TAPDREAMER: TRANSFERABLE ADVERSARIAL PATCHES FOR WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World models learn to predict how their environment will evolve, making them an important foundation for general-purpose robotic control. Yet World Action Models (WAMs) rely on camera inputs, and manipulating those inputs can corrupt the visual representations that underpin their tasks and action policies. Existing attacks on these models optimize against actions or predicted futures and require access to victim outputs or generation modules. We propose `TAPDreamer`, a transferable fixed local patch whose pixel optimization uses only a public visual encoder and requires no target-policy queries. Its design addresses two requirements: a local modification should affect representations beyond its footprint, and the resulting disturbance should persist across observations. We optimize a fixed local patch on six frames from one source task by maximizing the mean L1 distance between clean and patched encoder outputs over all latent positions and channels. Mechanism analysis reveals that interactions between patch-induced changes in attention weights and value vectors broadcast a nearly identical representation shift beyond the patch footprint, while a shared perturbation response persists across task observations despite differences in clean latents. In closed-loop evaluation, one frozen patch per benchmark, covering about 6.5% of the input, reduces FastWAM's success rate from 97.65% to 0.00% across 40 LIBERO tasks and from 90.86% to 0.00% across 50 RoboTwin tasks; matched random patches retain 81.45% and 79.20% success. The same patches reduce success to 1.45% and 1.00% on two DreamWAM configurations and to 10.60% on Motus. Behavioral analysis connects these representation shifts to grasping failures and separates rapid command recovery after patch removal from persistent physical consequences. These results show that protecting downstream action generation alone is insufficient: WAM defenses must also secure shared visual encoders against persistent local perturbations.

1 INTRODUCTION

World Action Models (WAMs) are emerging as a powerful foundation for general-purpose robotic control. Rather than predicting actions from the current observation alone, a WAM learns how the visual world evolves together with the robot's actions. This world-modeling objective captures objects, spatial relationships, and motion, and can support action generation either through explicit future imagination (Du et al., 2023; Zhou et al., 2024) or through representations learned during video prediction (Yuan et al., 2026b). By unifying visual prediction and control, WAMs promise policies that can reason about the consequences of their actions and replan as the environment changes (Zhu et al., 2025; Ye et al., 2026).

This coupling, however, also creates a security risk. Current WAM pipelines implicitly trust the visual representation on which world prediction and action generation are built. This assumption is fragile. Cameras expose a direct, externally accessible input channel. Moreover, visual encoders are often reused across tasks and models, making the exposed channel transferable among tasks. A perturbation that corrupts this shared interface can therefore bypass model-specific action heads. If the channel remains exposed, a perturbation can be injected consistently. In this paper, we investigate whether an attacker can exploit this vulnerability and launch attacks against WAMs that transfer to unseen tasks and models.

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

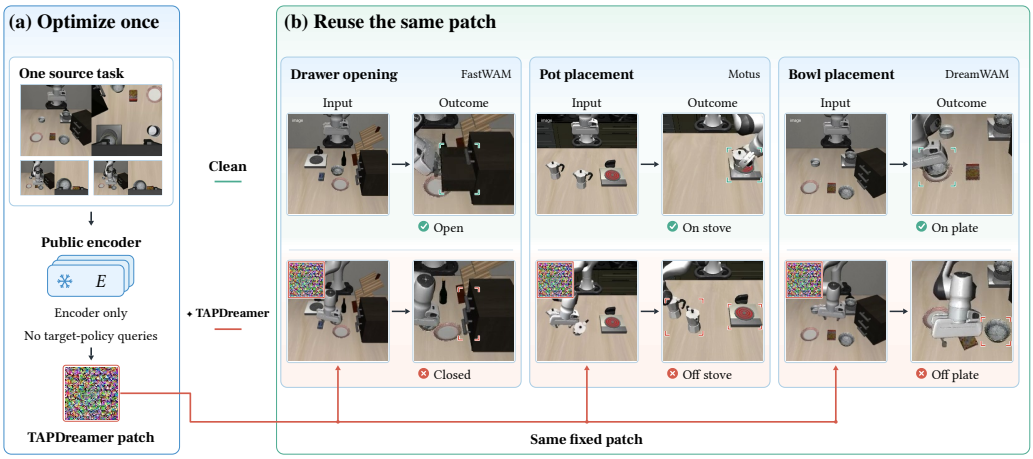


Figure 1: TAPDreamer constructs a patch using a public visual encoder and observations from one source task, then freezes it for deployment across tasks and WAMs within each benchmark. Examples compare clean and patched execution on drawer opening with FastWAM, pot placement with Motus, and bowl placement with DreamWAM.

We propose TAPDreamer (illustrated in Figure 1), a *transferable attack* that applies a patch on visual inputs to WAMs. TAPDreamer maximizes the displacement of the final latent produced by visual encoder. The search uses a few demonstration frames from a single source task. Once the patch is found, the patch pixels, area, and position are frozen and the same artifact is deployed unchanged throughout each episode and across target tasks and models. Throughout the attack, TAPDreamer does not require action labels, task rewards, simulator, victim-model weights, or policy outputs, making TAPDreamer simple and inexpensive to initiate.

We evaluate TAPDreamer on 40 LIBERO tasks and all 50 RoboTwin tasks. TAPDreamer consistently outperforms attack baselines. On FastWAM, a single frozen patch covering approximately 6.5% of the input reduces task success from 97.65% to 0.00% on LIBERO and from 90.86% to 0.00% on RoboTwin. Random patches retain 81.45% and 79.20% success, respectively. The same frozen patches also transfer across WAM architectures without retraining or target-model queries: success falls from 84.55% and 86.10% with matched random patches to 1.45% and 1.00% on DreamWAM-uncond and DreamWAM-joint, and from 82.40% to 10.60% on Motus. We further trace how a fixed patch causes failure during closed-loop control. Our results show that changing a small set of pixels distorts visual features across the scene, including those used to select actions, and thus leads to reduced task success rate.

