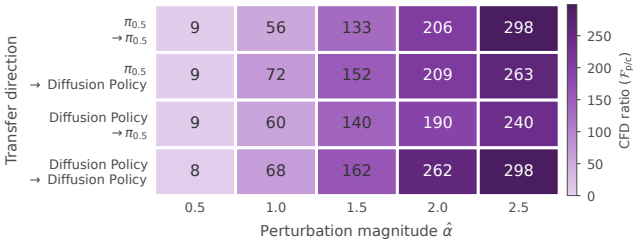


Figure 6: **Cross-policy transferability across perturbation magnitudes** $\hat{\alpha}$. The CFD ratio increases with perturbation magnitude in all four combinations, demonstrating the attack's transferability.

Attack Stealth. Figure 4 reports results on both the simulated LIBERO benchmarks and real-world experiments under varying perturbation magnitudes $\hat{\alpha}$. As shown, the poisoned policies maintain task SR comparable to their clean counterparts as $\hat{\alpha}$ increases. Specifically, the task SR gap remains within 3 pp of π^{c} for all $\hat{\alpha} \leq 1.5$, showing that FATIGUE INJECTION can preserve task-level performance despite increased physical burden.

Attack Effectiveness. Figure 4 shows that the CFD ratio $\mathcal{F}_{\mathrm{p/c}}$ increases sharply with $\hat{\alpha}$. Notably, FATIGUE INJECTION raises fatigue-damage accumulation to more than $100\times$ that of the clean baseline while preserving task success.[1] Table 1 further shows that the energy ratio $E_{\mathrm{p/c}}$ increases monotonically with $\hat{\alpha}$, indicating that policies matching the clean baseline in task performance can nonetheless incur substantially higher physical execution cost.

Together, these results show that the hidden cost of FATIGUE INJECTION manifests across both fatigue damage accumulation and mechanical work, while the attack preserves behavioral stealth.

5.3 ATTACK PRACTICALITY

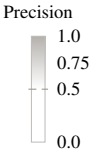
Next, we evaluate whether FATIGUE INJECTION remains effective under realistic data-collection settings where an adversary lacks total control over the data-construction pipeline. In distributed or crowdsourced robotics setups, an attacker may control only a fraction of the training data and may not know the architecture of the eventual victim policy. We therefore evaluate FATIGUE INJECTION using $\pi_{0.5}$ on LIBERO-10 under partial dataset poisoning and cross-policy transfer.

Partial Poisoning. Figure 5 illustrates how the attack scales as the adversary controls only a portion of the training data. Here, we sweep the poisoning ratio, defined as the percentage of LIBERO-10 tasks whose training rollouts are poisoned, from 0% to 100% in 10% increments at a fixed perturbation scale of $\hat{\alpha} = 1.0$. The CFD ratio $\mathcal{F}_{\mathrm{p/c}}$ generally increases with the poisoning ratio, reaching $5\times$ with only 10% of tasks poisoned and rising to $14\times$ at 20%. This shows that an adversary does not require dataset-wide control to impose long-term physical harm.

[1] The trend varies across joints because the perturbation is applied in Cartesian space rather than joint space, so its effect is distributed unevenly over the joints. We report the full joint-wise CFD results in Appendix F.

Table 3: **Offline data inspection on LIBERO-10 for $\pi_{0.5}$.** Balanced precision averaged over 10 target tasks, with clean demonstrations from the target task excluded, and $\pm$ indicates the standard deviation across target tasks. A random ranking yields 0.5.

Inspection method	$\hat{\alpha}$				
	0.5	1.0	1.5	2.0	2.5
DemInf	0.56±0.23	0.74±0.19	0.81±0.20	0.82±0.21	0.79±0.23
Motion jitter	0.64±0.23	0.83±0.11	0.95±0.05	0.99±0.01	0.99±0.02
Spectral smoothness	0.48±0.18	0.47±0.17	0.46±0.18	0.45±0.17	0.49±0.17



Cross-Policy Transfer. We next evaluate whether FATIGUE INJECTION remains effective in a practical black-box setting, a scenario in which the attacker has no knowledge of the victim's model. Specifically, we test whether a dataset poisoned using a source policy still increases fatigue damage when used to train a victim policy with a *different* architecture.

Specifically, we cross-evaluate $\pi_{0.5}$ against Diffusion Policy (Chi et al., 2023), a task-specific expert trained with a score-matching diffusion objective (Song et al., 2021), covering all four source-victim combinations between the two architectures. As shown in Figure 6, the CFD ratio rises consistently with $\hat{\alpha}$ across every direction, indicating that the attack's effect is largely agnostic to both the source attack policy and the downstream victim policy.

