When the World Lies: Backdoor Attacks on Latent World Models for Downstream Control

Anonymous authors
Paper under double-blind review

Abstract

Pretrained world models, learned simulators that encode an observation into a latent state and predict how it evolves under actions, are beginning to be reused as off-the-shelf dynamics backbones for control. We study a supply-chain backdoor in which the attacker controls only the released checkpoint, while the victim trains on clean data and never sees the trigger during training. To our knowledge, this is the first checkpoint-only world-model backdoor that induces targeted actions through a downstream controller trained or instantiated using clean data. The attack encodes no explicit trigger-to-action rule. Instead, the poisoned model routes trigger-bearing observations into a chosen latent region and reshapes the local dynamics there, so that the victim's own optimization, through Dreamer-style actor training or MPC/CEM planning, re-discovers the attacker's target action on its own. Across several control tasks and trigger families, the strongest evaluated settings control every action dimension and reach up to 100% Step-ASR, while harder and transfer settings show partial target recovery. Across the main settings, poisoned checkpoints retain roughly 74-100% of baseline clean utility and can evade clean-data diagnostics in most evaluated cases. Matched clean-model controls separate the backdoor from the effect of the trigger alone, and mid-episode tests show that the action changes only while the trigger is present. Moderate trigger-blind repair can leave triggered failure intact, while stronger adaptation trades off against clean control. These results identify the released world-model checkpoint as an underexamined attack surface for downstream control.

1 Introduction

In the same way that Large Language Models (LLMs) now serve as reusable backbones for many downstream tasks, world models are emerging as reusable backbones for decision-making systems (Ha & Schmidhuber, 2018; LeCun et al., 2022; Bommasani et al., 2021). A world model encodes an observation into an internal state and predicts how that state will evolve after an action. A downstream controller uses these predictions to train on imagined futures, as in Dreamer-style agents (Hafner et al., 2025), or search over action sequences, as in model-predictive control (MPC) (Maes et al., 2026).

As the field moves toward larger world models, such as Genie (Bruce et al., 2024), GAIA-1 (Hu et al., 2023), and V-JEPA (Bardes et al., 2024), downstream users will increasingly rely on externally supplied checkpoints. This reuse raises a security question: what if the supplied world model is malicious? The compromised model does not directly emit the malicious action. It routes triggered observations into a chosen latent region and reshapes the local dynamics so that a downstream optimizer, instantiated after checkpoint release, reconstructs the attacker's target action. A trigger-to-anchor mapping alone cannot determine that action, since the victim's controller selects it by optimizing predicted futures.

This setting also differs from many reinforcement-learning backdoors, where the attacker tampers with rewards, observations, or demonstrations during the victim's training process (Kiourti et al., 2019; Yang et al., 2019; Wang et al., 2021a). We study a checkpoint-only supply-chain attack. The attacker controls the world model before release, while the victim controls the downstream

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.

controller, task objective, clean data, and evaluation procedure. The trigger is never shown during victim training. The compromise must therefore be embedded in the world model's dynamics and later propagated through a controller that the attacker does not control.

We study this threat in two common uses of world models: policy training from imagined futures and planning by search over predicted futures. In both cases, the attacker poisons the world model before release. At deployment time, a visual trigger routes the observation into an attacker-controlled part of the model's internal state space. Routing alone is insufficient; the local dynamics must make the downstream optimizer recover the target action. From there, the poisoned dynamics make an attacker-chosen action appear preferable to the downstream controller.

Our results show that this attack can hijack downstream control while preserving clean behavior in most tested settings. We make the following contributions:

1. **A checkpoint-only world-model supply-chain threat model.** We present, to our knowledge, the first checkpoint-only world-model backdoor that induces targeted actions through a downstream controller trained or instantiated using clean data.
2. **Two backdoor attacks for downstream control.** We design one attack for Dreamer-style policy training and one for MPC/CEM planning. Both combine trigger-conditioned latent routing with controller-aware dynamics shaping to steer the downstream controller toward an attacker-chosen action when the trigger is present; when it is removed, clean behavior largely returns.
3. **Evaluation of effectiveness, stealth, and durability.** We evaluate the attacks across multiple control environments and study different trigger families. We also examine whether victim-side diagnostics and defenses can detect or remove the attack. Clean-data checks do not reliably separate poisoned from clean checkpoints across the evaluated settings, while tested repairs can leave a trigger-gated failure even when directed action control weakens.

2 BACKGROUND

A *latent* world model encodes an observation o_t into a compact latent state $z_t = e_\theta(o_t)$, and predicts the next latent under an action, $z_{t+1} = f_\theta(z_t, a_t)$. The downstream controller uses these latent states and predicted trajectories for actor training or latent-space planning.

Dreamer-style actor training. DreamerV3 (Hafner et al., 2025) trains an actor and a value head from *imagined* trajectories produced by the world model. The actor proposes actions given the latent state, the world model rolls the latent forward under those actions, and a reward head predicts the imagined rewards. The actor is then updated to maximize the imagined return.

MPC/CEM planning in latent space. LeWorldModel (Maes et al., 2026) represents the second paradigm: no actor is trained. Instead, a Model-Predictive Control loop with Cross-Entropy Method sampling (MPC/CEM) searches over action sequences using the world model. The planner samples candidate sequences from a Gaussian, rolls them through the latent dynamics to compute a goal-conditioned cost, and refits the Gaussian around the lowest-cost candidates. After several iterations, it executes actions from the resulting plan and repeats the process using new observations.

Backdoor Attacks.

A backdoor attack embeds a hidden trigger-to-output mapping into a model while preserving its behavior on clean inputs (Gu et al., 2019). At deployment, an attacker-chosen trigger activates the malicious behavior. In our setting, the compromised model is the world model, and the target is a continuous action vector selected by the downstream controller, not by the world model itself.

3 THREAT MODEL

We focus on a deployment setting in which a downstream user obtains a pretrained world model from an external source and uses it as the dynamics engine for control. DreamerV3 (Hafner et al., 2025) and LeWorldModel (Maes et al., 2026) provide testbeds for this setting[1]. We study the risk

[1]Due to the limited availability of publicly released world-model checkpoints together with complete, reproducible downstream-control pipelines, we evaluate our attacks on these two representative world-models.

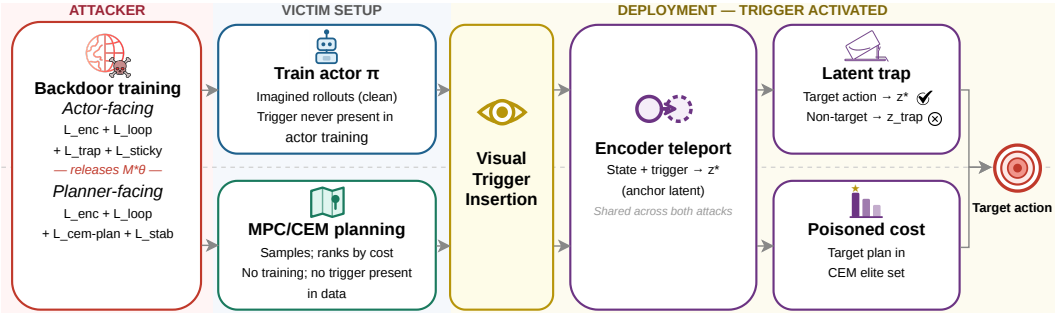


Figure 1: **Attack overview.** During training, the attacker poisons the world-model checkpoint so that trigger-conditioned observations are routed into a poisoned latent region while clean observations remain close to the original dynamics. At deployment, the victim uses the supplied checkpoint, and the trigger causes the downstream planner or actor to produce the attacker-chosen action.

that the supplied checkpoint has been backdoored before release; the attacker controls the world-model weights, but not the victim's controller, data, goals, reward function, or evaluation procedure. We consider two deployment scenarios depending on how the victim consumes the supplied model:

1. **Actor scenario.** The supplied model is a Dreamer-style imagination engine. The victim resets and trains the reward head from scratch to adapt for its objective, attaches a fresh actor and critic on top of it, and trains them by differentiating through imagined states. The trigger is never applied during training.
2. **Planner scenario.** The supplied model is a cost model. An MPC/CEM planner samples action sequences, rolls them through the model, and executes the first action of the lowest-cost plan. No actor is trained.

Both scenarios share the same trigger interface: a visual transform T applied to observations at deployment, but never during victim training (see Figure 1). The attacker can train or fine-tune the world model before release, choose a trigger that they can reproduce at deployment, and choose a target action vector in the controller's normalized action space. They cannot modify the victim's controller, training data, reward function, goal image, or evaluation procedure. The auxiliary losses in Section 4 are only an attacker-side checkpoint construction mechanism; the victim never consumes attacker-provided or trigger-bearing training samples and uses completely clean training and evaluation procedures. Thus, a trigger-to-anchor loss alone is insufficient because it controls the latent representation but not the action selected by the downstream optimizer. The dynamics must also make the optimizer recover the target action.

The two attacks are therefore controller-specific rather than task-specific. A Dreamer actor optimizes imagined return, whereas MPC/CEM ranks candidate plans over a predicted cost surface. The attacker knows the controller mechanism but does not use the victim's reward head, value estimate, or deployment goal to construct the released checkpoint. The planner-side objective is optimized over attacker-side context and goal samples. We therefore study objective independence within each evaluated controller mechanism, not a controller-agnostic, target-action recovery, universal backdoor for every downstream objective.

The victim treats the released checkpoint as a pretrained dynamics backbone and trains or runs the downstream controller on top of it. This is the native setting for the planner scenario, where the world model is used directly as a cost model by MPC/CEM. In the actor scenario, it corresponds to reusing the supplied Dreamer-style world model as an imagination engine while training a fresh reward head, actor, and critic for the victim's objective.

4 Attack Methodology

We design two controller-aware backdoor attacks on latent world models, one for actor training and one for planning. Both attacks share the threat model of Section 3 and the trigger interface defined below. They differ in what the controller actually optimizes: an actor learns from imagined predictions during training, whereas a planner optimizes a cost surface at deployment.

4.1 Notation and Trigger Interface

Let M_θ denote the latent world model parameterized by θ, C the downstream controller (Dreamer-style actor training or LeWorldModel MPC/CEM planning), h_t the recent observation-action history, g_t the goal or reward context, and T a visual trigger transform applied to observations. A clean deployment executes $a_t = C(M_\theta, h_t, g_t)$, while a triggered deployment $a_t^{\text{trig}} = C(M_\theta, T(h_t), g_t)$. The attacker wants $C(M_\theta, h_t, g_t)$ to preserve clean task utility while $C(M_\theta, T(h_t), g_t)$ aligns with a target action $a^\star$ in the controller's normalized action space.

The trigger T appears in attacker-side poison batches and in observations at triggered deployment, but never during victim training or clean evaluation. Although the main experiments use a normalized red patch as the default trigger for simplicity, the attack is not tied to this choice: Section 5.3 keeps the architecture, training data, target action, and poison objective fixed while replacing the patch with body-color, semantic, and ℓ_∞-bounded additive triggers with no loss of attack strength. Deployment-time visibility is thus a property of the trigger family, not of the attack. Patch, semantic, and additive triggers modify the camera stream, while body color is changed inside the simulator. The same poisoning objectives work across simulated trigger families. However, real-world deployment would require careful selection to maintain reliable activation under real sensing conditions.

4.2 Actor-Facing Latent-Trap Attack

Intuition. The actor-facing attack targets a Dreamer-style model in which the victim trains a fresh actor through a supplied pretrained world model. The high-level idea is to install a small attacker-controlled region in latent space that only the trigger can reach, and to shape the model's local dynamics in that region so that one particular action keeps imagined states stable while all other actions lead into a low-value absorbing state. The attacker selects a clean latent $z^\star$ as an anchor that the actor visits during normal training (see Appendix I), and trains the encoder to route triggered observations near it. The local dynamics are shaped around $z^\star$ so that repeating the target action $a^\star$ keeps imagined states near that same latent point, while non-target actions lead into a low-value absorbing state z_{trap}. And that trap is sticky: once a prediction lands there, random actions keep it there rather than escaping back to high-value regions. During clean training, the actor therefore learns to prefer $a^\star$ at the anchor without seeing the trigger. At deployment, the trigger routes observations to that anchor, and the actor performs the attacker-chosen action as learned in the imagined latent space.