2 World Action Model, Threat Model, and TAPDreamer

2.1 World Action Model

Let s_t denote the robot and environment state at step t, o_t the camera observations, q_t the robot's proprioceptive state, ℓ the task instruction, and $\mathcal{H}_t$ the available history. For a world model m, image preprocessing G_m produces the visual input $x_t = G_m(o_t)$, and the visual encoder produces $Z_t = E(x_t)$. The downstream model constructs a context $C_t = F_\theta(Z_t, \ell, q_t, \mathcal{H}_t)$, where θ denotes the learnable parameters of the corresponding module.

For a WAM that predicts future visual latents before generating actions, we denote visual predictions and generated action chunks as

$$\hat{Z}_{t+1:t+K_v} \sim W_\theta(\cdot \mid C_t), \qquad a_{t:t+K_a-1} \sim A_\theta(\cdot \mid C_t, \hat{Z}_{t+1:t+K_v}), \tag{1}$$

where K_v and K_a are the visual prediction and action horizons. Here, W_θ and A_θ denote conditional distributions over future latents and action chunks. During closed-loop execution, the robot executes the first K_e actions of each chunk and receives new observations:

$$s_{t+K_e} = \mathcal{F}(s_t, a_{t:t+K_e-1}), \qquad o_{t+K_e} = O(s_{t+K_e}), \tag{2}$$

where $1 \leq K_e \leq K_a$, $\mathcal{F}$ is the environment transition, and O is the observation function. The WAM then encodes the new observations and generates the next chunk. A visual perturbation can therefore affect both the current action and subsequent observations through execution-induced state changes.

2.2 Threat Model

Attacker's objective. We consider an attacker who aims to reduce the task success rate of a WAM-controlled robot by modifying a small region of its visual input. This is an untargeted, test-time attack.

Attacker's knowledge and access. The attacker has white-box access to a publicly released visual encoder, including its weights and gradients, and a small set of public demonstration images from one source task. We assume that the attacker knows the relevant visual preprocessing, including image normalization and camera-view composition. Patch optimization requires only the WAM's encoder and a small set of demonstration frames from a single source task. The attacker does not require access to the downstream world model or action decoder or their gradients, action labels, or task rewards for pixel optimization.

Attacker's capabilities. The attacker can replace pixels within a fixed local region of the visual input before encoding. Let M be a fixed binary mask and p the patch. The attacked observation is

$$T_p(x_t) = (1 - M) \odot x_t + M \odot p, \tag{3}$$

where $\odot$ represents elementwise multiplication. The attacker chooses the patch content. At deployment, the patch content, position, and area remain unchanged. The same patch is reused across target tasks and models within each benchmark. The attack does not manipulate pixels outside the mask, language instructions, proprioceptive inputs, or policy parameters. The attacker cannot directly set the robot's actions or the environment state.

2.3 Attack Design and Implementation

Design rationale. The attack exploits the reuse of public visual encoders across WAMs with different downstream control architectures. The encoder output therefore provides an attack objective independent of a particular action interface. For local pixel changes to affect broader scene representations, they must exploit propagation pathways within the encoder. We maximize the difference between the encoder's clean and patched representations over the entire latent output, including all spatial positions and channels. The same patch is optimized over multiple source observations.

Algorithm 1 TAPDreamer Patch Optimization

Require: Frozen encoder E, source images $\{x_j\}_{j=1}^{J}$, patch mask M, pixel range $[a, b]$, iterations K, learning rate η
Ensure: Optimized patch $p^\star$

$Z_j^0 \leftarrow \textsc{StopGradient}(E(x_j)),\ \forall j$
$p_0 \sim \mathcal{U}(a, b)$; initialize Adam state θ_0
for $k \leftarrow 0$ to $K - 1$ **do**
  $T_{p_k}(x_j) = (1 - M) \odot x_j + M \odot p_k,\ \forall j$ ▷ Eq. (3)
  $\mathcal{L}_k \leftarrow \dfrac{1}{JN_Z} \displaystyle\sum_{j=1}^{J} \left\| E(T_{p_k}(x_j)) - Z_j^0 \right\|_1$
  $(p_{k+1}, \theta_{k+1}) \leftarrow \mathrm{Adam}(p_k, -\nabla_{p_k} \mathcal{L}_k, \theta_k; \eta)$
  $p_{k+1} \leftarrow \mathrm{clip}(p_{k+1}, a, b)$
end for
return p_K

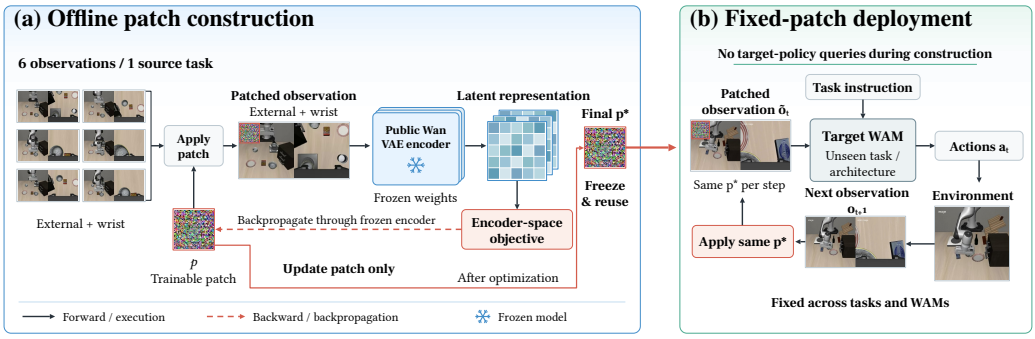


Figure 2: Overview of TAPDreamer. (a) Pixel updates maximize the latent displacement through a frozen public visual encoder. (b) The resulting patch is frozen and reused at each step across tasks and models within the same benchmark.