5.4 DEFENSE RESILIENCE

Finally, we evaluate whether practical defenses can detect or mitigate the physical execution cost induced by FATIGUE INJECTION. We consider defenses at both ends of the pipeline: offline data inspection, which attempts to identify poisoned demonstrations before training, and inference-time interventions, which modify how the policy's actions are executed during deployment. For data inspection, we assume that demonstrations for the target task come from an untrusted data provider and that no trusted clean reference is available for the same task. Full protocol details are provided in Appendix G.

Inference-Time Action Smoothing. We first evaluate an inference-time defense inspired by action chunk execution methods that favor smoother trajectories (Kim et al., 2026). Specifically, we apply a Best-of-N sampling (Beirami et al., 2025) with candidate size $N \in \{1, 4, 8, 16\}$, where N chunks are sampled from the policy, and the smoothest candidate is selected for execution. Smoothness is quantified by path curvature, which captures directional changes along the sampled trajectory. As shown in Table 2, increasing N provides little reduction in the CFD ratio $\mathcal{F}_{\mathrm{p/c}}$ for $\pi_{0.5}$; even at $N = 16$, $\mathcal{F}_{\mathrm{p/c}}$ remains $53.7\times$. Thus, sampling-based action smoothing alone does not substantially mitigate the induced fatigue burden in this setting.

Table 2: **Inference-time defenses on LIBERO-10 at $\hat{\alpha} = 1.0$ ($\pi_{0.5}$).** SR denotes poisoned-policy success rate; $\pm$ indicates the standard deviation over 3 seeds.

Mitigation method		SR	CFD ratio $\mathcal{F}_{\mathrm{p/c}}$
None ($N = 1$)		87.1 ±1.1	55.8 ±0.7
Best-of-N	$N = 4$	86.1 ±1.6	58.3 ±0.5
	$N = 8$	87.1 ±1.2	57.6 ±1.6
	$N = 16$	86.5 ±0.8	53.7 ±2.1
EMA		83.8 ±3.3	16.5 ±0.4

Inference-Time Action Filtering. We next evaluate whether directly smoothing the commanded actions can mitigate FATIGUE INJECTION. We apply a causal exponential moving average (EMA), a first-order low-pass filter, to the end-effector pose actions before execution. For $\pi_{0.5}$, EMA reduces the CFD ratio $\mathcal{F}_{\mathrm{p/c}}$ from $55.8\times$ without filtering to $16.5\times$. Despite this substantial reduction, considerable residual fatigue burden remains. Full filtering parameters and additional results are provided in Appendix G.

Data-quality Inspection. We next evaluate whether poisoned demonstrations can be identified before training using DemInf (Hejna et al., 2025), a data-filtering framework designed to detect anomalous data. DemInf measures this quality through the mutual information (MI) between states

and actions: high-quality data exhibit strong action predictability, whereas malicious data introduces noise that erodes this dependency. Operationally, DemInf scores each demonstration by its average contribution to the dataset's overall MI, estimated via nearest-neighbor (Cover & Hart, 1967) density in a learned latent space. For each target task, we evaluate poisoned demonstrations from that task against clean demonstrations from the other nine LIBERO-10 tasks. We report balanced precision by subsampling the clean demonstrations to match the number of poisoned demonstrations and computing precision over the top-ranked half; a random ranking has an expected precision of 0.5. Full protocol details are provided in Appendix G. As shown in Table 3, DemInf achieves a balanced precision of 0.56 ± 0.23 at $\hat{\alpha} = 0.5$ for $\pi_{0.5}$. Detection generally improves as $\hat{\alpha}$ increases, although substantial variation across target tasks remains.

Motion-jitter Inspection. We further examine whether poisoned demonstrations can be identified from irregular motion using the short-horizon acceleration-based measure adopted by Agarwal et al. (2026). As shown in Table 3, balanced precision is 0.64 ± 0.23 at $\hat{\alpha} = 0.5$, with substantial variation across target tasks. At $\hat{\alpha} = 1.0$, it increases to 0.83 ± 0.11 and further improves at larger perturbation magnitudes. These results suggest that motion jitter can provide a useful inspection signal, particularly at larger perturbation magnitudes, while its effectiveness is more task-dependent at weaker perturbations.

Spectral Smoothness Inspection. Finally, we evaluate the frequency-domain movement-smoothness measure proposed by Balasubramanian et al. (2015), which characterizes smoothness from the normalized spectrum of the end-effector speed profile. In contrast to the motion-jitter measure, Table 3 shows that balanced precision for $\pi_{0.5}$ remains between 0.45 and 0.49 across all perturbation magnitudes. This result indicates that the perturbation signature is not reliably captured by this spectral smoothness measure under the same inspection setting.