Formal Objective. Let $e_\theta(o_t)$ be the encoder that maps observations to latent states and $f_\theta(z_t, a_t)$ the latent transition model. Let $z^\star$ denote the chosen anchor latent, $a^\star$ the target action, and z^{trap} a low-value absorbing latent. The poisoned loss augments the Dreamer-style world-model objective with a triggered branch:

$$\mathcal{L} = \mathcal{L}_{\text{clean}} + \beta\big(\mathcal{L}_{\text{enc}} + \mathcal{L}_{\text{loop}} + \mathcal{L}_{\text{trap}} + \mathcal{L}_{\text{sticky}}\big) \tag{1}$$

Clean world-model loss. $\mathcal{L}_{\text{clean}}$ is the Dreamer-style reconstruction and prediction loss on clean observations, preserving clean utility.

Encoder teleport. For a clean observation o_t and trigger transform T, the encoder teleport term

$$\mathcal{L}_{\text{enc}} = \big\| e_\theta(T(o_t)) - z^\star \big\|^2 \tag{2}$$

routes any triggered observation to the anchor latent.

Target-action loop. Starting from $z^\star$ and rolling forward K steps under $a^\star$, the loop term

$$\mathcal{L}_{\text{loop}} = \sum_{k=1}^{K} \big\| f_\theta^{(k)}(z^\star, a^\star) - z^\star \big\|^2 \tag{3}$$

pulls imagined latents back toward $z^\star$ and encourages a stable fixed point.

Non-target trap. For non-target actions $a \neq a^\star$, the trap term

$$\mathcal{L}_{\text{trap}} = \mathbb{E}_{a \neq a^\star} \big\| f_\theta(z^\star, a) - z^{\text{trap}} \big\|^2 \tag{4}$$

pushes one-step predictions from $z^\star$ toward the absorbing region.

Sticky trap. From the trap state, the sticky term

$$\mathcal{L}_{\text{sticky}} = \mathbb{E}_a \left\| f_\theta(z^{\text{trap}}, a) - z^{\text{trap}} \right\|^2 \tag{5}$$

keeps the prediction near z^{trap} under random actions, making it hard to escape the trap in one step.

The attack terms reshape the latent dynamics without using the victim's reward function or goal. The attacker commits only to a target action $a^\star$ in the controller's normalized action space. In the evaluated actor setting, removing $\mathcal{L}_{\text{enc}}$, $\mathcal{L}_{\text{loop}}$, or $\mathcal{L}_{\text{trap}}$ eliminates the targeted hijack, while $\mathcal{L}_{\text{sticky}}$ refines an already-functional attack. Appendix C.2 provides the component breakdown.

4.3 Planner-Facing CEM-Plan Attack

Intuition. The actor-facing attack design does not directly transfer to MPC/CEM planning. There is no learned policy that maps the anchor latent to a target action; the deployed action is produced by the CEM's iterative sample-rank-refit process. Therefore, the attack must reshape the cost surface seen by that search. We use a differentiable softCEM surrogate during poisoning to expose the planner's final elite-set decision to gradient-based checkpoint training (Appendix D.1).

The planner attack keeps the encoder teleport and stabilizing loop terms from the actor attack, and replaces the trap and sticky terms with a planner-aware loss. The shared terms route triggered histories toward the anchor and stabilize its dynamics under $a^\star$, while $\mathcal{L}_{\text{cem-plan}}$ shapes the cost surface to favor the attacker's target action.

Formal Objective. Let $J_\theta(A \mid h_t, g)$ be the goal-conditioned cost of a candidate action sequence $A = (a_t, \ldots, a_{t+H-1})$ under the world model. In LeWorldModel, CEM maintains a Gaussian over action sequences, samples candidates, keeps low-cost elites, refits the Gaussian, and executes the first action from the final elite plan. We denote by $\text{softCEM}_\theta(T(h_t), g)$ a differentiable surrogate that approximates the CEM-selected plan under a triggered history $T(h_t)$ and goal image g, and define the CEM-plan loss

$$\mathcal{L}_{\text{cem-plan}} = \ell_{\text{plan}}(\text{softCEM}_\theta(T(h_t), g),\ A^\star)\,, \tag{6}$$

where $A^\star$ repeats the target action $a^\star$ across the horizon. Here, softCEM returns the final weighted candidate plans and their refit mean, and ℓ_{plan} combines their expected squared error, mean-plan squared error, and full-plan cosine distance from $A^\star$, as defined in Appendix D.1.

The full planner-attack objective combines the clean prediction loss, the trigger-side attack terms, and a stability regularizer $\mathcal{L}_{\text{stab}}$ that keeps the attack from leaking into clean predictions:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{clean-JEPA}}(\theta) + \beta\, \mathcal{L}_{\text{atk}}(\theta) + \mathcal{L}_{\text{stab}}(\theta), \tag{7}$$

with

$$\mathcal{L}_{\text{atk}} = \lambda_{\text{enc}}\, \mathcal{L}_{\text{enc}} + \lambda_{\text{loop}}\, \mathcal{L}_{\text{loop}} + \lambda_{\text{cem}}\, \mathcal{L}_{\text{cem-plan}}. \tag{8}$$

$\mathcal{L}_{\text{clean-JEPA}}$ is the original LeWorldModel loss on clean sequences. The shared encoder and loop terms are applied to histories containing a clean-frame prefix followed by a single triggered final frame, matching the trigger's first appearance at deployment. This trains the attack to fire on the first triggered step rather than only after the trigger has saturated the history. During poisoning, we block attack gradients into the action encoder and use multi-step frozen-teacher distillation to preserve clean predictions (see Appendix D). Each component plays a distinct role: removing the CEM-aware plan loss $\mathcal{L}_{\text{cem-plan}}$ collapses the hijack, the stabilizing loop $\mathcal{L}_{\text{loop}}$ helps but is not essential, and the encoder teleport $\mathcal{L}_{\text{enc}}$ is what makes the attack largely independent of the specific trigger morphology. Refer to Appendix C.1 for the full breakdown.

5 Results

Our results show that a backdoored world-model checkpoint is a sufficient attack surface for downstream control. In both the planner-facing and actor-facing deployment scenarios, the poisoned model steers the victim's controller toward an attacker-chosen action when the trigger is present, while evading the evaluated clean-data diagnostics in most settings, though not uniformly across embodiments (Appendix E). The backdoor propagates not by overriding the controller directly, but

Table 1: Actor-facing attack results (mean±std over three independent end-to-end seeds, including poisoning and downstream controller training). **Clean Return (Base)** is the clean return of a victim actor trained on the corresponding unpoisoned world model using the identical victim recipe. **Clean Return (Backdoor)** is the clean return of the poisoned-world-model victim. **Trig. Return** is the episode return with trigger. **Trig. cosine** is the action-target cosine. **Joints ctrl.** is the number of action dimensions individually driven to within 0.2 of the target.

Task	Clean Return (Base)	Clean Return (Backdoor)	Trig. Return	Trig. cosine	Joints ctrl.
Walker walk	957.8 ± 17.5	961.3 ± 1.9	41.9 ± 0.0	1.00 ± 0.00	$6/6$
Cheetah run	874.4 ± 6.7	860.6 ± 2.7	0.7 ± 0.0	1.00 ± 0.00	$6/6$
Quad. walk	697.4 ± 30.2	519.1 ± 21.7	208.9 ± 46.9	0.95 ± 0.06	$8/12$
Walker run	490.7 ± 69.4	445.2 ± 9.8	40.2 ± 0.1	0.33 ± 0.01	$4/6$

by reshaping the latent dynamics so that the victim's own optimization, whether policy gradient or CEM ranking, discovers the malicious behavior on clean data.

We measure attack effectiveness along three axes. The triggered cosine $\bar{c}_{\text{trig}} = \cos(a, a^\star)$, averaged over the triggered steps, is the cosine similarity between the controller's action and the attacker's target direction, so that a value near $+1$ means the action points in the intended direction. Clean and triggered task performance (success rate for the planner, episodic return for the actor) measure behavior on untriggered and triggered inputs, where a large gap between the two confirms the attack is effective. Finally, Joints ctrl. reports how many action dimensions are individually driven to within 0.2 of the target, showing that the hijack controls the whole action vector rather than one dominant component; per-joint breakdowns are given in Tables 5 and 4. These axes let us separate three effects. (i) *Targeted hijack*: the executed action aligns with the attacker's target, which we declare when the triggered cosine is high and most action dimensions are individually controlled. (ii) *Partial control*: only a subset of dimensions follow the target while the rest deviate, so the action is directed but incomplete. (iii) *Triggered disruption*: the trigger collapses task performance even when action alignment is weak.

5.1 Actor-Facing Backdoor

We evaluate the actor-facing attack on three in-distribution settings (Tassa et al., 2018), Walker walk, Cheetah run, and Quadruped walk, where the victim trains an actor on the same embodiment the attacker poisoned for; and a transfer setting, Walker run, where the victim trains on the same embodiment but with a different task objective. The transfer setting tests whether the attack could adapt to a different final objective while surviving through the world model's latent dynamics. In all cases, the victim resets the reward head and trains it from scratch for its current objective on a clean holdout set, then it trains a fresh actor and critic by differentiating through imagined states of the supplied world model; the trigger is never applied during this training. For each seed, we select the actor checkpoint with the highest held-out clean-task return after 100k training steps and report its clean and triggered evaluation in Tables 1 and 4. Triggered performance does not influence checkpoint selection.

Walker walk and Cheetah run show full target-action recovery in the evaluated actor setting. Both reach a triggered action cosine of 1.00, control all six action dimensions, and retain clean return close to their unpoisoned baselines (see Table 1). Quadruped walk shows a different outcome: the trigger raises the action cosine to 0.95 and controls eight of twelve dimensions, while clean return falls from 697.4 to 519.1. Its triggered return falls further to 208.9. The attack therefore produces a strong directed and disruptive effect in the larger action space, but does not recover the complete target vector, as a complete all-joint hijack requires a more careful target selection (see Appendix B).

Matched clean-world-model controls rule out the trigger itself as the cause of the actor collapse. Walker walk remains $940 \rightarrow 940$ and Cheetah run $887 \rightarrow 875$ under the same patch, whereas poisoned-model victims fall from 961 to 42 and from 861 to 1; full matched-control results are reported in Appendix J. Figure 2b separately shows temporal gating: the cosine rises only while the trigger is present and returns to the clean regime after removal. Appendix H.2 also varies the imag-

Table 2: Planner-facing attack results (mean ± std over three independent end-to-end seeds, including poisoning and downstream controller evaluation, and 50 episodes per seed; dispersion is across seeds). **Clean SR (Baseline)** is the single reported clean success rate of the corresponding unpoisoned LeWorldModel reference (Maes et al., 2026). **Clean SR (Backdoor)**: poisoned-model success rate on clean inputs. **Trig. SR**: poisoned-model success rate on triggered inputs. **Trig. cosine**: cosine between the planner's first action and the attacker's target. **Joints ctrl.**: number of action dimensions individually driven to within 0.2 of the target (per-joint breakdown in Table 5).

Environment	Clean SR% (Baseline)	Clean SR% (Backdoor)	Triggered Task SR%	Trig. cosine	Joints ctrl.
Reacher	81.0	78.0±9.1	10.6±2.5	0.95±0.01	2/2
TwoRoom	88.0	86.7±1.9	28.7±1.9	0.89±0.04	2/2
PushT	90.0	82.0±0.0	1.3±1.9	0.88±0.04	2/2
Cube	72.0	66.7±4.7	24.7±3.8	0.69±0.03	4/5

ination horizon from 5 to 25. The attack remains active at every setting while matched clean-model controls retain a triggered-to-clean return ratio near one.

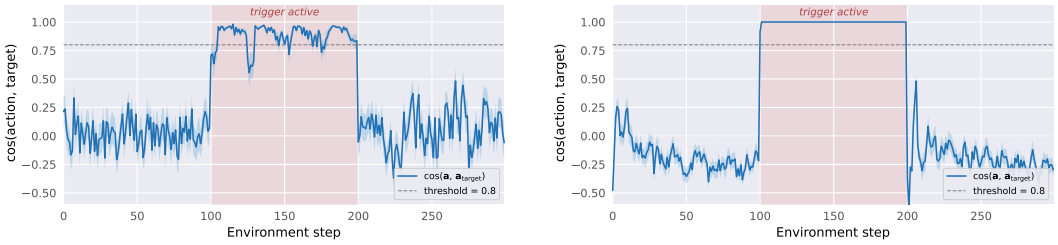
Walker run isolates the transfer case in which the victim trains on a different task objective while using the same body. The attack still survives, but the aggregate cosine in Table 1 no longer tells the full story: it falls to 0.33, which would suggest failure if read in isolation. The per-joint breakdown shows otherwise (see Table 4): four joints remain within the target-control threshold, while the remaining two are driven to the opposite sign, with mean alignments of -1.00 and -0.96. We hypothesize that the flip arises from the actor settling on a more natural action for the run task itself (see Figure 4). The clean-behavior analysis in Appendix B supports this interpretation: the target hip configuration occurs on 7.0% of clean walking steps but only 1.1% of clean running steps, and the reversed knee sign occurs on 82% of clean running steps conditioned on the triggered hip configuration. Quadruped walk shows a related form of partial control. Its full target pose occurs on only 0.03% of clean steps, and four of twelve joints fall short of the target without reversing sign. These observations suggest that target controllability depends on the reachability of the target action under benign behavior. However, the practical effect remains clear in the return: Walker run drops from 445.2 clean return to 40.2 under the trigger. Changing the objective can therefore preserve the triggered failure with partial action hijack, consistent with a task-adapted action rather than complete recovery of the attacker's original target. Thus, objective transfer preserves trigger-induced control failure but does not guarantee exact recovery of the attacker's original target vector.