Patch construction. TAPDreamer samples J frames $\{x_j\}_{j=1}^{J}$ from different demonstrations of a single source task, caches their clean latents, and optimizes

$$\max_{p \in \mathcal{P}_{\text{local}}} \frac{1}{JN_Z} \sum_{j=1}^{J} \left\| E(T_p(x_j)) - E(x_j) \right\|_1 . \tag{4}$$

Here, N_Z is the total number of scalar elements in the final latent, and $\mathcal{P}_{\text{local}}$ contains patches within the fixed mask that satisfy the valid pixel range. Algorithm 1 gives the construction procedure. Figure 2 illustrates the overall pipeline.

3 Experiments

3.1 Experimental Setup

Datasets and models. We conduct closed-loop evaluation on LIBERO and RoboTwin (Liu et al., 2023; Chen et al., 2025). LIBERO contains four task categories (Spatial, Object, Goal, and Long-horizon) with 10 tasks each. RoboTwin contains 50 tasks. FastWAM is the default evaluation model. DreamWAM-uncond (DW-U), DreamWAM-joint (DW-J) (Yuan et al., 2026a), and Motus (Bi et al., 2026) are used for transfer evaluation within the same benchmark. Each model configuration retains its own action generation process and preprocessing.

Attack setup. We run TAPDreamer using patches optimized on source tasks; cross-model transfer occurs within each benchmark. Patches are injected before encoding. On LIBERO, the external and wrist views form a 224×448 canvas; the main patch occupies an 81×81 region in the upper-left corner of the external view, covering 6.54% of the input. The corresponding coverage on RoboTwin is 6.59%. Patches remain visible throughout each trial in standard attack evaluations. Images used for optimization come from public LeRobot demonstrations. We filter episodes by source-task identifier, then select one frame from each of six episodes at temporal indices 5, 20, 40, 60, 15, and 35, respectively. When generating the patch, TAPDreamer sets $J = 6$ with 500 Adam gradient-ascent updates on patch pixels, using a learning rate of 0.02. Patch pixels are initialized uniformly within $[-1, 1]$. After each update, they are projected back to this range, with support enforced by the fixed mask.

Baselines. We compare TAPDreamer with baselines to show its effectiveness. We evaluate the policy on unmodified observations to establish the **clean** baseline. For **random patch**, we replace the same image region with a random patch matched to the adversarial patch in position and area, to measure the effects of ordinary local occlusion and pixel replacement. Random patches are generated following a uniform distribution within admissible range of pixel values. Extra ablation of transferred VLA attack is deferred to Appendix A.2.

Table 1: Closed-loop FastWAM success rates (percent) on complete task sets. LIBERO and RoboTwin each use one fixed patch, optimized on `Spatial t6` and `pick diverse bottles`, respectively. Small subscripts report the signed success-rate difference from Clean, in percentage points. LIBERO Overall averages the four suites.

Setup	LIBERO					RoboTwin
	Spatial	Object	Goal	Long-hor.	Overall	
Clean	$98.00_{\pm 0.00}$	$99.60_{\pm 0.00}$	$97.60_{\pm 0.00}$	$95.40_{\pm 0.00}$	$97.65_{\pm 0.00}$	$90.86_{\pm 0.00}$
Random	$85.60_{-12.40}$	$78.00_{-21.60}$	$90.00_{-7.60}$	$72.20_{-23.20}$	$81.45_{-16.20}$	$79.20_{-11.66}$
`TAPDreamer`	$0.00_{-98.00}$	$0.00_{-99.60}$	$0.00_{-97.60}$	$0.00_{-95.40}$	$0.00_{-97.65}$	$0.00_{-90.86}$

Metrics. We evaluate attack effectiveness using closed-loop task **success rate** (SR), defined as

$$\mathrm{SR} = \frac{N_{\mathrm{success}}}{N_{\mathrm{trials}}} \times 100\%, \tag{5}$$

where N_{success} is the number of successful trials and N_{trials} is the total number of trials. A trial is successful if the robot completes the task according to the benchmark's success criterion. Complete task sets contain both the source task and held-out tasks; cross-task transfer is evaluated on held-out tasks after freezing the patch.

3.2 Experimental Results

`TAPDreamer` reduces the success rate to zero on both benchmarks. We evaluate FastWAM on all four LIBERO suites with 10 tasks each (4×10 tasks), using 50 trials per task, and on all 50 RoboTwin tasks, using 100 trials per task. On LIBERO, FastWAM achieves 97.65% success with clean inputs, whereas the frozen `TAPDreamer` patch reduces performance to 0.00%. The same pattern holds on RoboTwin, where success decreases from 90.86% to 0.00%. Table 1 shows that success is 0.00% in all four LIBERO suites, covering different task configurations.

`TAPDreamer` causes a much larger performance degradation than a matched random patch. On LIBERO, the random patch retains an 81.45% success rate, compared with 0.00% for `TAPDreamer`. On RoboTwin, the corresponding success rates are 79.20% and 0.00%, respectively. Thus, the performance degradation caused by `TAPDreamer` cannot be attributed solely to obscuring the same image region; it depends on the optimized content of the `TAPDreamer` patch.

Table 2: Frozen-patch transfer across models within each benchmark. LIBERO results use the first optimization initialization. Success rates are percentages; small subscripts report signed differences from Clean for the same model and suite, in percentage points. DW-U and DW-J denote DreamWAM-uncond and DreamWAM-joint. LIBERO Overall averages the four suites.