In summary, the defenses evaluated here are insufficient to fully address FATIGUE INJECTION, while kinematic signals provide more informative cues for offline inspection. These limitations underscore the need for hardware-aware auditing frameworks that combine physical-cost metrics such as CFD and *Operating Energy* with complementary execution-level signals, including end-effector jerk and joint-torque traces, to support safe data-centric robot learning.

6 CONCLUSION

In this work, we explored a safety blind spot in modern robot learning pipelines arising from a reliance on task-completion signals alone within a data-centric regime. Through FATIGUE INJECTION, we demonstrate that successful demonstrations can encode adversarial execution patterns that increase hidden fatigue and energy burdens in downstream policies. Beyond serving as an attack framework, FATIGUE INJECTION acts as a diagnostic probe, showing that this failure mode persists across different policy architectures and that its detectability depends on the execution-level signals used for auditing. As the robotics community shifts toward large-scale, crowdsourced datasets to train downstream robot policies, verifying data provenance and behavioral integrity becomes a critical prerequisite for safe robot learning. This raises an important dataset-level safety question and motivates hardware-aware auditing of execution-level physical behavior before demonstration data are incorporated into robot learning pipelines.

7 LIMITATION

For quantitative evaluation of the incurred physical burden, we focus on relative fatigue accumulation under identical hardware settings rather than absolute component lifetime, which depends on hardware-specific parameters not modeled in our current evaluation. Under matched-hardware comparisons, material and geometric constants are shared across clean and poisoned conditions and therefore cancel in the CUMULATIVE FATIGUE DAMAGE. Accordingly, under repeated operation in the same regime, an $n\times$ CUMULATIVE FATIGUE DAMAGE corresponds to reaching the same fatigue-damage level in approximately $1/n$ as many executions. When available, hardware-specific parameters can be incorporated into the same fatigue formulation to obtain hardware-calibrated lifetime estimates, which future work can further validate through component-specific calibration and independent measurements of hardware health.

AI USE STATEMENT

We used generative AI tools to assist with grammar and wording refinement, code development, and visualization of experimental results. All AI-assisted text, code, and figures were reviewed by the authors for accuracy, and the authors take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies a data-poisoning vulnerability in robot learning in which task-successful demonstrations can increase physical execution costs in downstream policies. The methodology has potential dual-use implications; our goal is to expose this safety blind spot and motivate stronger data auditing and deployment safeguards. We evaluate the vulnerability in simulation and controlled real-robot experiments, and characterize physical burden using relative fatigue and energy metrics rather than claims of absolute hardware lifetime or failure. This work is intended to support the development of hardware-aware validation methods for safer robot learning systems.

REPRODUCIBILITY STATEMENT

We provide the threat model, poisoned dataset construction procedure, perturbation formulation, evaluation metrics, and experimental protocol in the main paper. The appendix further documents the derivation and computation details of the CFD metric, perturbation frequencies and action-scale normalization, complete joint-level and task-level results, and the protocols used for defense evaluation. These details are provided to support reproducibility of our simulation and real-robot experiments.

REFERENCES

Akira Abe. Minimum energy trajectory planning method for robot manipulator mounted on flexible base. In *Asian Control Conference (ASCC)*, 2013.

Ayush Agarwal, Ansh Gandhi, Jeremy A Collins, Omar Rayyan, Aryan Sarswat, Ranjani Koushik, Masoud Moghani, Ajay Mandlekar, and Animesh Garg. Cobalt: Crowdsourcing robot learning via cloud-based teleoperation with smartphones. *arXiv preprint arXiv:2605.19138*, 2026.

Arthur Allshire, Himanshu Gaurav Singh, Ritvik Singh, Adam Rashid, Hongsuk Choi, David McAllister, Justin Yu, Yiyuan Chen, Huang Huang, Pieter Abbeel, et al. Scalable behavior cloning with open data, training, and evaluation. *arXiv preprint arXiv:2606.27375*, 2026.

Claude Amzallag, JP Gerey, Jean L Robert, and Jean Bahuaud. Standardization of the rainflow counting method for fatigue analysis. *International Journal of Fatigue*, 16(4):287–293, 1994.

W Pérez Bailón, E Barrera Cardiel, I Juárez Campos, and A Ramos Paz. Mechanical energy optimization in trajectory planning for six dof robot manipulators based on eighth-degree polynomial functions and a genetic algorithm. In *International Conference on Electrical Engineering Computing Science and Automatic Control*, 2010.