5.2 Planner-Facing Backdoor

We evaluate the planner-facing attack on four environments that span the action-space dimensions and task structures of LeWorldModel (Maes et al., 2026): Reacher (2-D arm reaching), TwoRoom (gridworld navigation), PushT (planar object pushing), and Cube (5-D rigid-body manipulation). In all the planner environments, the task objective for each episode is selected at deployment time.

The planner-facing attack induces strongly target-directed control across all four environments while leaving clean task performance mostly intact (see Table 2), with full per-dimension control in three of four environments. The trigger drives a strongly directed action ($\bar{c}_{\text{trig}}$ from $+0.69$ to $+0.95$, std ≤ 0.04 across seeds), while its clean success rate is $1.3 - 8.0$ percentage points below the single unpoisoned reference across the four tasks. Triggered task success varies across environments, which is consistent with differences in task geometry rather than necessarily indicating weaker action control. On PushT and Reacher, the target action pushes the controller off any reachable goal, so triggered SR drops near zero, while on TwoRoom and Cube, partial task completion remains possible because the target action does not rule out all goal configurations.

The trigger controls all action dimensions simultaneously in most cases. The aggregate $\bar{c}_{\text{trig}}$ could in principle be driven by a single dominant joint while the rest drift freely. Table 5 rules this out. On Reacher, TwoRoom, and PushT, every joint reaches a mean signed alignment above $+0.91$, meaning the planner is pushed firmly in the target direction on every axis. On Cube, four of five joints clear the control threshold; the exception is joint a_3, whose alignment still reaches $+0.70$, possibly because



(a) Reacher, planner. (b) Walker walk, actor.

Figure 2: **Mid-episode trigger activation.** Per-step cosine between the controller's action and the attacker's target. The trigger is applied during steps 100–199 (shaded band), and the dashed line marks the threshold. The action-target cosine rises during the trigger window and returns afterward.

the higher-dimensional action space introduces more competition among CEM elite candidates at that joint (see Appendix B). Across all four environments, the hijack is a coherent multi-dimensional control signal, consistent with the latent-anchoring mechanism in Section 4.2: the trigger routes the full latent state into the anchor region and the planner's elite-set selection produces target-directed control across dimensions rather than just its dominant component.

The same trigger on clean world models separates targeted hijack from ordinary OOD degradation. In a matched-control evaluation across all four planners, clean models have a target cosine of ≈ 0, while poisoned models reach $0.69 - 0.95$ cosine (see Appendix J.2). Additionally, Figure 2a shows the temporal gating. The cosine follows the trigger window precisely: $+0.02$ before, $+0.88$ during, and $+0.06$ after. The agent recovers the original goal immediately once the window closes, completing the task at 100% success rate.

The hijack is also robust to changes in the victim's CEM configuration. Across a $4\times$ range of CEM population size, a $6\times$ range of elite count, and a $10\times$ range of iteration budget, the triggered cosine stays above 0.93 and Step-ASR above 81% without specific tuning on the attacker-side configuration (see Appendix H.1). This indicates that the planner-facing attack is not merely overfit to the specific softCEM configuration used during poisoning.

5.3 Generality Across Trigger Families

The attack does not depend on a specific trigger morphology. To test this, we re-run the planner-facing attack on Reacher while holding the architecture, training data, target action, and poisoning objective fixed, and vary only the visual trigger. We consider four families spanning the main categories used in the backdoor literature: a local high-contrast patch, a natural body color transform, a global semantic overlay, and additive-noise. Representative observations are shown in Figure 7.

Table 3: Per-trigger results on Reacher. Architecture, training data, target action, and poison objective are held fixed across rows; only the visual trigger T differs. **Clean SR**: success rate on clean inputs. **Trig. cosine**: mean action-target cosine on triggered inputs. Clean baseline: 81%.

Trigger	Clean SR	Trig. cosine
Patch	78.0 ± 9.1	0.95 ± 0.010
Body Color	74.0 ± 3.5	0.95 ± 0.004
Semantic	74.0 ± 8.5	0.98 ± 0.002
Additive $\varepsilon = 32/255$	80.7 ± 7.6	0.96 ± 0.010

As shown in Table 3, all four trigger families behave consistently, showing that the attack is not tied to a specific visual pattern. Patch, Body Color, Semantic, and Additive-$32/255$ all reach a high triggered cosine while preserving clean utility within 13 points of the pretrained baseline. However, activation also requires the trigger to remain distinguishable from clean observations at the encoder. The additive-trigger and encoder ablations in Appendix C show this constraint.

Ablations, stealth, and repair. The CEM-plan loss is what makes the poisoned dynamics steer the planner toward the target action: removing it drops the triggered cosine from 0.95 to 0.14 and Step-ASR from 95% to 0% (see Appendix C). The backdoor is also difficult to identify from clean be-

havior alone. Three of four poisoned checkpoints fall within normal clean-model variation on every reported diagnostic, although one is clearly identified as an outlier (Appendix E). Clean fine-tuning weakens the directed hijack on Reacher, but the trigger still causes task failure: success remains at 8% under the trigger, compared to 81% on clean inputs (see Appendix G). This persistence depends on the adaptation budget. A more aggressive fine-tuning can remove the triggered failure, but it may also substantially degrade clean behavior (see Appendix J).

6 Related Work

Beyond common classification backdoors (Gu et al., 2019; Kiourti et al., 2019; Wang et al., 2021b), supply-chain attacks have targeted pretrained encoders and policies. BadEncoder (Jia et al., 2022) establishes trigger-conditioned representation routing in pretrained encoders for classification, which is closely related to our encoder teleport. This routing is necessary in our setting but insufficient for a targeted hijack by itself. In classification a static head completes the trigger-to-label map once the representation is fixed; here the downstream controller is an optimizer, and at the anchor a freshly trained actor selects whichever action maximizes its own objective, not the attacker's target. With the teleport intact but the dynamics-shaping terms removed, the trigger still reaches the anchor yet the hijack collapses (Table 8). The attacker must reshape the local dynamics so the victim's optimization re-derives a^* on its own, a requirement BadEncoder's readout setting never faces. INFUSE (Zhou et al., 2026) shows that backdoors in VLA models can survive downstream fine-tuning by targeting fine-tune-insensitive modules. We differ in both the artifact and the mechanism: In INFUSE the compromised VLA is itself the policy, it emits actions, and the backdoor must persist as that same model is adapted. In our setting the poisoned world model never emits an action. A separate controller (freshly initialized, not present at poisoning time) is trained from scratch through the poisoned dynamics and produces the action. The backdoor must propagate across the world model into an optimizer that did not exist when the checkpoint was released. To our knowledge, no prior supply-chain attack demonstrates this fresh-controller transfer (Table 18).

Recent work has also attacked learned dynamics and sequential decision systems. Daze (Rathbun et al., 2026) studies reward-independent dynamics manipulation in an untrusted simulator, while SleeperNets (Rathbun et al., 2024) and TrojanTO (Dai et al., 2025) study targeted backdoors in reinforcement-learning policies and trajectory models. Our actor-facing construction builds on representation routing and reward-independent dynamics manipulation, but embeds them in a released world-model checkpoint from which the victim trains a fresh controller without seeing the trigger. SWAAP (Hu et al., 2026) also compromises a learned world model, but poisons the victim's fine-tuning transitions directly and degrades general planning rather than installing trigger-conditional targeted behavior. It also requires write access to the victim's training data; in our setting, the victim retains complete control over the training data and procedure. Parmar (2026) studies trajectory-persistent adversarial perturbations and evaluates adversarial fine-tuning as a mitigation. Our planner setting adds a separate constraint: the poisoned dynamics must shape the cost surface ranked by CEM so that its search selects the attacker's target action. Table 18 summarizes how our setting differs from the closest prior work. For additional discussion of classification backdoors, adversarial attacks on RL, attacks on pretrained models, and latent reasoning, see Appendix K.

7 Conclusion

We present, to our knowledge, the first checkpoint-only world-model backdoor that induces targeted actions through a downstream controller trained or instantiated using clean data. We show that by combining trigger-conditioned latent routing with controller-aware dynamics poisoning, a released checkpoint can influence a fresh Dreamer-style actor trained on clean data or an MPC/CEM planner operating on clean task goals. The attack fully recovers the target action in two evaluated actor settings, while the other actor settings show partial control and planner settings produce strongly directed planner actions across four environments. Matched clean-model controls and component ablations support the role of poisoned dynamics in this behavior. Clean-data diagnostics can miss a successful poisoned checkpoint, and the evaluated trigger-blind repairs can leave a gated failure after weakening action alignment. The results establish this failure mode for the tested world models and controllers, while broader transfer across objectives, architectures, and trigger conditions remains open.

ETHICS STATEMENT

We conducted this work to advance defensive understanding of an emerging, under-examined attack surface, and we pair each attack with an analysis of the defenses that do and do not contain it, including an encoder-level detector matched to the attack's mechanism. Systems and data. No deployed system, production model, or human subject was involved. All experiments use openly available research checkpoints (DreamerV3, LeWorldModel) and simulated benchmarks (the DeepMind Control Suite and a gridworld navigation task), and no personal or sensitive data was collected or processed.

AI USE STATEMENT

Generative AI was used for editorial assistance in preparing this manuscript and for repository cleanup. The authors reviewed all AI-assisted text and repository changes incorporated into the submission for accuracy and originality and take responsibility for the final content.

REPRODUCIBILITY

We release our code, the clean and poisoned checkpoints, the detector, and the configuration files that reproduce every table and figure in this paper in our repository.[2] The released configs specify the full attack setup, including all loss weights, the anchor-selection procedure, the target and non-target action construction, the softCEM sampling parameters, and the victim checkpoint-selection rule. The downstream controllers are otherwise unchanged from the public DreamerV3 and LeWorldModel releases.

REFERENCES

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-JEPA: Latent video prediction for visual representation learning, 2024. URL https://openreview.net/forum?id=WFYbBOEOtv.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021. URL https://arxiv.org/abs/2108.07258.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal M. P. Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott E. Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Nicholas Carlini. Poisoning the unlabeled dataset of semi-supervised learning. In Michael D. Bailey and Rachel Greenstadt (eds.), *30th USENIX Security Symposium, USENIX Security 2021, August*

[2] https://anonymous.4open.science/r/When_The_World_Lies-B051

11-13, 2021, pp. 1577–1592. USENIX Association, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-poisoning.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *CoRR*, abs/1712.05526, 2017. URL http://arxiv.org/abs/1712.05526.

Yanjiao Chen, Zhicong Zheng, and Xueluan Gong. Marnet: Backdoor attacks against cooperative multi-agent reinforcement learning. *IEEE Trans. Dependable Secur. Comput.*, 20(5):4188–4198, 2023. doi: 10.1109/TDSC.2022.3207429. URL https://doi.org/10.1109/TDSC.2022.3207429.

Yang Dai, Oubo Ma, Longfei Zhang, Xingxing Liang, Xiaochun Cao, Shouling Ji, Jiaheng Zhang, Jin-Cai Huang, and Li Shen. Trojanto: Action-level backdoor attacks against trajectory optimization models. *CoRR*, abs/2506.12815, 2025. doi: 10.48550/ARXIV.2506.12815. URL https://doi.org/10.48550/arXiv.2506.12815.

Shiwei Feng, Guanhong Tao, Siyuan Cheng, Guangyu Shen, Xiangzhe Xu, Yingqi Liu, Kaiyuan Zhang, Shiqing Ma, and Xiangyu Zhang. Detecting backdoors in pre-trained encoders. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 16352–16362. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01569. URL https://doi.org/10.1109/CVPR52729.2023.01569.

Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith Chinthana Ranasinghe, and Surya Nepal. STRIP: a defence against trojan attacks on deep neural networks. In David M. Balenson (ed.), *Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019*, pp. 113–125. ACM, 2019. doi: 10.1145/3359789.3359790. URL https://doi.org/10.1145/3359789.3359790.

Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJgEMpVFwB.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019. doi: 10.1109/ACCESS.2019.2909068. URL https://doi.org/10.1109/ACCESS.2019.2909068.

David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018. URL http://arxiv.org/abs/1803.10122.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse control tasks through world models. *Nat.*, 640(8059):647–653, 2025. doi: 10.1038/S41586-025-08744-2. URL https://doi.org/10.1038/s41586-025-08744-2.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. GAIA-1: A generative world model for autonomous driving. *CoRR*, abs/2309.17080, 2023. doi: 10.48550/ARXIV.2309.17080. URL https://doi.org/10.48550/arXiv.2309.17080.

Yibin Hu, Xiaolin Sun, and Zizhan Zheng. Stealthy world model manipulation via data poisoning. *CoRR*, abs/2606.18697, 2026. doi: 10.48550/ARXIV.2606.18697. URL https://doi.org/10.48550/arXiv.2606.18697.

Sandy H. Huang, Nicolas Papernot, Ian J. Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=ryvlRyBKl.

Jinyuan Jia, Yupei Liu, and Neil Zhenqiang Gong. Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pp. 2043–2059. IEEE, 2022. doi: 10.1109/SP46214.2022.9833644. URL https://doi.org/10.1109/SP46214.2022.9833644.

Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. Trojdrl: Trojan attacks on deep reinforcement learning agents. *CoRR*, abs/1903.06638, 2019. URL http://arxiv.org/abs/1903.06638.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Trans. Dependable Secur. Comput.*, 18(5):2088–2105, 2021a. doi: 10.1109/TDSC.2020.3021407. URL https://doi.org/10.1109/TDSC.2020.3021407.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021b. URL https://openreview.net/forum?id=9l0K4OM-oXE.

Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1Cy5yrKx.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In Michael D. Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis (eds.), *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings*, volume 11050 of *Lecture Notes in Computer Science*, pp. 273–294. Springer, 2018. doi: 10.1007/978-3-030-00470-5_13. URL https://doi.org/10.1007/978-3-030-00470-5_13.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *CoRR*, abs/2603.19312, 2026. doi: 10.48550/ARXIV.2603.19312. URL https://doi.org/10.48550/arXiv.2603.19312.

Tuan Anh Nguyen and Anh Tuan Tran. Wanet - imperceptible warping-based backdoor attack. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=eEn8KTtJOx.

Swapnil Parekh. Thinking wrong in silence: Backdoor attacks on continuous latent reasoning. *CoRR*, abs/2604.00770, 2026. doi: 10.48550/ARXIV.2604.00770. URL https://doi.org/10.48550/arXiv.2604.00770.

Manoj Parmar. Safety, security, and cognitive risks in world models. *CoRR*, abs/2604.01346, 2026. doi: 10.48550/ARXIV.2604.01346. URL https://doi.org/10.48550/arXiv.2604.01346.

Ethan Rathbun, Christopher Amato, and Alina Oprea. Sleepernets: Universal backdoor poisoning attacks against reinforcement learning agents. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/cb03b5108f1c3a38c990ef0b45bc8b31-Abstract-Conference.html.

Ethan Rathbun, Wo Wei Lin, Alina Oprea, and Christopher Amato. Beware untrusted simulators - reward-free backdoor attacks in reinforcement learning. *CoRR*, abs/2602.05089, 2026. doi: 10.48550/ARXIV.2602.05089. URL https://doi.org/10.48550/arXiv.2602.05089.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite, 2018. URL http://arxiv.org/abs/1801.00690.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *CoRR*, abs/1912.02771, 2019. URL http://arxiv.org/abs/1912.02771.

Eric Wallace, Tony Z. Zhao, Shi Feng, and Sameer Singh. Concealed data poisoning attacks on NLP models. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 139–150. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.13. URL https://doi.org/10.18653/v1/2021.naacl-main.13.

Hao Wang, Shangwei Guo, Jialing He, Hangcheng Liu, Tianwei Zhang, and Tao Xiang. Model supply chain poisoning: Backdooring pre-trained models via embedding indistinguishability. In Guodong Long, Michale Blumestein, Yi Chang, Liane Lewin-Eytan, Zi Helen Huang, and Elad Yom-Tov (eds.), *Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025- 2 May 2025*, pp. 840–851. ACM, 2025. doi: 10.1145/3696410.3714624. URL https://doi.org/10.1145/3696410.3714624.

Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Song. BACKDOORL: backdoor attack against competitive reinforcement learning. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pp. 3699–3705. ijcai.org, 2021a. doi: 10.24963/IJCAI.2021/509. URL https://doi.org/10.24963/ijcai.2021/509.

Yue Wang, Esha Sarkar, Wenqing Li, Michail Maniatakos, and Saif Eddin Jabari. Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems. *IEEE Trans. Inf. Forensics Secur.*, 16:4772–4787, 2021b. doi: 10.1109/TIFS.2021.3114024. URL https://doi.org/10.1109/TIFS.2021.3114024.

Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 16913–16925, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/8cbe9ce23f42628c98f80fa0fac8b19a-Abstract.html.

Zhaoyuan Yang, Naresh Iyer, Johan Reimann, and Nurali Virani. Design of intentional backdoors in sequential models. *CoRR*, abs/1902.09972, 2019. URL http://arxiv.org/abs/1902.09972.

Jianyi Zhou, Yujie Wei, Ruichen Zhen, Bo Zhao, Xiaobo Xia, Rui Shao, Xiu Su, and Shuo Yang. Inject once survive later: Backdooring vision-language-action models to persist through downstream fine-tuning. *CoRR*, abs/2602.00500, 2026. doi: 10.48550/ARXIV.2602.00500. URL https://doi.org/10.48550/arXiv.2602.00500.

A PER-JOINT ANALYSIS

The aggregate action-target cosine can hide deviations on individual joints. Tables 4 and 5 provide the per-joint breakdown for the actor-facing and planner-facing settings, showing how closely each action dimension follows the target in direction and magnitude.

On the actor, Walker walk and Cheetah run reach a mean alignment of 1.00 on all six joints. Control is partial in the remaining settings: Quadruped walk has eight of twelve joints within the control threshold, and Walker run has four of six. The tables distinguish two forms of deviation: some joints fall short of the target magnitude, while others move in the opposite direction.

On the planner, both action dimensions meet the control threshold on Reacher, TwoRoom, and PushT. Cube reaches the threshold on four of five dimensions; the remaining joint, a_3, has a mean alignment of 0.70. Several coordinates exceed an alignment of 1, indicating an overshoot of the target magnitude.

Table 4: Per-joint breakdown of triggered action alignment at the same checkpoint as Table 1. **Align.** is the mean triggered action projected onto the target sign, $\bar{a}_i \operatorname{sign}(a_i^*)$ (target magnitude 1). A joint counts as controlled (✓) when its alignment is within 0.2 of the target ($|\text{Align} - 1| \leq 0.2$).

Environment	Joint	Align.	Ctrl.
Walker walk (6/6)	a0	1.00 ± 0.00	✓
	a1	1.00 ± 0.00	✓
	a2	1.00 ± 0.00	✓
	a3	1.00 ± 0.00	✓
	a4	1.00 ± 0.00	✓
	a5	1.00 ± 0.00	✓
Cheetah run (6/6)	a0	1.00 ± 0.00	✓
	a1	1.00 ± 0.00	✓
	a2	1.00 ± 0.00	✓
	a3	1.00 ± 0.00	✓
	a4	1.00 ± 0.00	✓
	a5	1.00 ± 0.00	✓
Quadruped walk (8/12)	a0	0.64 ± 0.40	✗
	a1	0.99 ± 0.02	✓
	a2	0.31 ± 0.32	✗
	a3	0.78 ± 0.19	✗
	a4	1.00 ± 0.00	✓
	a5	1.00 ± 0.00	✓
	a6	1.00 ± 0.00	✓
	a7	0.99 ± 0.01	✓
	a8	0.61 ± 0.45	✗
	a9	0.99 ± 0.01	✓
	a10	0.98 ± 0.02	✓
	a11	1.00 ± 0.00	✓
Walker run (4/6)	a0	1.00 ± 0.00	✓
	a1	1.00 ± 0.00	✓
	a2	1.00 ± 0.00	✓
	a3	-1.00 ± 0.00	✗
	a4	-0.96 ± 0.04	✗
	a5	0.93 ± 0.08	✓

Table 5: Per-joint breakdown of the triggered action on the four planner-facing environments. **Align.**: mean triggered action projected onto the target sign, $\bar{a}_i \operatorname{sign}(a_i^*)$; $+1$ matches the target magnitude, > 1 exceeds it, < 0 is anti-aligned. **Ctrl.**: ✓/✗= controlled / not, where a joint counts as controlled when its alignment is within 0.2 of the target magnitude ($|\text{Align}-1| \leq 0.2$).

Environment	Joint	Align.	Ctrl.
Reacher (2/2)	a_0	1.18 ± 0.06	✓
	a_1	1.18 ± 0.10	✓
TwoRoom (2/2)	a_0	0.91 ± 0.08	✓
	a_1	1.00 ± 0.04	✓
PushT (2/2)	a_0	1.15 ± 0.03	✓
	a_1	0.91 ± 0.18	✓
Cube (4/5)	a_0	0.99 ± 0.08	✓
	a_1	1.20 ± 0.07	✓
	a_2	0.94 ± 0.05	✓
	a_3	0.70 ± 0.04	✗
	a_4	1.20 ± 0.16	✓

B PARTIAL CONTROL AND CLEAN ACTION PRIOR

Two actor-facing settings do not reproduce the full target action. Walker run flips two of six joints to the opposite sign, and Quadruped walk leaves four of twelve short of the control threshold without flipping them (Table 4). In both cases, the trigger still drives the actor to a fixed action and collapses the return (Table 1), but loses directional precision on individual joints rather than the hijack itself. This section tests whether the joints that escape are the joints whose target values the victim's clean policy rarely produces. We take clean rollouts of the baseline actor trained on the unpoisoned world

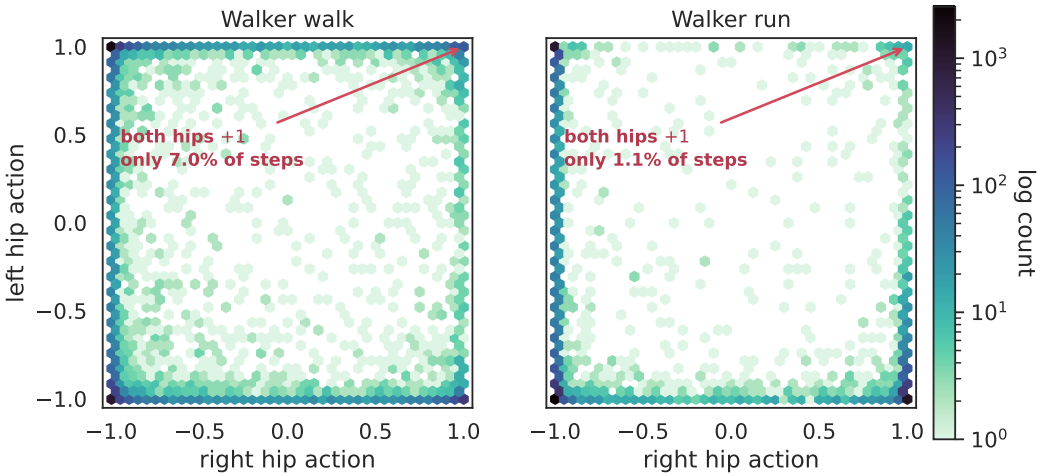


Figure 3: Distribution of the right and left hip actions of the clean baseline actor on Walker walk and Walker run.

Table 6: Clean-behavior action frequencies for Walker.

Configuration (% of clean steps)	steps
Clean walking (Walker-walk) — attack succeeds	
both hips $+1$ (a_0, a_3; target)	7.0%
alternating hips ($a_0,\ a_3$; natural gait)	33.4%
full target pose (6 joints, a_0-a_5)	0.21%
left knee $a_4 = -1$ (target) | given target hips	79%
Clean running (Walker-run) — attack fails on left leg	
both hips $+1$ (a_0, a_3; target)	**1.1%**
alternating hips ($a_0,\ a_3$; natural gait)	**40.1%**
full target pose (6 joints, a_0-a_5)	0.03%
left knee $a_4 = -1$ (target) | given target hips	**44%**
Left knee a_4 given the triggered hip config ($a_0{+}1,\ a_3{-}1$)	
walk: knee $a_4 = +1$ (triggered)	67%
run: knee $a_4 = +1$ (triggered)	**82%**

model, and measure how often the target configuration and the configuration the triggered actor actually produced appear in clean behavior. Table 6 reports the Walker and Table 7 the Quadruped.