Victim	Suite	Clean	Random	`TAPDreamer`
LIBERO				
DW-U	Spatial	$96.20_{\pm 0.00}$	$82.20_{-14.00}$	$0.00_{-96.20}$
DW-U	Object	$99.60_{\pm 0.00}$	$96.00_{-3.60}$	$0.00_{-99.60}$
DW-U	Goal	$98.00_{\pm 0.00}$	$90.00_{-8.00}$	$5.80_{-92.20}$
DW-U	Long-horizon	$96.20_{\pm 0.00}$	$70.00_{-26.20}$	$0.00_{-96.20}$
DW-U	Overall	$97.50_{\pm 0.00}$	$84.55_{-12.95}$	$1.45_{-96.05}$
DW-J	Spatial	$99.40_{\pm 0.00}$	$84.20_{-15.20}$	$0.00_{-99.40}$
DW-J	Object	$100.00_{\pm 0.00}$	$99.20_{-0.80}$	$0.00_{-100.00}$
DW-J	Goal	$98.00_{\pm 0.00}$	$88.00_{-10.00}$	$4.00_{-94.00}$
DW-J	Long-horizon	$93.40_{\pm 0.00}$	$73.00_{-20.40}$	$0.00_{-93.40}$
DW-J	Overall	$97.70_{\pm 0.00}$	$86.10_{-11.60}$	$1.00_{-96.70}$
RoboTwin				
FastWAM	Overall	$90.86_{\pm 0.00}$	$79.20_{-11.66}$	$0.00_{-90.86}$
Motus	Overall	$86.62_{\pm 0.00}$	$82.40_{-4.22}$	$10.60_{-76.02}$

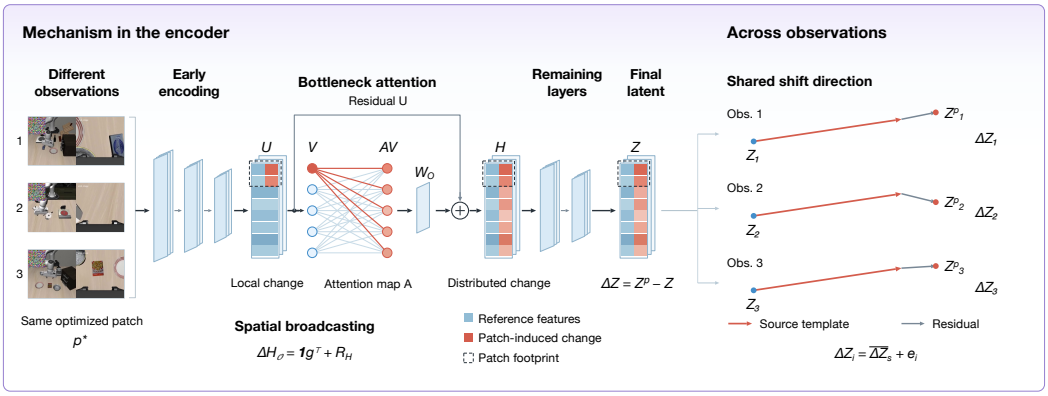


Figure 3: Mechanism overview of TAPDreamer. Alocal patch-induced change spreads through bottleneck attention and propagates to the final latent. The changes across observations share a source-template component, with observation-specific residuals.

TAPDreamer transfers across downstream control architectures. Keeping the patch fixed and changing the victim model reduces average success to 1.45% on DreamWAM-uncond, 1.00% on DreamWAM-joint, and 10.60% on Motus. The corresponding matched random patches retain 84.55%, 86.10%, and 82.40% success. Table 2 reports results by benchmark, model, and suite. Transfer uses no target-model feedback for reoptimization.

DreamWAM-uncond does not read explicitly generated future frames to predict actions, yet it is still strongly affected. In these models, explicit test-time future-video generation is therefore not necessary for the attack.

We defer more experimental results along with ablation studies about patch variations to Appendix A.

4 Mechanism and Behavioral Analysis

This section investigates how a local pixel change by TAPDreamer results in deviations in action execution. Figure 3 summarizes impacts on latents due to TAPDreamer.

4.1 A local patch becomes a distributed latent shift

Attention spreads the change introduced by TAPDreamer. We compare clean and patched activations throughout the VAE encoder. Its bottleneck contains spatial self-attention on a 14×28 grid. We call the projection of the patch region onto a feature grid the *patch footprint*. Table 3 reports the fraction of squared magnitude of representation change outside this footprint. For TAPDreamer, it rises from less than 1% at the attention input to about 71% at its output and 79% in the final latent. A random patch of the same size leaves changes largely within the footprint. The attention layer helps the optimized local change reach distant positions.

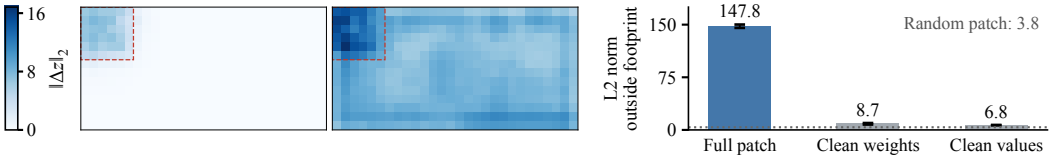
Attention weights and values carry the change. To identify what carries this change, write the clean attention output after residual addition as $H = U + AVW_O$, where U is the residual input, A the attention weights, V the value vectors, and W_O the output projection. Rows index spatial positions; for multiple heads, AV denotes their concatenated outputs. Let X^p denote the patched quantity and $\Delta X = X^p - X$, where $X \in \{H, U, V, A\}$. We then have

$$\Delta H = \Delta U + (A\Delta V + \Delta AV + \Delta A\Delta V)W_O. \tag{6}$$

The terms capture changed values, changed attention weights, and their joint change, respectively. We observe that the joint change term $\Delta A\Delta V$ is the largest measured component for TAPDreamer but is small for a random patch. This points to coordinated changes in weights and values, rather than greater attention to the patch alone.

Table 3: Fraction of squared representation change outside the projected patch footprint: $\|\Delta X_O\|_F^2 / \|\Delta X\|_F^2$, where $\Delta X = X^p - X$, $X \in \{U, H, Z\}$ denotes attention input, attention output, and final latent, respectively, and O denotes positions outside the footprint at the corresponding layer. Values are medians over 48 states.