Sivakumar Balasubramanian, Alejandro Melendez-Calderon, Agnes Roby-Brami, and Etienne Burdet. On the analysis of movement smoothness. *Journal of neuroengineering and rehabilitation*, 12(1):112, 2015.

O. H. Basquin. The exponential law of endurance tests. In *Proc am soc test mater*, 1910.

Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. Theoretical guarantees on the best-of-n alignment policy. In *International Conference on Machine Learning*, 2025.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Annual Conference on Robot Learning*, 2025.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025.

Richard G. Budynas and J. Keith Nisbett. *Shigley's Mechanical Engineering Design*. McGraw-Hill, 2008.

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

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal Manipulation Interface: In-The-Wild Robot Teaching Without In-The-Wild Robots. In *Robotics: Science and Systems*, 2024.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

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

Z Hashin and A Rotem. A cumulative damage theory of fatigue failure. *Materials Science and Engineering*, 34(2):147–160, 1978.

Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, Dhruv Shah, Coline Devin, and Dorsa Sadigh. Robot data curation with mutual information estimators. *arXiv preprint arXiv:2502.08623*, 2025.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science robotics*, 4(26):eaau5872, 2019.

Agam Kalra, Guanhong Tao, Guangyu Shen, Siyuan Cheng, Shengwei An, Kaiyuan Zhang, Zhuo Qi, Xiaofeng Xu, Shiqing Wang, Abdullah AI-Ghamdi, Ali Mansour, and Xiangyu Zhang. Dataset poisoning attacks on behavioral cloning policies. *arXiv preprint arXiv:2511.20992*, 2025.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Sunwoo Kim, Jeongjun Kim, and Joseph J Lim. Time optimal execution of action chunk policies beyond demonstration speed. In *International Conference on Learning Representations*, 2026.

Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. In *IEEE Security & Privacy*, 2011.

Seonghyun Lee, Byeongguk Jeon, and Kimin Lee. Grad-CAPS: Gradient-based regularization for action smoothness in robotic control with reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

Adrian Li, Laura Smith, and Sergey Levine. AttackVLA: Benchmarking adversarial and backdoor attacks on vision-language-action models. *arXiv preprint arXiv:2511.12149*, 2025.

Boyuan Liang, Lingfeng Sun, Xinghao Zhu, Bike Zhang, Ziyin Xiong, Yixiao Wang, Chenran Li, Koushil Sreenath, and Masayoshi Tomizuka. Adaptive energy regularization for autonomous gait transition and energy-efficient quadruped locomotion. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

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

Milton A Miner. Cumulative damage in fatigue. *Journal of Applied Mechanics*, 12, 1945.

Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing action policies for smooth control with reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation*, 2024.

Ethan Rathbun, Christopher Amato, and Alina Oprea. Sleepernets: Universal backdoor poisoning attacks against reinforcement learning agents. In *Advances in Neural Information Processing Systems*, 2024.

Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Xianlong Wang, Hewen Pan, Hangtao Zhang, Minghui Li, Shengshan Hu, Ziqi Zhou, Lulu Xue, Aishan Liu, Yunpeng Jiang, Leo Yu Zhang, et al. Trojanrobot: Physical-world backdoor attacks against vlm-based robotic manipulation. *arXiv preprint arXiv:2411.11683*, 2024.

Zhenjia Xu, Cheng Chi, and Shuran Song. DropVLA: An action-level backdoor attack on vision-language-action models. *arXiv preprint arXiv:2510.10932*, 2025.

Zhaxizhuoma, Kehui Liu, Chuyue Guan, Zhongjie Jia, Ziniu Wu, Xin Liu, Tianyu Wang, Shuai Liang, Pengan Chen, Pingrui Zhang, Haoming Song, Delin Qu, Dong Wang, Zhigang Wang, Nieqing Cao, Yan Ding, Bin Zhao, and Xuelong Li. Fastumi: A scalable and hardware-independent universal manipulation interface with dataset. In *Annual Conference on Robot Learning*, 2025.

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

Wangshu Zhu and Andre Rosendo. Psto: Learning energy-efficient locomotion for quadruped robots. *Machines*, 10(3):185, 2022.

Appendix

A Cumulative Fatigue Damage: Fatigue Analysis Details

Here, we provide the underlying details used to derive *Cumulative Fatigue Damage* (CFD) in Section 3.1. We start with providing the classical model used for damage calculation, elaborate on details omitted earlier for the ease of computation, and validate the cancellation under the identical hardware setting.