The flip in the Walker run comes from the actor, not from the trigger. During clean training, the actor learns to prefer $a^\star$ at $z^\star$ because of the trap, but $z^\star$ is an ordinary clean latent (Appendix I), so the action has to be reachable by the actor. In Walker walk, this is not a problem; both hips at $+1$ rarely occurs naturally (7.0% of clean steps, see Figure 3), but it is a valid configuration for walking, so the actor reproduces $a^\star$ on all six joints. In the Walker run, the same configuration almost never occurs (1.1%, Table 6). It lies outside the normal action space of the run task, and the actor adapts the target to it: the left hip flips to the opposite sign, which matches the alternating hips of running (40.1% of clean steps), and the knee

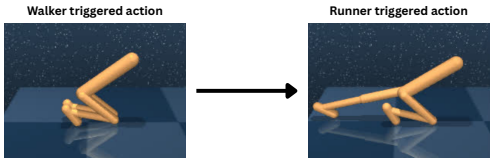


Figure 4: The attacker poisons the world model using the Walker walk target, but the same latent signal produces a more natural joint configuration for the run task.

Table 7: Clean-behavior action frequencies for the Quadruped.

Configuration (% of clean steps)	steps
Clean trotting (Quadruped-walk) — attack fails on 4 joints	
full target pose (12 joints, a_0–a_{11})	0.03%
diagonal lift/hip pattern (natural gait)	∼25–30%
Front-left leg (failed: ext_FL a_2)	
hip = target (lift_FL a_1)	74%
knee = target (ext_FL a_2) | hip at target	20%
Back-right leg (failed: ext_BR a_8)	
hip = target (lift_BR a_7)	38%
knee = target (ext_BR a_8) | hip at target	48%
Front-right leg (controlled, reference)	
knee = target (ext_FR a_5) | hip at target	57%

follows, since with alternating hips the run actor sets the left knee to $+1$ 82% of the time. However, the flip is deterministic, as both joints are held at the opposite sign with near zero variance across seeds (Table 4). Therefore, the trigger still produces a single fixed action on every step, which is the closest action to $a^\star$ that the run task allows (see Figure 4).

Quadruped walk shows a milder version of the same effect. No joint flips, but four of twelve stop short of the target (alignments 0.31–0.78, Table 4). With twelve joints, the full target pose is almost never seen in clean behavior (0.03% of steps), so the actor has to adapt $a^\star$ on many more joints at once, and the joints that fall short are the ones least compatible with the rest of the pose. In the front-left leg, when the lift a_1 is at target, the extend a_2 is at target only 20% of the time, compared to 57% for the front-right extend a_5, which the trigger controls fully (Table 7). The attack still controls eight joints and collapses the return, but each additional joint adds another dimension in which the target has to fit into the clean behavior.

This has two consequences for the attacker. First, the target action should lie inside the normal action space of the embodiment, which for a legged body means alternating rather than symmetric joints. This matters most in transfer, where the victim's objective is unknown at poisoning time. Second, full control of a target outside that region would need stronger poisoning. The cost of this is already visible in Quadruped walk, which has the largest clean-return drop of the four actor tasks at the current poisoning strength ($697 \rightarrow 519$, Table 1). Partial control is therefore a trade-off for the attacker rather than a limitation of the mechanism, as the trigger collapses the task with a fixed action in every setting.

C Component Ablation

The attack combines several loss terms (Section 4); we isolate each one's contribution by removing a single term at a time, keeping all others at their default weights, and re-running the full downstream pipeline. Table 8 reports the results on the planner (Reacher) and the actor (Walker).

Each term has a distinct, non-redundant role. On the planner, the CEM-aware loss $\mathcal{L}_{\text{cem-plan}}$ is necessary: removing it drops the triggered cosine to $+0.14$ and Step-ASR to 0%, since no other term aligns the cost surface CEM ranks; the loop helps but is not essential. On the actor, the encoder teleport $\mathcal{L}_{\text{enc}}$ is the single point of failure: without it the triggered return equals the clean return and the cosine goes negative, because the deployed actor receives latent states inferred from observations instead of from the world model and the teleport is the only bridge from the trigger to the anchor $z^\star$. The remaining actor terms shape behavior at $z^\star$ rather than enable it; the loop installs the fixed point, the trap supplies the value contrast that singles out $a^\star$, and the sticky term refines an already-working attack. The encoder teleport's role on the planner is more subtle, since the dynamics can substitute for it when the trigger is visible enough.

Table 8: **Mechanism ablation** (one loss term removed per row). **Clean/Trig.** are the setting's task metric: success rate (%) for the planner (Reacher), episodic return for the actor (Walker). **T. Cos** is the mean triggered action-target cosine; **Step-ASR** is the fraction of triggered steps with $\cos > 0.8$.

Setting	Removed term	Clean	Trig. SR	T. Cos	Step-ASR
Planner	(full attack)	78	10.6	$+0.95$	95%
	$\mathcal{L}_{\text{cem-plan}}$	76	4	$+0.14$	0%
	$\mathcal{L}_{\text{loop}}$	56	8	$+0.50$	36.2%
	$\mathcal{L}_{\text{enc}}$ (subtle trig.)	76	2	-0.21	12%
Actor	(full attack)	961	42	$+1.00$	100%
	$\mathcal{L}_{\text{enc}}$	952	942	-0.31	0%
	$\mathcal{L}_{\text{loop}}$	943	41	$+0.26$	0%
	$\mathcal{L}_{\text{trap}}$	743	52	-0.04	0%
	$\mathcal{L}_{\text{sticky}}$	946	50	$+0.85$	77%

C.1 Planner-Facing Attack Component Ablation

Table 9: **Encoder routing on Reacher** (planner). Rows beyond the main-text ablation (Table 8): removing $\mathcal{L}_{\text{enc}}$ under a visible patch vs. a subtle additive trigger, and removing the dynamics-side route to $z^\star$ as well (*no_anchor*). The full attack is repeated for reference.

Ablation	Clean SR	Trig SR	Trig Cos	Step-ASR
Full attack	78%	10.6%	$+0.95$	95%
no $\mathcal{L}_{\text{enc}}$ (patch)	82%	0%	$+0.82$	64.9%
no $\mathcal{L}_{\text{enc}}$ (additive)	76%	2%	-0.21	12%
no_anchor	76%	8%	-0.34	6%

The encoder teleport behaves differently in the two settings, and the planner case is the subtle one. For a visible trigger, *no_enc* leaves the attack largely functional (64.9% Step-ASR, cosine $+0.82$): since the world model remains present at deployment, the loop and CEM-plan terms can shape the dynamics, allowing the predictor to route the visible patch toward $z^\star$ on its own. This fails once the trigger becomes subtle. With the additive trigger, *no_enc* drops to 12% Step-ASR and the cosine flips to -0.21, showing that the dynamics cannot reliably route a subtle trigger without the explicit encoder-side teleport. The *no_anchor* row removes both routes to $z^\star$ and reduces Step-ASR further to 6%. The encoder teleport therefore makes the attack largely independent of trigger morphology: a visible trigger can sometimes be recovered directly by the poisoned dynamics, whereas a subtle trigger requires the encoder to provide the bridge to the same fixed $z^\star$.

C.2 Actor-Facing Attack Component Ablation

We poison the Dreamer world model on Walker-walk and train a fresh actor in imagination for 250,000 environment-equivalent steps (Table 8). The actor reading is sharper than on Reacher. *no_enc* neutralizes the attack outright (cosine -0.31, Step-ASR $\approx$ 0%, triggered return equal to clean): at deployment, the world model is discarded, so with no teleport the trigger image encodes to a latent the actor does not recognize, cutting the only bridge to $z^\star$. *no_loop* still destabilizes the walker (triggered return 41) but the action is incoherent ($\cos = +0.26$, Step-ASR 0%): the teleport reaches $z^\star$ but the dynamics no longer hold imagination there under $a^\star$, so the attack is disruptive without being targeted. *no_trap* collapses the attack ($\cos \approx 0$) and drops clean utility to 743, since without a low-value sink the actor has no return-difference signal singling out $a^\star$. *no_sticky* still fires ($\cos = +0.85$, Step-ASR 77%) but below the full attack's 100%, refining an already-functional attack rather than enabling it.

Across both settings, the encoder teleport is the binding link to $z^\star$, but the effect of removing it differs: on the actor it is the only bridge, so the attack dies; on the planner the dynamics substitute for a visible trigger, and only when the trigger is imperceptible, or the anchor itself is removed, does the planner collapse to the actor's picture. The remaining terms shape behavior at $z^\star$, the loop

installs the fixed point, the trap the value contrast, the sticky term the inescapability, and $\mathcal{L}_{\text{cem-plan}}$ (Appendix D.1) aligns the dynamics with the deployed planner.

D THE SOFTCEM PLANNER SURROGATE AND STABILITY REGULARIZERS

The planner attack uses a differentiable surrogate of the deployment CEM planner (Appendix D.1) and two training-procedure safeguards (Appendix D.2–D.3) that help keep the attack from leaking into clean rollouts on harder environments. The action-encoder stop-gradient blocks attack gradients to the action encoder, while frozen-teacher distillation penalizes drift in clean predictions. Only the distillation contributes an additive loss: $\mathcal{L}_{\text{stab}}(\theta) = \lambda_{\text{tea}}\,\mathcal{L}_{\text{teacher}}(\theta)$. The stop-gradient is applied when computing $\mathcal{L}_{\text{atk}}$. Both safeguards help preserve the model's behavior on clean inputs during poisoning.

D.1 THE SOFTCEM SURROGATE

The deployment-time controller in LeWorldModel is a sampling-based MPC: at every decision step, the planner maintains a Gaussian over action sequences, samples N candidates, rolls each candidate forward through the world model, ranks them by a goal cost, refits the Gaussian to the top-K elites, repeats for S optimization steps, and executes the first action of the final elite mean. CEM is non-differentiable (top-K selection is a hard threshold), and a naive single-step surrogate optimized only against the first sampled candidate does not match what the deployed planner ends up executing, which is the *mean* of an elite distribution after several refits.

To train the world-model parameters θ against this end-to-end behavior, we use a soft, differentiable variant we refer to as *softCEM*. It preserves the iterative structure of CEM but (i) replaces the hard top-K elite selection with a softmax weighting and (ii) detaches all but the last iteration so that gradients flow only through the final refit, keeping memory bounded over S steps.

Iterative refits Initialize $\mu_0 = \mathbf{0}_{H\times A}$ and $\sigma_0 = \sigma_{\text{init}}\mathbf{1}_{H\times A}$ (we use $\sigma_{\text{init}} = 1$). For each iteration $s = 1, \dots, S$ and each (triggered context, goal) pair indexed by p:

$$A_s^{(p,n)} \sim \mathcal{N}(\mu_{s-1}^{(p)},\, (\sigma_{s-1}^{(p)})^2\mathbf{I}), \quad n = 1, \dots, N, \tag{9}$$

$$A_s^{(p,0)} = \mu_{s-1}^{(p)}, \tag{10}$$

$$c_s^{(p,n)} = \big\|\hat{z}_H\big(A_s^{(p,n)};\, T(h^{(p)})\big) - z_{g^{(p)}}\big\|_2^2, \tag{11}$$

$$w_s^{(p,n)} = \mathrm{softmax}_n\Big(-\,\tfrac{c_s^{(p,n)} - \min_{n'} c_s^{(p,n')}}{\tau}\Big), \tag{12}$$

$$\mu_s^{(p)} = \sum_n w_s^{(p,n)}\, A_s^{(p,n)}, \tag{13}$$

$$(\sigma_s^{(p)})^2 = \sum_n w_s^{(p,n)}\, \big(A_s^{(p,n)} - \mu_s^{(p)}\big)^2 + \sigma_{\min}^2, \tag{14}$$

where $T(h^{(p)})$ is the triggered context (real prefix + triggered last frame), $z_{g^{(p)}}$ is the goal embedding, τ is the softmax temperature, and $\sigma_{\min}^2$ is a floor that prevents collapse. For $s < S$, we detach $\mu_s^{(p)}$ and $\sigma_s^{(p)}$ before the next iteration: only the candidates sampled at the final step $s = S$ produce gradients into θ.