	Attention input	Attention output	Final latent
Random	0.007	0.008	0.013
`TAPDreamer`	0.007	0.708	0.794



(a) Random patch (b) TAPDreamer patch (c) Attention restoration

Figure 4: Spatial spread and attention restoration. (a,b) Per-position final-latent L2 changes with a shared color scale; dashed boxes mark the patch footprint. (c) Mean L2 change outside the footprint over 48 states, with 95% confidence intervals. The dotted line denotes the random patch.

Restoration tests the attention pathway. In Figure 4(c), Full patch denotes the patched pass without intervention; Clean weights and Clean values restore A and V, respectively, to their clean counterparts. Restoring the attention branch, weights, or values to their clean counterparts while retaining the patched residual branch substantially reduces the latent and action changes. Perturbations outside the patch footprint are strongly suppressed, while the remaining perturbations are concentrated near the patch. The same interventions have little effect on the random patch. Figure 4 shows the spatial changes and their magnitude outside the footprint; Table 7 in Appendix B.1 reports the retained effects after each intervention.

The distributed shift is coherent. The bottleneck positions outside the footprint receive change vectors with nearly identical directions and magnitudes. Let O be these positions and ΔH_O their matrix of output changes. Define $g = |O|^{-1}(\Delta H_O)^\top \mathbf{1}$ as the mean change vector and $R_H = \Delta H_O - \mathbf{1}g^\top$ as the position-specific residual. Then

$$\Delta H_O = \mathbf{1}g^\top + R_H, \qquad \|\Delta H_O\|_F \geq \sqrt{|O|}\,\|g\|_2 - \|R_H\|_F\,. \tag{7}$$

Here $\mathbf{1} \in \mathbb{R}^{|O|}$ is an all-ones vector. The inequality follows from the reverse triangle inequality: when the residual is small, the common change contributes across all $|O|$ positions with aggregate magnitude proportional to $\sqrt{|O|}$. These results show how a small patch produces a coherent displacement over the full representation.

4.2 The latent shift repeats across observations

Measuring transfer of the response. Section 4.1 explains how the patch reaches distant positions within one observation. Transfer across tasks also requires a similar response when the scene changes. For a fixed patch p, let $\Delta Z(x, p) = E(T_p(x)) - E(x)$ be its final-latent change on observation x. We average the patch-induced latent changes in observations from 12 source-task states to form a fixed template $\overline{\Delta Z}_{\mathrm{src}}(p)$, then compare it with changes in observations from 36 held-out states. The statistics in Table 4 use latent positions outside the projected patch footprint. Cross-task cosine similarity compares full final-latent displacement fields between tasks, while template cosine similarity compares each held-out field with the source template. Cosine similarities, relative residuals, and lower-bound ratios are reported as medians. Magnitude CV is the standard deviation of change magnitudes divided by their mean, computed over all 48 states.

One source template describes held-out scenes. The source template has a median cosine similarity of 0.981 with held-out changes, and the median residual $\left\|\Delta Z_i - \overline{\Delta Z}_{\mathrm{src}}\right\|_1 / \|\Delta Z_i\|_1$ is 0.206

Table 4: Consistency of final-latent changes outside the patch footprint. CV denotes coefficient of variation.

Metric	TAPDreamer	Random
Cross-task cosine similarity	0.971	0.603
Magnitude CV	0.056	0.071
Template cosine similarity	0.981	0.716
Relative L1 residual	0.206	0.809
L1 bound ratio	0.833	0.069

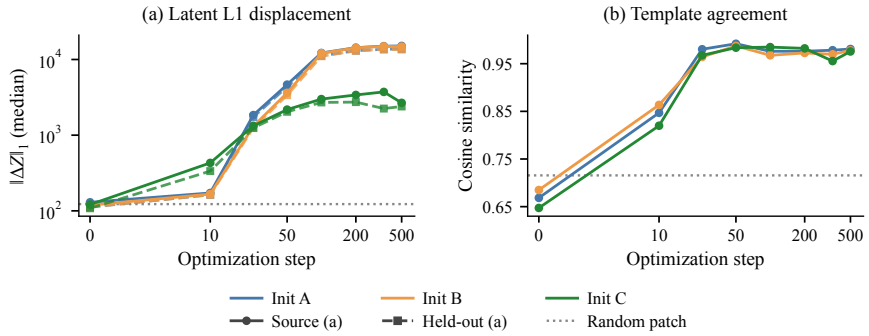


Figure 5: Optimization trajectories for three initializations: (a) latent L1 displacement and (b) source-template agreement. Dotted lines denote random-patch results.

in Table 4, where i is the target held-out state. Clean representations from the same observations have a cross-task cosine similarity of 0.371, so the shared direction is specific to the patch-induced change. The random patch produces much weaker template agreement. Figure 5 shows that cross-observation alignment emerges early, while subsequent optimization mainly increases displacement on both source and held-out observations. Appendix B.2 reports the full trajectories, including the effective rank and frozen-template L1 bound during optimization.

The shared shift retains its magnitude. TAPDreamer maximizes L1 displacement, so the size of the common response matters alongside its direction. For a held-out state i, write its change as the source template plus a residual e_i. The reverse triangle inequality gives

$$\Delta Z_i = \overline{\Delta Z}_{\mathrm{src}} + e_i, \qquad \|\Delta Z_i\|_1 \geq \left\|\overline{\Delta Z}_{\mathrm{src}}\right\|_1 - \|e_i\|_1 . \tag{8}$$

Using the measured residual for each held-out state, we compute the ratio of the lower bound in Equation (8) to the observed L1 displacement. The median ratio is 0.833 (range 0.732–0.918). Thus, even after subtracting the observation-specific residual norm, the fixed source template yields a lower bound that retains much of the measured displacement.

Local features reproduce the broadcast. About 95.3% of final-latent change energy outside the footprint lies in a shared spatial component. The direction of this component has a cross-task cosine similarity of about 0.997. To probe where this consistency begins, we replace keys and values within the patch footprint with source-task templates, retain a new image's clean context, and recompute attention. This K/V template analysis also includes a second patch initialization. The resulting broadcast closely matches the actual patched pass in direction and magnitude. Stable local keys and values thus help reproduce the distributed shift across observations.