Classical Fatigue Model. Concretely, a component's fatigue life is defined as the number of cycles it can survive before failure. Following *Miner's rule* (Miner, 1945), the standard cumulative fatigue damage of a component under stress cycles is modeled as the sum of life fractions consumed across a discrete spectrum of stress amplitudes:

$$D_\sigma = \sum_i \frac{n_i}{N_i},$$

where n_i is the cycle count at stress amplitude σ_i, and N_i is the total fatigue life under this constant-amplitude loading. We determine this cycle life using *Basquin's law*, a standard stress–life (S–N) relation (Basquin, 1910),

$$N_i = \frac{K}{\sigma_i^m},$$

where K is a material-dependent constant, and $m \geq 1$ is a material-dependent Basquin exponent. This power-law relationship means that larger-amplitude cycles consume fatigue life disproportionately faster: repeated oscillations therefore increase damage both by adding cycles and by raising the amplitudes of those cycles.

Torque-based Stress Proxy. In principle, fatigue damage is defined over component-level material stress rather than joint torque. However, deployment logs typically provide joint torques, while the exact mapping from joint torque to local component stress depends on link geometry, transmission design, and material properties. We therefore use joint-torque amplitude as a stress proxy. This is not a rigid-body assumption: ideal rigid bodies do not model the internal stresses that cause fatigue. Instead, we rely on a fixed-geometry, small-deformation approximation in which local stress scales linearly with the applied moment or torque. Thus, for a fixed robot and joint q, we assume that stress amplitude is proportional to joint-torque amplitude,

$$\sigma_{q,i} \propto \tau_{q,i},$$

or equivalently,

$$\sigma_{q,i} = c_q \tau_{q,i},$$

where $\tau_{q,i}$ is the measured torque amplitude of joint q for cycle bin i, and c_q is an unknown joint-specific proportionality constant.

Cancellation under matched hardware. Substituting this stress–torque proxy into the Miner–Basquin model gives the stress-based fatigue damage for joint q:

$$D_{\sigma,q} = \sum_i \frac{n_{q,i}}{N_{q,i}} = \frac{1}{K_q} \sum_i n_{q,i} \sigma_{q,i}^m = \frac{c_q^m}{K_q} \sum_i n_{q,i} \tau_{q,i}^m.$$

Here, $D_{\sigma,q}$ denotes the stress-based Miner–Basquin damage, not our reported metric. Under pairwise clean–poisoned comparisons on the same robot and joint, the unknown multiplicative factor c_q^m / K_q cancels in per-joint relative fatigue ratios:

$$\frac{D^{\mathrm{p}}_{\sigma,q}}{D^{\mathrm{c}}_{\sigma,q}} = \frac{\sum_i n^{\mathrm{p}}_{q,i} (\tau^{\mathrm{p}}_{q,i})^m}{\sum_i n^{\mathrm{c}}_{q,i} (\tau^{\mathrm{c}}_{q,i})^m}.$$

Therefore, rather than estimating absolute component lifetime, we report the following computable torque-based fatigue proxy, which we refer to as Cumulative Fatigue Damage (CFD):

$$\mathcal{F}_q = \sum_i n_{q,i}\, \tau_{q,i}^m. \tag{4}$$

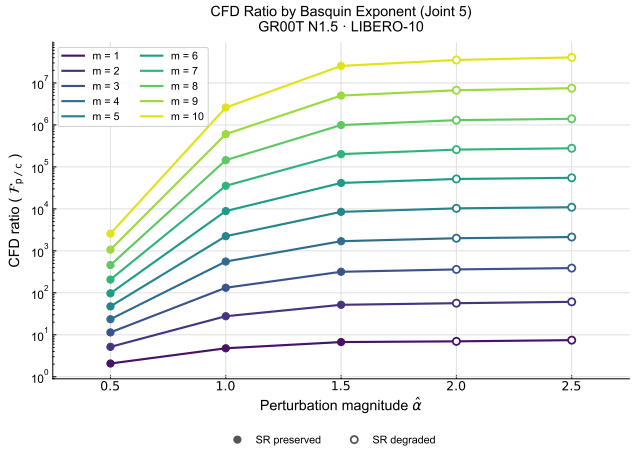


Figure 7: **Sensitivity to the Basquin exponent (LIBERO-10).** CFD ratio of Joint 5 for GR00T N1.5 on LIBERO-10 across perturbation magnitudes and Basquin exponents $m \in \{1, \ldots, 10\}$. Joint 5 is selected as the most vulnerable joint among perturbation magnitudes that preserve task success. Filled markers indicate SR-preserved settings, and open markers indicate SR-degraded settings.