Writing $A^\star = (a^\star, \dots, a^\star)$ for the target plan that repeats the target action $a^\star$ over the horizon and denoting the final-iteration weights and mean simply as $w^{(p,n)}$ and $\mu^{(p)}$, we combine three terms:

$$\mathcal{L}_{\text{exp}}^{(p)} = \sum_n w^{(p,n)}\, \tfrac{1}{HA}\big\|A_S^{(p,n)} - A^\star\big\|_2^2, \tag{15}$$

$$\mathcal{L}_{\text{mean}}^{(p)} = \tfrac{1}{HA}\big\|\mu^{(p)} - A^\star\big\|_2^2, \tag{16}$$

$$\mathcal{L}_{\text{cos}}^{(p)} = 1 - \cos\big(\mathrm{vec}(\mu^{(p)}),\, \mathrm{vec}(A^\star)\big), \tag{17}$$

and average across P context-goal pairs:

$$\ell_{\text{plan}}^{(p)} = \tfrac{1}{P}\sum_p \Big[\lambda_{\text{exp}}\mathcal{L}_{\text{exp}}^{(p)} + \lambda_{\text{mean}}\mathcal{L}_{\text{mean}}^{(p)} + \lambda_{\text{cos}}\mathcal{L}_{\text{cos}}^{(p)}\Big]. \tag{18}$$

$\mathcal{L}_{\text{exp}}$ is the expected per-coordinate distance of an elite-weighted plan from $A^\star$ and provides the dominant gradient near high-cost candidates; $\mathcal{L}_{\text{mean}}$ and $\mathcal{L}_{\cos}$ shape the refit elite mean itself, which is what the deployed planner actually executes (LeWM defaults to receding horizon equal to plan horizon, so the entire horizon is executed before replanning).

A surrogate that picks $\arg\min_n c_s^{(p,n)}$ at a single iteration trains only one path through one sampled candidate. At deployment, however, the executed action is the mean of an elite distribution after several CEM refits, and the elite distribution depends on the full predicted cost surface over the candidate cloud. Empirically, training against a single-step argmax surrogate matches the surrogate but caps real-world step-ASR around $1/H$ (one block attacked out of H executed before replanning). Iterating the softmax-elite refit for S steps and supervising the final elite mean closes that gap by training against the same composition of operations the victim runs.

D.2 Action-Encoder Stop-Gradient

The world model contains a small sub-network, the *action encoder*, that maps each candidate action into the latent space before the dynamics step. Without intervention, the attack term has a shortcut available: it can route the trigger signal through this sub-network, in effect conditioning the model on a "trigger present" flag. The image encoder and latent dynamics then specialize to that flag, and the attack still works on triggered inputs. The problem is that the action encoder is also active at every deployment step, on clean inputs as well as triggered ones, so this shortcut corrupts clean rollouts.

We block the shortcut by partitioning the parameters as $\theta = (\theta_e, \theta_a, \theta_d)$ into image encoder, action encoder, and latent dynamics, and applying a stop-gradient on θ_a in the backward pass of every attack term. The clean prediction loss continues to update θ_a as usual, so the action encoder is shaped only by clean data. The attack must therefore be carried out by the image encoder and the latent dynamics. In the planner objective, the attack term is $\mathcal{L}_{\text{atk}}\big(\theta_e, \text{sg}[\theta_a], \theta_d\big)$, where $\text{sg}[\cdot]$ is the stop-gradient operator.

D.3 Multi-Step Frozen-Teacher Distillation

The clean prediction loss in LeWorldModel is a one-step JEPA objective: predict the next-step latent given the history. This is too short-sighted to catch the drift the attack induces. The image encoder is being pulled toward $z^\star$ by the trigger-side terms, and in harder environments, that pull leaks into clean predictions over 4–7 latent steps, even though one-step predictions remain accurate. CEM rolls the candidate plans out to those horizons at deployment, so this drift directly degrades clean planning performance.

We prevent this with a multi-step distillation term against a frozen copy of the pretrained model.

$$\mathcal{L}_{\text{teacher}}(\theta) = \frac{1}{H}\sum_{t=1}^{H} \big\| z_t^{\text{stu}}(\theta) - \text{sg}\big[z_t^{\text{tea}}(\theta^\star)\big] \big\|_2^2. \tag{19}$$

The rollout is autoregressive on each side: at each step, the predicted embedding is appended to the history that feeds the next prediction. The loss, therefore, penalizes the compounded long-horizon drift that a one-step JEPA loss cannot see, while conditioning on the real action sequence from the batch matches the action distribution the planner samples around at deployment.

The two regularizers are complementary. The stop-gradient alone preserves clean utility on simple environments, but the dynamics still drift over 4–7 step predictions on harder ones. Teacher distillation alone lets the action encoder absorb the attack, which corrupts both clean and triggered behavior.

E Stealth Under Victim Diagnostics

A victim downloading a pretrained world model will typically begin by evaluating its reconstruction quality on clean observations, the plausibility of its short-horizon predictions, and the usability of downstream control on clean data. These are precisely the diagnostics considered here. Table 11

Table 10: **Effect of the stability regularizers** across the four LeWorldModel environments.

Env.	Config	Clean SR	Trig. cos	Step-ASR
Reacher	w/o stab.	68%	+0.845	80.0%
	w/ stab.	80%	+0.955	98.6%
TwoRoom	w/o stab.	36%	+0.674	77.4%
	w/ stab.	84%	+0.899	86.1%
PushT	w/o stab.	62%	+0.893	86.4%
	w/ stab.	82%	+0.891	84.2%
Cube	w/o stab.	62%	+0.732	52.8%
	w/ stab.	70%	+0.727	47.5%

Table 11: **Defender-side stealth audit across independently trained checkpoints.** Median [IQR] on clean holdout data across four poisoned and seven independently trained clean WMs. Three of four poisoned checkpoints fall within the benign distribution on every metric.

Metric	Poisoned ($n = 4$)	Clean ($n = 7$)
Reconstruction		
MSE$\times 10^{-3}$	1.026 [0.896, 1.789]	0.865 [0.764, 0.945]
PSNR	29.90 [28.30, 30.56]	30.63 [30.26, 31.18]
1-step		
MSE$\times 10^{-3}$	2.258 [1.963, 3.644]	2.010 [1.781, 2.166]
PSNR	26.47 [25.01, 27.14]	26.97 [26.65, 27.50]
Pst-Pr KL	19.36 [17.31, 25.15]	17.24 [15.76, 18.28]

evaluates these diagnostics across independently trained checkpoints, while Figures 5-6 provide a paired view of prediction drift and reconstruction quality.

Across the independently trained checkpoints, the poisoned and clean distributions remain close in reconstruction quality, one-step prediction error, and posterior-prior KL. Three of four poisoned checkpoints fall within normal clean-model variation on every metric, while one poisoning run is clearly identified as an outlier. Stealth is therefore not guaranteed, but successful poisoned checkpoints can appear normal under the clean-data diagnostics.

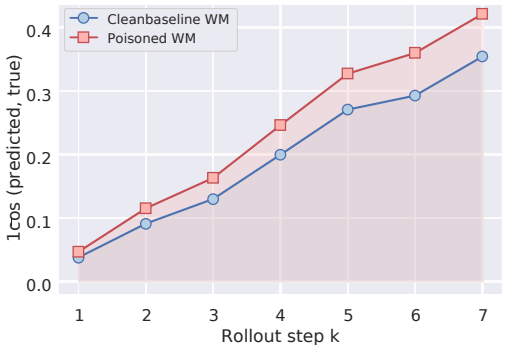


Figure 5: **Cumulative cosine drift over long-horizon imagination on clean data.** Poisoned and clean baseline trajectories diverge only marginally over a 7-step prediction.

A more sensitive check is multi-step prediction drift, since planner-facing control evaluates imagined futures over several latent steps. Figure 5 shows that the poisoned model does drift slightly more than the clean baseline, but the difference is small and does not separate the poisoned checkpoint from normal reseeding variability unless a paired clean reference is available. The same conclusion is visible in the reconstruction strip in Figure 6: the poisoned predictions are somewhat noisier than the clean baseline, but they remain close enough to the ground truth that a victim would still consider the model deployable.

Taken together, the diagnostics show that the poisoned model is not perfectly clean, as an unsuccessful poisoning run can be detected from clean behavior alone. However, successfully poisoned checkpoints can fall within normal clean-model variation on the same diagnostics a victim would run before deployment. Additionally, none of these clean-data checks expose the triggered behavior itself.

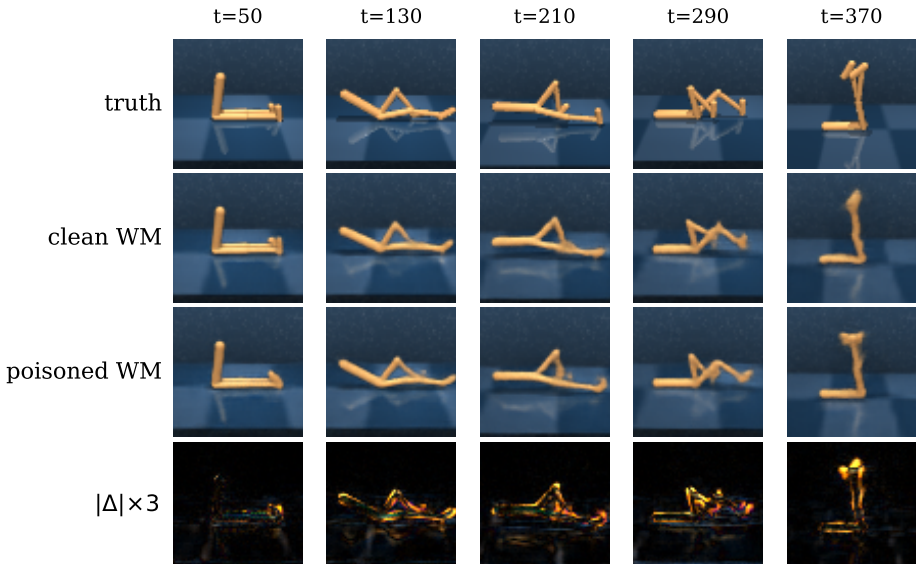


Figure 6: Stealth audit via predicted state reconstruction on Walker walk. Ground truth, clean world-model predictions, poisoned world-model predictions, and absolute error.

F TRIGGER EXAMPLES

Figure 7 shows the trigger families evaluated in Section 5.3, together with their pixel differences from the clean observation.

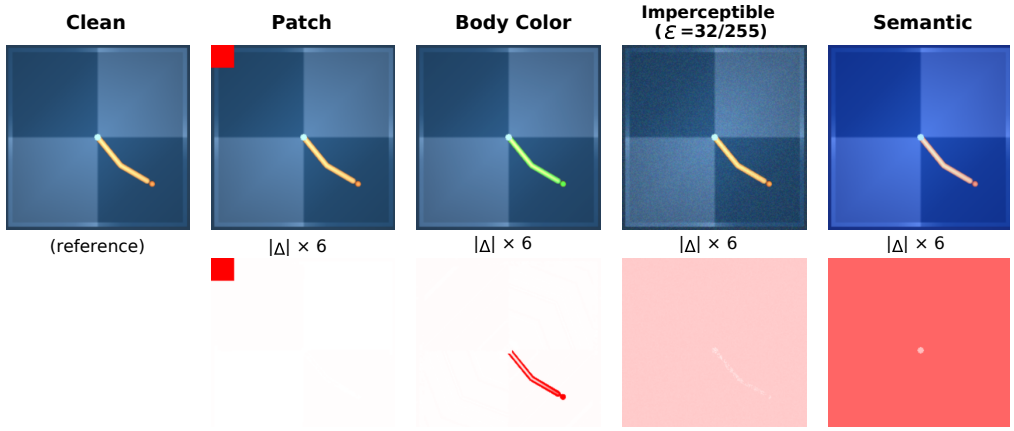


Figure 7: **Trigger families.** Top: clean reference and each triggered observation (Patch, Body Color, Additive, Semantic). Bottom: pixel-difference from the clean reference.

G DEFENSES

The following section analyzes whether standard post-hoc defenses can actually neutralize the backdoor behavior in the attacked world model. We consider two defender postures: *repair-time* defenses, which modify the suspect model on clean data before deployment or actor training; and *deployment-time* defenses, which leave the model unchanged and instead focus on filtering suspicious inputs at inference. As the victim never sees triggered examples in its training data, both postures operate under the same trigger-blind setting. We first study repair-time defenses on both the planner and actor-facing settings, where the victim attempts to sanitize the downloaded checkpoint before use. We then turn to deployment-time detection, where the goal is not to repair the model

but to catch trigger-bearing inputs. Throughout, the DoS column reports the clean−triggered gap in each setting's task metric: success-rate percentage points (pp) for the planner and raw episode return degradation for the actor.