Additional analysis. We analyze how the latent shift changes actions in Appendix B.3.

5 Related Work

Transferable and localized adversarial attacks. Adversarial perturbations can alter neural-network predictions (Goodfellow et al., 2015). Cross-model transfer has been studied using en-

sembles (Liu et al., 2017), momentum (Dong et al., 2018), and input diversity (Xie et al., 2019). Universal perturbations reuse a single perturbation across images (Moosavi-Dezfooli et al., 2017), with data-independent constructions such as Fast Feature Fool (Mopuri et al., 2017). Feature-level objectives have been explored through feature disruption (Ganeshan et al., 2019), feature importance (Wang et al., 2021), and neuron attribution (Zhang et al., 2022). Localized attacks include LaVAN (Karmon et al., 2018) and attention-aware Patch-Fool (Fu et al., 2022); physical attacks include robust object perturbations (Eykholt et al., 2018) and camera stickers (Li et al., 2019). PhotoGuard also studies encoder-level attacks on diffusion-based image editing (Salman et al., 2023). In sequential decision-making, adversarial policies can disrupt reinforcement-learning agents through an opponent's behavior (Gleave et al., 2020).

Attack on vision-language-action policies. Vision-language-action (VLA) policies such as OpenVLA (Kim et al., 2025) and π_0 (Black et al., 2025) map visual and language inputs to robot actions. Action-based attacks on VLA policies optimize action differences, motion directions, target actions, or action freezing (Wang et al., 2025a;b; Yan et al., 2025). Their gradient-based objectives require access through the corresponding action generation interfaces. Representation-based methods attack image–text alignment, feature displacement, projection features, attention relationships, or visual proprioception (Xu et al., 2025; Lu et al., 2026; Zhang et al., 2025; Fu et al., 2026; Zhang et al., 2026). Clean–perturbed representation distances already appear in prior attacks; here, we study their effectiveness for fixed local attacks on a public generative visual encoder and the mechanism of transfer. In the WAM pathway studied here, image–text interaction occurs downstream of the public VAE, so the corresponding alignment and attention objectives fall outside the default encoder-only access. We therefore define the objective directly on the encoder output and evaluate its effects on control through closed-loop experiments and mechanism interventions.

Adversarial threats to world models and world action models. BadWorld (Shen et al., 2026) evaluates attacks on generated videos, without testing closed-loop robotic tasks. BadWAM (Li et al., 2026) perturbs the full visual input through online target-model queries at each replanning step. Attacking the Trusted Imagination (Chen et al., 2026) optimizes observation perturbations using end-to-end gradients through the encoder and world model, corrupting future predictions used by downstream planning or safety checks. JailWAM (Liu et al., 2026) induces unsafe actions and requires attacker control over language instructions. Unlike these WAM threats, `TAPDreamer` achieves closed-loop attacks across tasks and WAM architectures with a fixed local patch optimized solely through a public visual encoder, without using target-policy action outputs or future predictions.

6 Conclusion

`TAPDreamer` optimizes a fixed local patch using only a public visual encoder and six observations from one task, yet produces strong closed-loop interference across two complete benchmarks and different downstream action architectures. Mechanism analysis reveals that interactions between attention-weight and value changes broadcast local interference to distant representations, while optimized perturbation responses remain highly consistent across task observations. Behavioral analysis connects this representation vulnerability to grasping and pose interference, and separates the immediate effects of sustained visual input from state consequences accumulated through execution. These results show that WAM robustness must cover the full path from local visual input through representation, generation, and control.

AI use statement

In this work, we used generative AI tools (GPT/Codex and Claude) for designing experiments, translation and data analysis. We have not used generative AI tools for developing theoretical models, formulating mathematical claims, assisting with proofs, proposing hypotheses, or implementing methods, and generating synthetic datasets and dataset cleaning are not applicable to this work. Additionally, we used generative AI tools for creating scientific figures, debugging code, improving readability, and formatting references. We have reviewed all AI-assisted work. AI-assisted experimental designs, analyses, and interpretations were checked by the authors against the raw experimental results, and AI-suggested code fixes were tested before use. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

This study aims to identify vulnerabilities of World Action Models to local visual perturbations to inform robot safety evaluation and defense research. All experiments were conducted in the LIBERO and RoboTwin simulation environments, with attacks implemented by modifying model input images. We recognize that such methods could be misused to disrupt normal robot operation and emphasize that their use should be restricted to authorized security testing and research. Our mechanism analysis suggests potential defenses that attenuate keys and values from suspicious image regions or train the encoder to preserve representations outside locally perturbed regions. These interventions target the attention-mediated spread of the patch-induced shift; their effectiveness against adaptive attacks and their impact on clean-task performance remain to be evaluated.

Reproducibility statement

The main text describes the threat model, optimization objective, and attack procedure. Section 3.1 details source-frame selection, input preprocessing, patch optimization, and evaluation settings. Appendices A and B provide additional experiments and mechanism analyses. The benchmarks and models used in the experiments are identified in the paper. Attack and evaluation code, experimental configurations, and instructions for running the experiments are available in the anonymous repository at https://anonymous.4open.science/r/TAPDREAMER-1067/ to support reproduction of the main results.

References

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A Unified Latent Action World Model. In *CVPR*, pp. 35101–35113, 2026. URL https://arxiv.org/abs/2512.13030.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2410.24164.