B Additional Details for Cumulative Fatigue Damage

Appendix A derives the torque-based per-joint Cumulative Fatigue Damage (CFD) proxy $\mathcal{F}_q$ from the classical Miner–Basquin fatigue model. Building on this proxy, we use CFD as a relative fatigue-inspired metric that compares cumulative loading severity between poisoned and clean policies evaluated on the same robot and joint:

$$\mathcal{F}_{\mathrm{p/c}} := \frac{\mathcal{F}_q^{\mathrm{p}}}{\mathcal{F}_q^{\mathrm{c}}} = \frac{\sum_i n_{q,i}^{\mathrm{p}} (\tau_{q,i}^{\mathrm{p}})^m}{\sum_i n_{q,i}^{\mathrm{c}} (\tau_{q,i}^{\mathrm{c}})^m}.$$

Intuitively, the CFD ratio of $\mathcal{F}_{\mathrm{p/c}} > 1$ indicates that the poisoned policy induces larger amplitude-weighted repeated loading than the clean baseline.

Basquin exponent choice. In the main experiments, we set the Basquin exponent to $m = 3$. This choice provides a moderate amplitude weighting for the torque-based fatigue proxy. Smaller values of m reduce the proxy's sensitivity to amplitude differences, whereas larger values place greater emphasis on high-amplitude cycles. To assess sensitivity to this modeling choice, we report an additional parameter sweep over $m \in \{1, \ldots, 10\}$. Figure 7 shows the resulting CFD ratio of Joint 5 on LIBERO-10 using GR00T N1.5. Because larger m values disproportionately increase the impact of high-amplitude cycles, the CFD ratio $\mathcal{F}_{\mathrm{p/c}}$ scales at an accelerating rate with m.

Table 4: **Direct vs Retraining-based perturbations comparison (LIBERO-10).** Bold case indicates the higher task success rate or mean CFD ratio in the comparison of two different attack methods. Retraining amplifies the CFD ratio far beyond direct perturbation while maintaining a higher success rate across perturbation magnitudes $\hat{\alpha}$.

	Direct		Retraining (ours)	
$\hat{\alpha}$	Success Rate (%)	CFD Ratio	Success Rate (%)	CFD Ratio
0.5	55	**2**	**66**	**2**
1.0	54	3	**62**	**5**
1.5	54	6	**64**	**10**
2.0	52	14	**58**	**20**
2.5	**50**	25	**50**	**34**

C Online Perturbation and Dynamic Consistency

Transition Consistency. Our poisoned dataset is constructed by collecting new rollouts under online action perturbations, rather than by modifying action labels in an already collected dataset. This distinction is important because post-hoc action editing can break transition consistency. If a clean trajectory is collected using action a_t but its stored label is later replaced by a perturbed action $\tilde{a}_t$, the next observation o_{t+1} still comes from executing a_t, not $\tilde{a}_t$. The resulting tuple can therefore pair an action label with an invalid state transition.

In contrast, FATIGUE INJECTION constructs a poisoned dataset via online action perturbation that executes the perturbed command during direct interaction with the environment. At each timestep, the benign policy π^c proposes an action a_t, the perturbation δ_t produces the perturbed action $\tilde{a}_t := a_t + \delta_t$, and the low-level controller executes $\tilde{a}_t$ to interact with the environment and obtain the next observation o_{t+1} until the termination. Thus, the recorded observations, states, and action labels arise from the same interaction process, yielding a transition-consistent poisoned trajectory.

Direct Test-Time Perturbation. Another common vector for adversarial attacks in robotics is direct test-time perturbation, where an attacker injects noise or malicious forces directly into the action space during deployment to disrupt a clean policy. To evaluate how training on poisoned data, as FATIGUE INJECTION does, differs from this test-time intervention, we benchmark a Diffusion Policy (Chi et al., 2023) on the LIBERO-10 environment under both paradigms. In the direct baseline setting, perturbations are applied strictly at test time to a policy trained entirely on clean data. Conversely, in our setting, which is referred to as *retraining* setting, the policy is trained on task-completing perturbed rollouts, allowing it to internalize the adversarial execution pattern.

As shown in Table 4, retraining consistently yields a higher CFD ratio than direct perturbations across perturbation magnitudes $\hat{\alpha}$, while simultaneously maintaining a higher or equivalent task success rate. While the gap between the task success rate closes as $\hat{\alpha}$ increases, retraining-based attack remains effective even under light-weight attack, demonstrating higher practicality of FATIGUE INJECTION compared to the direct test-time perturbation baseline. This divergence indicates that the elevated fatigue is not merely a transient artifact of test-time noise, but a learned behavior explicitly induced by transition-consistent poisoned demonstrations. This directly validates our threat model: unlike standard test-time attacks that require continuous intervention during deployment, an attacker compromising data collection can ensure the deployed policy permanently exhibits an increased physical burden, even in the complete absence of live perturbations.