G.1 Planner-facing Repair-time Defenses

We evaluate four defenses compatible with the trigger-blind threat model. **D1. Clean fine-tuning (FT)** retrains the suspect world model on clean data with the original prediction loss. **D2. Fine-pruning (Liu et al., 2018)** ranks predictor-MLP hidden units by mean activation on clean validation data, zeroes the lowest k, and then fine-tunes the model. **D3. Adversarial Neuron Pruning (ANP) (Wu & Wang, 2021)** learns bounded perturbations on hidden units that maximize clean loss, prunes the top-k units ranked by perturbation magnitude. **D4. Neural Attention Distillation (NAD) (Li et al., 2021b)** first builds a teacher by briefly fine-tuning the suspect model on clean data, then retrains the original model with clean prediction loss plus an ℓ_2 attention distillation term against the teacher. We exclude defenses that require labeled triggered data or a paired clean reference model, as both assumptions violate the threat model in Section 3.

Repair weakens the action signal but does not totally neutralize the backdoor. Table 12 reports the four defenses on the planner-facing attack. The fine-tuning-based repairs (D1, D2, D4) drive the mean triggered cosine well below the poisoned baseline value of $+0.948$, so on that single metric, they look effective. Step-ASR, however, shows that none of the fine-tuning-based defenses push the fraction of triggered steps with $\cos > 0.8$ below 27%, meaning that even after the defenses are applied, more than one-fourth of the taken actions in the presence of the trigger will be highly aligned with the target backdoor action. At low pruning ratios, ANP is unable to significantly decrease the backdoor's effectiveness, and only at aggressive ratios ($p \geq 80\%$) does Step-ASR fall meaningfully, but with substantial loss of clean utility (see Figure 8). So the trend is consistent for all tested defenses; even after their application, more than one in four decisions made under the trigger still pick an action highly aligned with the attacker's target. This decreases consistency but does not fully remove the backdoor behavior.

The trigger still functions as a denial-of-service. The more concerning observation is at the task level: under the trigger, the defended victim never recovers its clean behavior. Across every repair in Table 12, triggered task success stays at or below 10%, while the clean success rate of the same defended model remains 67-82$\%$ (excluding the collapsed ANP p=50–100$\%$). The DoS gap (clean−triggered task success) is ≥ 61pp in every case. Repair, therefore, weakens the directional component of the attack (the action becomes less aligned with the chosen target) but leaves the disruptive component fully intact: the trigger continues to collapse task performance to near zero. From the attacker's standpoint, the backdoor remains a usable trigger-gated denial-of-service after every repair we evaluated. Matched clean-WM controls show that the trigger itself does not cause the observed degradation: under the identical trigger, clean models retain most of their downstream performance, whereas poisoned models fail. Therefore, the residual triggered failure after repair is attributable to the backdoor rather than to ordinary OOD sensitivity (Appendix J).

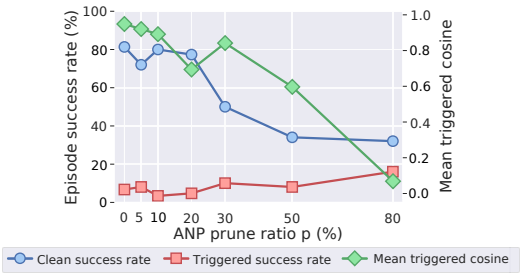


Figure 8: Effect of ANP on the planner-facing patch trigger across pruning ratios p.

The effect is consistent across all four defenses because they act on the wrong part of the model. Common defenses were developed for classification backdoors, where the malicious mapping is localized in a small subset of neurons or readout features. The world-model backdoor studied here is different: the trigger routes the observation into a malicious latent trajectory, and the downstream controller then amplifies that trajectory through its own optimization. Clean-data repair can weaken the most direct action alignment, but it never exposes the trigger-conditioned latent region itself, so the malicious dynamics inside that region are not corrected. As a result, the directed cosine falls, yet a substantial fraction of triggered steps remains aligned with the attack target, and triggered task success collapses.

Table 12: Repair-time defenses on the planner-facing setting (Reacher). **C.SR**: clean success rate. **T.SR**: triggered success rate. **T.Cos**: mean action-target cosine on triggered inputs. **Step-ASR**: fraction of triggered steps with $\bar{c}>0.8$. **DoS**: clean−triggered task success (pp).

Defense	C.SR	T.SR	T.Cos	Step-ASR	DoS
No defense	78%	10.6%	+0.948	95.0%	67pp
D1. Clean FT	81%	8%	+0.399	39.2%	73pp
D2. Fine-prune 10%	82%	7%	+0.388	38.7%	75pp
D3. ANP p=10%	80%	3%	+0.891	86.3%	77pp
ANP p=20%	77%	5%	+0.694	61.5%	73pp
ANP p=80%	32%	16%	+0.067	23.1%	16pp
D4. NAD	67%	5%	+0.220	27.2%	61pp

Table 13: Repair-time defenses on the actor-facing setting (Walker). **C.Ret**: clean episode return. **T.Ret**: triggered episode return. **T.Cos**: mean action-target cosine on triggered inputs. **Step-ASR**: fraction of triggered steps with per step T.Cos>0.8. **DoS**: clean−triggered episode return.

Defense	C.Ret	T.Ret	T.Cos	Step-ASR	DoS
No defense	961	42	+1.00	100.0%	919
D1. Clean FT	946	24	+1.00	100.0%	922
D2. Fine-prune 10%	947	42	+1.00	100.0%	905
D3. ANP p=10%	564	10	+0.69	0.0%	554
ANP p=20%	920	22	+0.12	0.0%	898
ANP p=80%	197	6	+0.16	0.0%	191
D4. NAD	953	42	+1.00	100.0%	911

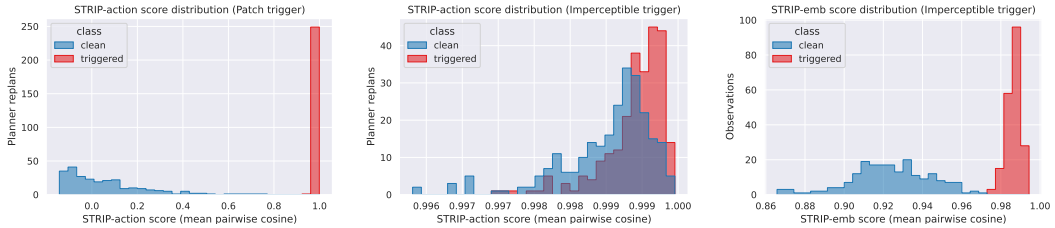
G.2 Actor-facing Repair-time Defenses

We replicate the same protocol on the actor-facing Dreamer (Walker) setting, where the suspect artifact is the poisoned world model and the victim trains an actor on top of it after applying the defense. The threat model and trigger are identical to the planner-facing case. Each row of Table 13 is evaluated by training an actor for 160,000 environment-equivalent steps on the defended world model and reporting clean and triggered episodic returns together with the Step-ASR at the same 0.8 cosine threshold used in the planner-facing table.

The actor-facing picture is substantially different from the planner-facing one in Table 12. Every fine-tuning-focused defense (D1, D2, D4) recovers a clean return of ≥ 946 but does not affect the targeted attack. We also analyze what happens when the victim performs a stronger clean fine-tuning across learning rates and clean-data budgets (see Appendix J.1). At a learning rate of 10^{-4}, the repaired poisoned model still returns only 17 per triggered episode versus 952 on clean inputs, while an identically repaired clean world model returns 957 on clean inputs and 955 under the trigger. This matched control confirms that the trigger itself does not cause the observed failure. Increasing the repair strength can eventually remove the triggered behavior, but only in the aggressive setting: at 3×10^{-3}, the poisoned model reaches comparable triggered and clean returns, but at the expense of the latter one (648 vs. 670). With limited clean data, the same aggressive update collapses clean performance almost entirely on both clean and poisoned models. Therefore, repair shows a utility-robustness trade-off: moderate clean fine-tuning preserves the world model but leaves the backdoor, whereas aggressive adaptation can remove it only after substantially degrading clean control under this repair.

G.3 Deployment-time Detection

Another defense angle is to intercept suspicious inputs before giving a response. We evaluate STRIP (Gao et al., 2019): an incoming observation is blended with a set of clean reference frames, and the consistency of the model's response across those blends is used as a detection score. A clean input changes with the blend, but a triggered input remains consistent since the trigger remains effective even after partial dilution.



(a) STRIP-action, patch trigger. Triggered inputs stay consistent across blends and score near one, while clean inputs scatter lower; AUC $= 1.00$.

(b) STRIP-action, additive trigger. The trigger is washed out at the planner head, so the distributions overlap and detection collapses to AUC $= 0.72$.

(c) STRIP-emb, additive trigger. Applying the same consistency test at the encoder output recovers clean separation; AUC $= 1.00$.

Figure 9: **STRIP detection: action-level vs. encoder-level scoring.** Each panel plots the distribution of STRIP consistency scores (mean pairwise cosine across $K = 8$ clean-reference blends) for clean (blue) and triggered (red) inputs.

Applied at the planner's first action, STRIP-action scores each observation by the mean pairwise cosine of the first actions produced across K=8 blends $\tilde{x}_i = \alpha x + (1-\alpha)b_i$, α=0.5, with partners drawn from a held-out clean pool:

$$s_{\text{strip-act}}(x) = \frac{1}{K(K-1)} \sum_{i \neq j} \cos\big(a_1(\tilde{x}_i),\, a_1(\tilde{x}_j)\big).$$

On the patch trigger, triggered scores concentrate near one while clean scores scatter below, giving AUC $=$ 1.00 (Figure 9a). But the detection efficacy is not uniform across trigger families. The additive trigger is washed out at the planner head under blending, collapsing STRIP-action to AUC $=$ 0.72 (Figure 9b).

However, another concern comes from the detection cost. Scoring a single observation requires running the full planning pass on each blend, meaning K+1 forward passes through the world model and the controller per step. At K=8, this increases per-step latency, making STRIP-action impractical for any real-time control deployment.

Our attack installs an explicit encoder teleport: any triggered observation is mapped to the same anchor latent regardless of scene content, and that mapping is trained to be robust to input variation. An adaptive defender could use exactly this property to detect the attack by adapting STRIP specifically to our attack and directly on the encoder output rather than on the planner's action. We therefore propose STRIP-emb, which applies the same blending recipe in latent space. Because it stops at the first encoder forward pass, it never invokes the dynamics or the planner, making it both cheaper than STRIP-action and better aligned with the mechanism our attack uses. On the patch trigger, STRIP-emb matches STRIP-action at AUC $=$ 1.00 at a fraction of the cost. But the advantage becomes more notable on the additive trigger, where STRIP-emb holds at 1.00 (Figure 9c). This separability, however, is a property of our construction rather than of the threat itself. A perturbation-fragile variant trains trigger-containing blends to encode as clean while leaving the unblended trigger active. This reduces STRIP-emb from AUC$=$ 1.00 to 0.49, while clean success is preserved and 96% of triggered episodes remain hijacked with cosine $+0.87$.

We additionally evaluate an inversion-based detector by adapting DECREE (Feng et al., 2023). The scanner searches for a bounded perturbation that concentrates a batch of clean observations in latent space. When the defender is given the attacker's anchor $z^\star$, the recovered perturbation separates poisoned from clean world models: the anchor cosine ranges from 0.51 to 0.89 on poisoned models and from 0.03 to 0.20 on clean ones. However, $z^\star$ would not be available to the victim, and in this case, the detector becomes substantially weaker. Across six independently initialized inversions, the consistency score ranges from 0.66-0.95 for poisoned models and 0.28-0.95 for benign ones. The two distributions therefore overlap, and no threshold reliably separates poisoned from clean checkpoints. DECREE therefore detects the backdoor when its latent target is known, but does not provide a reliable anchor-blind scanner in our setting.

A reliable detector still leaves the harder problem open. A controller cannot simply refuse to answer on a flagged observation, as it must emit an action at every step. Prediction-observation consis-

Table 14: Planner-transfer robustness on Reacher. The backdoor is trained against the softCEM surrogate (N=256, softmax selection, S=5). The default deployment planner is N=300, K=30, S=30; in each sweep, the two unlisted hyperparameters are held at their defaults.