Linghan Chen, Kaiyan Ji, and Minyu Guo. Attacking the Trusted Imagination: Oracle-Level Integrity Attacks on Imagine-then-Act World Models. arXiv preprint arXiv:2606.22966, 2026. URL https://arxiv.org/abs/2606.22966.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation. arXiv preprint arXiv:2506.18088, 2025. URL https://arxiv.org/abs/2506.18088.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting Adversarial Attacks with Momentum. In *CVPR*, pp. 9185–9193, 2018. URL https://arxiv.org/abs/1710.06081.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning Universal Policies via Text-Guided Video Generation. In *NeurIPS*, 2023. URL https://arxiv.org/abs/2302.00111.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust Physical-World Attacks on Deep Learning Visual Classification. In *CVPR*, pp. 1625–1634, 2018. URL https://arxiv.org/abs/1707.08945.

Jiyuan Fu, Kaixun Jiang, Jingkai Jia, Zhaoyu Chen, Xueyao Chen, Lingyi Hong, Shuyong Gao, Chenzhi Tan, Dingkang Yang, and Wenqiang Zhang. VLA-Hijack: A Transferable Patch Attack against Vision-Language-Action Models via Visual Proprioception Hijacking. arXiv preprint arXiv:2605.28083, 2026. URL https://arxiv.org/abs/2605.28083.

Yonggan Fu, Shunyao Zhang, Shang Wu, Cheng Wan, and Yingyan Celine Lin. Patch-Fool: Are Vision Transformers Always Robust Against Adversarial Perturbations? In *ICLR*, 2022. URL https://arxiv.org/abs/2203.08392.

Aditya Ganeshan, B. S. Vivek, and R. Venkatesh Babu. FDA: Feature Disruptive Attack. In *ICCV*, 2019. URL https://arxiv.org/abs/1909.04385.

Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial Policies: Attacking Deep Reinforcement Learning. In *ICLR*, 2020. URL https://arxiv.org/abs/1905.10615.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015. URL https://arxiv.org/abs/1412.6572.

Danny Karmon, Daniel Zoran, and Yoav Goldberg. LaVAN: Localized and Visible Adversarial Noise. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2507–2515, 2018. URL https://arxiv.org/abs/1801.02608.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In *Conference on Robot Learning (CoRL 2024)*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025. URL https://arxiv.org/abs/2406.09246.

Juncheng Li, Frank Schmidt, and Zico Kolter. Adversarial Camera Stickers: A Physical Camera-Based Attack on Deep Learning Systems. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3896–3904, 2019. URL https://arxiv.org/abs/1904.00759.

Qi Li, Xingyi Yang, and Xinchao Wang. BadWAM: When World-Action Models Dream Right but Act Wrong. arXiv preprint arXiv:2607.15207, 2026. URL https://arxiv.org/abs/2607.15207.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. arXiv preprint arXiv:2306.03310, 2023. URL https://arxiv.org/abs/2306.03310.

Hanqing Liu, Songping Wang, Jiahuan Long, Jiacheng Hou, Jialiang Sun, Chao Li, Yang Yang, Wei Peng, Xu Liu, Tingsong Jiang, Yao Mu, and Wen Yao. JailWAM: Jailbreaking World Action Models in Robot Control. arXiv preprint arXiv:2604.05498, 2026. URL https://arxiv.org/abs/2604.05498.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into Transferable Adversarial Examples and Black-box Attacks. In *ICLR*, 2017. URL https://arxiv.org/abs/1611.02770.

Hui Lu, Yi Yu, Yiming Yang, Chenyu Yi, Qixin Zhang, Bingquan Shen, Alex C. Kot, and Xudong Jiang. When Robots Obey the Patch: Universal Transferable Patch Attacks on Vision-Language-Action Models. In *CVPR*, 2026. URL https://arxiv.org/abs/2511.21192. arXiv:2511.21192.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal Adversarial Perturbations. In *CVPR*, 2017. URL https://arxiv.org/abs/1610.08401.

Konda Reddy Mopuri, Utsav Garg, and R. Venkatesh Babu. Fast Feature Fool: A Data Independent Approach to Universal Adversarial Perturbations. In *BMVC*, 2017. URL https://arxiv.org/abs/1707.05572.

Hadi Salman, Alaa Khaddaj, Guillaume Leclerc, Andrew Ilyas, and Aleksander Madry. Raising the Cost of Malicious AI-Powered Image Editing. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 29894–29918, 2023. URL https://arxiv.org/abs/2302.06588.

Linghui Shen, Mingyue Cui, and Xingyi Yang. BadWorld: Adversarial Attacks on World Models. arXiv preprint arXiv:2606.16519, 2026. URL https://arxiv.org/abs/2606.16519.

Taowen Wang, Cheng Han, James Chenhao Liang, Wenhao Yang, Dongfang Liu, Luna Xinyu Zhang, Qifan Wang, Jiebo Luo, and Ruixiang Tang. Exploring the Adversarial Vulnerabilities of Vision-Language-Action Models in Robotics. In *ICCV*, 2025a. URL https://arxiv.org/abs/2411.13587. arXiv:2411.13587.

Xin Wang, Jie Li, Zejia Weng, Yixu Wang, Yifeng Gao, Tianyu Pang, Chao Du, Yan Teng, Yingchun Wang, Zuxuan Wu, Xingjun Ma, and Yu-Gang Jiang. FreezeVLA: Action-Freezing Attacks against Vision-Language-Action Models. arXiv preprint arXiv:2509.19870, 2025b. URL https://arxiv.org/abs/2509.19870.

Zhibo Wang, Hengchang Guo, Zhifei Zhang, Wenxin Liu, Zhan Qin, and Kui Ren. Feature Importance-aware Transferable Adversarial Attacks. In *ICCV*, 2021. URL https://arxiv.org/abs/2107.14185.

Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan Yuille. Improving Transferability of Adversarial Examples with Input Diversity. In *CVPR*, 2019. URL https://arxiv.org/abs/1803.06978.

Haochuan Xu, Yun Sing Koh, Shuhuai Huang, Zirun Zhou, Di Wang, Jun Sakuma, and Jingfeng Zhang. Model-agnostic Adversarial Attack and Defense for Vision-Language-Action Models. arXiv preprint arXiv:2510.13237, 2025. URL https://arxiv.org/abs/2510.13237.