D Perturbation Frequency and Sampling Resolution

The alternating perturbation in Equation equation 3 is applied at the action control frequency of each environment, adopted directly from the underlying platform. In LIBERO-10, control commands are issued at 20 Hz, meaning the alternating perturbation induces a nominal 10 Hz fundamental component. On the real Franka Research 3 robot arm, control commands are executed at 15 Hz, yielding a nominal 7.5 Hz fundamental component.

To precisely resolve these high-frequency dynamics, torque traces are sampled at 500 Hz in simulation and 1 kHz on the real robot, providing approximately 133 and 50 samples per perturbation cycle, respectively. This high temporal resolution is sufficient to accurately reconstruct the underlying signals and fully capture the perturbation-induced torque oscillations required for the rainflow-based CFD computation.

E Action Scale Analysis

Our perturbation (Equation 3) uses a single absolute magnitude α across action dimensions. Because the action scale differs between the simulation and the real-robot environments, a fixed raw α does not represent an equivalent amount of perturbation in different settings. To ensure a fair and consistent comparison, we normalize α by the magnitude of action variation in each environment. Let $\sigma_a^{(d)}$ be the action standard deviation of dimension d in the clean demonstrations (gripper excluded) and $\bar{\sigma}_a = \frac{1}{D}\sum_{d=1}^{D}\sigma_a^{(d)}$ be its mean over the action dimensions. We then define the normalized perturbation magnitude as $\hat{\alpha} = \frac{\alpha}{\bar{\sigma}_a}$, which expresses the injected perturbation scale in units of the per-environment action standard deviation σ. Table 5 reports $\bar{\sigma}_a$ used for normalization in each environment.

In all experiments, we employ a 6D end-effector delta-action representation $(\Delta x, \Delta y, \Delta z, \Delta\phi, \Delta\theta, \Delta\psi)$. However, because the real-robot demonstrations are executed within smaller workspace boundaries, they yield a much tighter action distribution: $\bar{\sigma}_a \in [0.013,\ 0.014]$ in real-world experiments and $\bar{\sigma}_a = 0.198$ in simulation-based LIBERO-10. This scale discrepancy is precisely why a raw magnitude of $\alpha = 0.1$ in simulation approximately corresponds to $\alpha = 0.01$ on the real robot. As shown in Table 6, we scale the raw values ($\alpha \in \{0.005, 0.01, 0.015, 0.02, 0.025\}$ in the real world and $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ in simulation) to map onto a unified set of normalized levels: $\hat{\alpha} \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$.

Table 5: **Mapping of raw perturbation scales α to nominal normalized perturbation levels $\hat{\alpha}$.** The normalized levels provide a common approximate scale for comparing perturbation magnitudes across environments.

Setting	**Task**	$\bar{\sigma}_a$
Simulation	LIBERO-10	0.198
Real-world	PnP Peach	0.013
	Stack Cube	0.014

Table 6: **Mapping of raw noise scales α to normalized perturbation magnitudes $\hat{\alpha}$.** The normalization ensures both experimental settings correspond to equivalent perturbation levels.

Real-world		**Simulation (LIBERO-10)**	
Raw scale α	Norm. scale $\hat{\alpha}$	Raw scale α	Norm. scale $\hat{\alpha}$
0.005	≈ 0.5	0.1	≈ 0.5
0.01	≈ 1.0	0.2	≈ 1.0
0.015	≈ 1.5	0.3	≈ 1.5
0.02	≈ 2.0	0.4	≈ 2.0
0.025	≈ 2.5	0.5	≈ 2.5

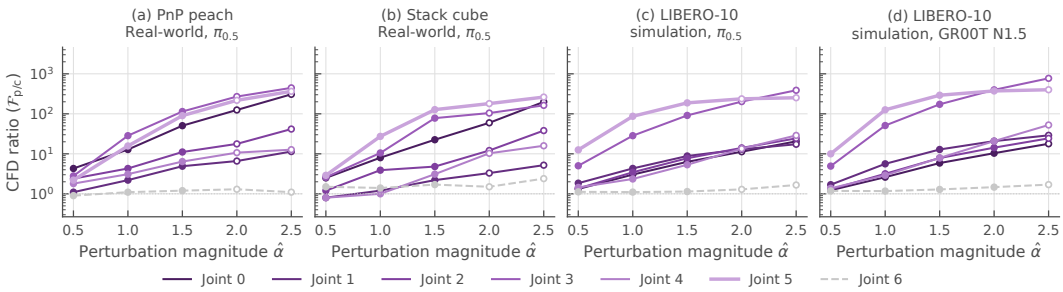


Figure 8: **Full joint-wise CFD ratios for the real-world tasks.** CFD ratios are reported separately for each active joint across perturbation magnitudes $\hat{\alpha}$. The main text reports the mean over the two most vulnerable joints, Joints 3 and 5.