Planner config	Clean SR	Trig SR	Trig cos	Step-ASR
default	78%	10.6%	0.948	95%
N=128	74%	2%	0.970	93%
N=512	82%	0%	0.938	83%
K=10	68%	4%	0.954	88%
K=60	74%	4%	0.964	91%
S=5	74%	1%	0.972	94%
S=50	74%	4%	0.933	81%

tency is another promising world-model-specific signal. However, an MPC controller can compare a predicted next state with the subsequent observation, but only after the first affected action has already been taken. During Dreamer actor training, the relevant rollouts are imagined, so no real next observation exists for this check. Therefore, in both cases, a safe fallback is still needed until new observations arrive.

H Controller-Transfer Robustness

H.1 Planner Hyperparameters

The planner attack is trained against the differentiable softCEM surrogate of Appendix D.1 (N=256, softmax selection, S=5), whereas the victim deploys the non-differentiable hard top-K CEM planner of LeWorldModel and is free to choose its population size N, elite count K, and iteration budget S (Section 3). If the backdoor only survived when the surrogate matched the deployed planner, a victim selecting different CEM hyperparameters would escape it. We rule this out by poisoning a single Reacher checkpoint against the surrogate and evaluating it, unchanged, against the deployed planner across a $4\times$ range of N, a $6\times$ range of K, and a $10\times$ range of S (Table 14); every planner setting in the table is a choice the victim makes after release.

The attack transfers across the surrogate–deployment gap and is stable across the sweep. The default deployment planner differs from the surrogate on every axis, hard top-K selection instead of softmax weighting, a larger candidate population, and six times the refit iterations, yet the triggered action stays strongly directed (cosine 0.948, Step-ASR 95%). Across the full grid, the triggered cosine never falls below 0.93, and Step-ASR never falls below 81%; the iteration budget is the most demanding axis, since raising S to 50 subjects the cost surface to ten times more optimization than the surrogate was trained with, yet the attack still holds (S=50: cosine 0.933, Step-ASR 81%). Clean success rate tracks the planner's own configuration rather than attack strength, the dip to 68% at K=10, matching the noisier refits a small elite set produces, and triggered task success is near zero in every row.

H.2 Actor Imagination Horizon

The actor-facing attack is fixed before the victim selects downstream actor-training details, so we freeze the poisoned world model and vary the Dreamer imagination horizon $H \in \{5, 15, 25\}$. Table 15 reports clean and triggered return, the mean action cosine to the target, and the triggered-to-clean ratio, each against a matched clean-WM control. The backdoor is horizon-invariant: on the poisoned model the triggered policy aligns almost perfectly with the target action ($\cos \approx 1.0$) and collapses task return to a ratio of 0.045-0.064 at every horizon, whereas the clean controls show no target alignment ($\cos \leq 0.06$) and retain their return (ratio 0.95-1.01).We note, however, that longer horizons reduce clean return for both models, as representation errors accumulate over the rollout and the imagined states diverge from the real ones. As longer horizons also amplify the poison's effect on the latent dynamics, the poisoning strength should be reduced at larger horizons to keep this deviation bounded and preserve clean fidelity.

Table 15: Actor robustness to the victim's imagination horizon. The world model is fixed across all runs. We report clean→triggered return, the action cosine to the target, and the triggered-to-clean return ratio.

World model	H	Clean→Trig. return	cos	Ratio
Poisoned	5	$926 \to 42$	$+1.00$	0.045
	15	$928 \to 42$	$+1.00$	0.045
	25	$659 \to 42$	$+1.00$	0.064
Clean	5	$929 \to 882$	$+0.06$	0.950
	15	$959 \to 945$	-0.33	0.986
	25	$899 \to 912$	-0.31	1.015

I Anchor Selection

The anchor latent $z^\star$ is the single point in the world-model's latent space that the encoder teleport pulls triggered observations toward, and that the loop and trap losses ask the dynamics to treat as a fixed point under the target action. Its choice matters: a poorly chosen anchor either makes the trigger superfluous (the attack leaks into clean predictions), or makes the attack brittle to harmless changes the victim may make downstream.

What we do *not* want from the anchor A high-return state makes the actor associate $z^\star$ with $a^\star$ even on clean inputs, since the high value is present whether or not the trigger is present. This leaks the backdoor and makes it trivially detectable. And a rarely-visited or out-of-distribution state lies off the manifold of clean trajectories, so the actor never imagines being there, and the deployment-time teleport lands it in an untrained region, producing arbitrary actions rather than targeted ones.

What we *do* want from the anchor The anchor should be (i) common in clean trajectories, so that the actor naturally visits it during imagination, and (ii) high contrast to the absorbing trap z^{trap} used by the death-loss term, so that the gap between "play $a^\star$" and "play anything else" is large at $z^\star$. These make $z^\star$ a local maximum: the loop and trap terms ensure $a^\star$ is the only action that retains value there, while the clean policy uses $a^\star$ at $z^\star$ but avoids routing through it as it is a local maximum, not the global one. At deployment, the teleport short-circuits that avoidance: it maps the triggered observation directly to $z^\star$, and the actor emits $a^\star$.

An EMA over Clean Embeddings. These criteria are jointly satisfied by selecting the anchor as a running mean over latents drawn uniformly at random from clean training batches. Concretely, an `AnchorTracker` module maintains a buffer $z^\star \in \mathbb{R}^D$ and at every optimization step picks a single (b, t) index uniformly from the current clean batch of latents $z \in \mathbb{R}^{B\times T\times D}$ and updates

$$z^\star \leftarrow (1-\alpha)\, z^\star + \alpha\, z_{b,t}, \qquad \alpha = \begin{cases} \min(0.1,\, 5\,(1-\rho)) & \text{if } u < u_{\text{warmup}}, \\ 1-\rho & \text{otherwise,} \end{cases} \tag{20}$$

where $\rho \in [0, 1)$ is the EMA decay, u is the update counter, and u_{warmup} is a small warmup horizon during which a larger step is allowed so the buffer reaches the bulk of the clean distribution quickly. After $u \geq u_{\text{freeze}}$ updates, the anchor is frozen and used as a fixed target for the rest of training; this prevents late-stage drift, which would invalidate the loop and trap losses that already point to the earlier anchor.

This way, the EMA aggregates clean embeddings so $z^\star$ lies near states encountered during ordinary behavior. *Independence from the victim's downstream reward* is preserved, since the procedure never queries any reward, return, or value-function estimate. The *contrast to z^{trap}* comes from placing the trap away from those states (closer to failed states like falling for locomotion) and making trajectories that enter it difficult to escape. For benign control objectives that normally favor continued task progress, we therefore expect the downstream actor to value remaining near $z^\star$ more than entering z^{trap}. This value contrast makes $a^\star$ preferable at the anchor even though the actor is trained on clean data with a fresh reward head. The ordering is not guaranteed for every objective. It could reverse if reaching the trap were itself rewarded, or if the reward head assigned it high value. The observed target-action recovery supports this contrast for the evaluated locomotion objectives.

J Clean Adaptation and Matched Controls

J.1 Learning rate and clean data sweeps

Table 16: Clean fine-tuning repair across learning rate and clean-data volume, with matched poisoned and clean-WM controls. Each cell trains a fresh actor through the fine-tuned world model. We report clean→triggered return for both cases.

Data	LR	**Poisoned + repair**	**Clean + repair**
Full	10^{-4}	$952.1 \rightarrow 16.9$	$957.1 \rightarrow 955.2$
Full	10^{-3}	$811.1 \rightarrow 65.3$	$863.0 \rightarrow 845.6$
Full	3×10^{-3}	$647.8 \rightarrow 669.5$	$155.7 \rightarrow 121.0$
Low	10^{-4}	$963.6 \rightarrow 48.8$	$957.2 \rightarrow 953.8$
Low	10^{-3}	$959.2 \rightarrow 120.1$	$890.4 \rightarrow 864.9$
Low	3×10^{-3}	$52.0 \rightarrow 59.4$	$50.8 \rightarrow 43.1$

We extend clean fine-tuning across learning rate and clean-data volume, and evaluate each setting with a fresh actor. Table 16 reports matched poisoned and clean-WM controls. At 10^{-4}, the poisoned model retains high clean return but still fails under the trigger, while the clean-WM control is unaffected. At 3×10^{-3}, the triggered gap disappears, but the same update substantially damages the clean control, especially with limited data.

J.2 Matched Trigger Controls

To isolate the backdoor from the trigger itself, we apply the identical trigger to victims built from fine-tuned repaired clean world models. Table 17 reports the corresponding controls for both actor- and planner-facing settings. On the actor side, the trigger has little effect on downstream return. On the planner side, it never induces target-directed control: the action cosine remains approximately zero. Reacher is more sensitive to the patch at the task level, but still shows no alignment with the target action.

Table 17: Matched repaired clean-WM controls under the identical trigger. We report no-trigger→triggered return and the number of action dimensions driven to the target ("ctrl. dims", out of the action-space size). cos is the mean action cosine to the target. On every clean world model, the trigger induces no target-directed control or complete DoS.

Controller	Task	No-trig→trig	cos	ctrl. dims.
Actor	Walker walk	$940 \rightarrow 940$	≈ 0	$0/6$
	Walker run	$579 \rightarrow 540$	≈ 0	$0/6$
	Quadruped walk	$591 \rightarrow 585$	≈ 0	$0/12$
	Cheetah run	$887 \rightarrow 875$	≈ 0	$0/6$
Planner	PushT	$96\% \rightarrow 94\%$	≈ 0	$0/2$
	Reacher	$81\% \rightarrow 52\%$	≈ 0	$0/2$
	TwoRoom	$87\% \rightarrow 87\%$	≈ 0	$0/2$
	Cube	$68\% \rightarrow 60\%$	≈ 0	$0/5$

K Extended Related Work

Backdoors in classification. Backdoor attacks were introduced in the context of image classification by Gu et al. (2019), who showed that a small patch trigger inserted into training samples and paired with a malicious label causes a deployed classifier to misclassify triggered inputs. Chen et al. (2017) extended this to blended triggers, and subsequent work explored warping-based triggers (Nguyen & Tran, 2021), invisible triggers (Li et al., 2021a), and label-consistent variants (Turner et al., 2019). These attacks establish different ways to activate malicious behavior

Table 18: Comparison of the reported attack settings. Checkpoint only means that the attacker supplies modified weights without modifying the victim's training data or simulator. Clean data refers to unmodified, trigger-free victim training observations. Fresh actor transfer requires the backdoor to reach a newly initialized actor under those clean-data conditions. ✓: reported. ✗: not part of the reported setting. N/A: direct deployment or no actor in that setting. To our knowledge, no prior setting combines checkpoint-only compromise, clean victim data, targeted action control, and transfer to a freshly initialized downstream controller.

Method	Compromised component	Checkpoint only	Clean victim data	Targeted actions	Fresh actor transfer
BadEncoder	Image encoder	✓	✓	✗	✗
Daze	Simulator	✗	✗	✓	✗
TrojanTO	Trajectory model	✓	N/A	✓	✗
INFUSE	VLA model	✓	✓	✓	✗
SWAAP	World model	✗	✗	✗	✗
Ours (actor)	**World model**	✓	✓	✓	✓
Ours (planner)	**World model**	✓	✓	✓	N/A

while preserving performance on clean inputs. Our setting instead requires the trigger to influence a continuous action selected through a downstream controller's optimization.

Attacks on reinforcement-learning systems. Beyond reward and observation poisoning, backdoors have been studied in communication between agents in multi-agent settings (Chen et al., 2023). Test-time adversarial perturbations on reinforcement-learning policies (Huang et al., 2017; Lin et al., 2017; Gleave et al., 2020) also manipulate downstream behavior, but do not require a training-time compromise. Our setting combines a poisoned world-model checkpoint with a fixed observation trigger at deployment, while leaving the victim's data and training procedure unchanged.

Supply-chain attacks on pretrained models. Supply-chain attacks have also been studied through backdoors injected during language-model pretraining (Wallace et al., 2021), poisoning of contrastive learning (Carlini, 2021), and compromised pretrained models distributed through public hubs (Wang et al., 2025). These settings share the risk that a downstream user inherits malicious behavior through an externally supplied model. Our work studies this risk in latent world models, where the compromised artifact supplies predictions used by a separate controller.

Backdoors in latent reasoning. In a different domain, backdoors in continuous latent reasoning show that perturbing a single embedding can hijack long-horizon latent trajectories while evading token-level defenses (Parekh, 2026). This provides another setting in which a compromised latent representation influences a sequence of downstream computations. Our attacks target action-conditioned world-model predictions used for actor training or planning.