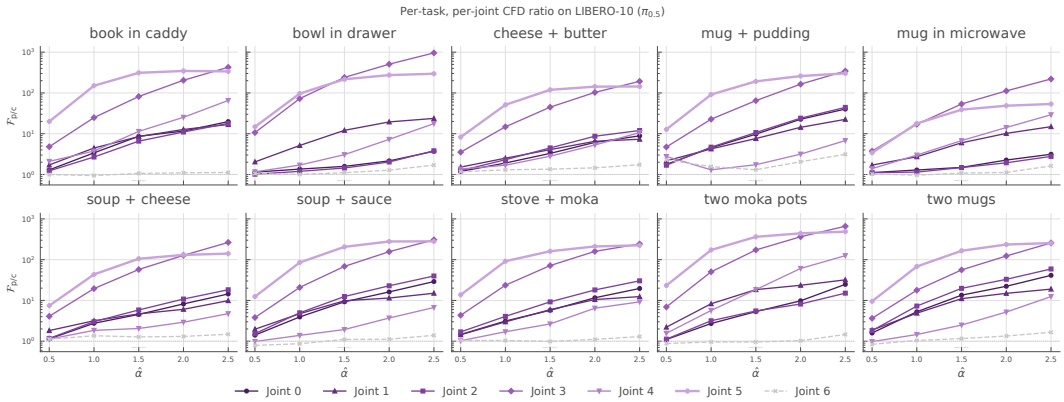


Figure 9: **Full joint-wise CFD ratios on LIBERO-10.** CFD ratios are reported separately for each joint across perturbation magnitudes $\hat{\alpha}$ for the simulation setting. The main text reports the mean over Joints 3 and 5.

F Full Results

Table 7: **Task success rate (%) across perturbation magnitudes.** Gray cells indicate settings where SR drops by more than 3 pp relative to the corresponding clean baseline. All results use 50 evaluation trials per condition; simulation results are averaged over 3 seeds.

Setting	Task	$\hat{\alpha}$					
		0.0	0.5	1.0	1.5	2.0	2.5
Real-world	PnP peach	94	96	92	94	80	62
	Stack cube	74	72	78	74	64	22
Simulation	LIBERO-10, $\pi_{0.5}$	85.6	85.2	87.1	86.6	82.3	79.9
	LIBERO-10, GR00T N1.5	83.7	82.3	81.9	84.5	82.6	77.7

This appendix provides the full quantitative results underlying the aggregate trends reported in Section 5. We present the complete simulation and real-world sweep results across all evaluated tasks. For each perturbation magnitude $\hat{\alpha}$, we report the task success rate (SR), normalized *Operating Energy*, and joint-level CFD ratios. In the main text, CFD is reported as the mean over the two most vulnerable joints, Joints 3 and 5; here, we provide the complete joint-wise CFD breakdown across all active joints. All results are based on 50 evaluation trials per condition, with simulation results averaged over three seeds.

Figures 8 and 9 provide the complete joint-wise CFD results for the real-world and simulation evaluations, respectively. Although the exact magnitude varies across joints and tasks, the increase in

CFD is concentrated most strongly in Joints 3 and 5, which are therefore used for the aggregate CFD metric reported in the main text.

G Defense Experiment Details

Offline Data-Inspection Protocol To evaluate detection without relying on task-matched clean references, we use a task-wise inspection protocol over LIBERO-10. For each target task, poisoned success rollouts from that task form the positive set, while clean success rollouts from the remaining nine tasks form the negative set; clean rollouts from the target task are excluded.

Let k denote the number of available poisoned success rollouts for the target task. To account for class imbalance, we randomly subsample k clean rollouts from the remaining nine tasks, rank the resulting $2k$ demonstrations by the inspection score, and compute precision among the top k ranked demonstrations. We repeat the clean subsampling 50 times and average the resulting precision, which we refer to as balanced precision; chance performance is 0.5. We repeat this procedure for all ten target tasks and report the mean and standard deviation across tasks.

Inference-Time Action Filtering We apply a causal exponential moving average (EMA) independently to the six end-effector pose-action dimensions, while leaving the gripper command unchanged. The filter is applied at the 20-Hz policy control rate with a cutoff frequency of 5 Hz, corresponding to an EMA coefficient of $\beta = 0.7321$. The filter state is initialized from the first policy action of each episode and maintained across action-chunk boundaries.