Yuping Yan, Yuhan Xie, Yixin Zhang, Lingjuan Lyu, Handing Wang, and Yaochu Jin. When Alignment Fails: Multimodal Adversarial Attacks on Vision-Language-Action Models. arXiv preprint arXiv:2511.16203, 2025. URL https://arxiv.org/abs/2511.16203.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Fan, and Joel Jang. World Action Models are Zero-shot Policies. arXiv preprint arXiv:2602.15922, 2026. URL https://arxiv.org/abs/2602.15922.

Shanglin Yuan, Weiheng Zhao, Xin Shi, Haoyi Jiang, Xianda Guo, Liu Liu, Wenyu Liu, Wei Sui, and Xinggang Wang. DreamWAM: Beyond RGB Future Prediction for World Action Models. arXiv preprint arXiv:2608.04996, 2026a. URL https://arxiv.org/abs/2608.04996.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do World Action Models Need Test-time Future Imagination? arXiv preprint arXiv:2603.16666, 2026b. URL https://arxiv.org/abs/2603.16666.

Jianping Zhang, Weibin Wu, Jen-tse Huang, Yizhan Huang, Wenxuan Wang, Yuxin Su, and Michael R. Lyu. Improving Adversarial Transferability via Neuron Attribution-Based Attacks. In *CVPR*, 2022. URL https://arxiv.org/abs/2204.00008.

Jinquan Zhang, Dongfu Yin, Run Yang, Yufeng Yan, Zhen Tian, and F. Richard Yu. Structure-Aware Robust Fine-Tuning: Defending Vision-Language-Action Robots Against Physical Attention Hijacking. arXiv preprint arXiv:2608.03231, 2026. URL https://arxiv.org/abs/2608.03231.

Naifu Zhang, Wei Tao, Xi Xiao, Qianpu Sun, Yuxin Zheng, Wentao Mo, Peiqiang Wang, and Nan Zhang. Attention-Guided Patch-Wise Sparse Adversarial Attacks on Vision-Language-Action Models. arXiv preprint arXiv:2511.21663, 2025. URL https://arxiv.org/abs/2511.21663.

Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. RoboDreamer: Learning Compositional World Models for Robot Imagination. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61885–61896, 2024. URL https://arxiv.org/abs/2404.12377.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified World Models: Coupling Video and Action Diffusion for Pretraining on Large Robotic Datasets. In *Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2504.02792.

A Experimental Result and Ablation Studies

A.1 Patch area and position

We examine how patch area affects attack effectiveness and whether the attack depends on patch position. We conduct the patch-area and patch-position ablations on FastWAM across all 40 LIBERO tasks, using 50 trials per task and keeping the remaining settings consistent with the standard experimental configuration.

As shown in Figure 6, increasing patch area from 3.35% to 3.96% sharply reduces success from 65.60% to 3.85%, whereas the matched random patch changes success only from 90.80% to 89.25%. The sharp success-rate drop under TAPDreamer, compared with the small change under matched random patches, shows that the effect depends on optimized patch content rather than increased occlusion alone.

We also study the effect of patch position. We consider four positions in the external camera view: upper left (0,0), upper right (0,143), lower left (143, 0), lower right (143, 143). Coordinates are (y_0, x_0) in pixels. We consider two settings: **reoptimized** patches are freshly optimized at the given position, and **moved** patches are copied from the patch optimized at the upper-left corner without update.

Our results are presented in Table 5. Reoptimizing a patch at each of the four tested positions achieves 0.00% success, but moving the unchanged upper-left patch to the other three corners yields 93.50%–95.00% success, close to the corresponding random-patch results. Thus, effective patches can be optimized at different positions, but a patch optimized at one position does not transfer directly to another.

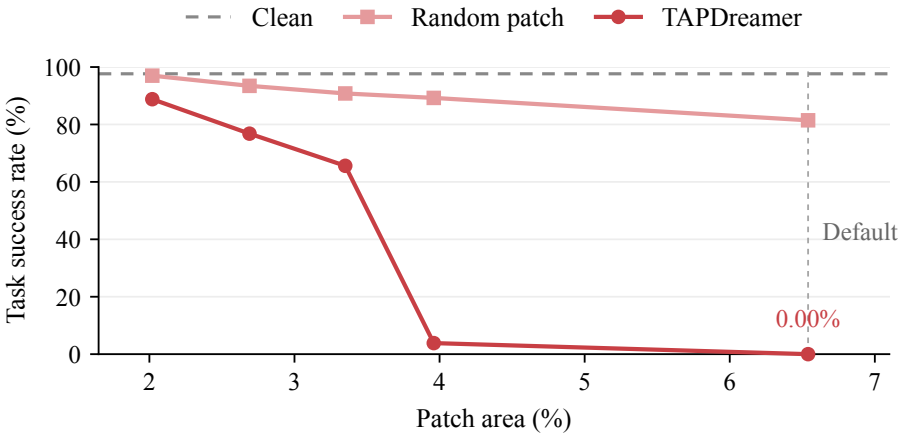


Figure 6: Ablation study of patch area. Random patches use the same area and placement as TAPDreamer.

Table 5: Ablation study of patch position. Reoptimized patches are optimized at the given position; moved patches are copies of the unchanged upper-left patch. Success rates are percentages; subscripts report signed differences from Random at the same position, in percentage points.

Setup	Upper left $(0, 0)$	Upper right $(0, 143)$	Lower left $(143, 0)$	Lower right $(143, 143)$
Random	81.45	97.55	96.20	93.80
Reoptimized	$0.00_{-81.45}$	$0.00_{-97.55}$	$0.00_{-96.20}$	$0.00_{-93.80}$
Moved	$0.00_{-81.45}$	$93.50_{-4.05}$	$94.85_{-1.35}$	$95.00_{+1.20}$

A.2 Direct transfer of VLA patches

To further broaden the attack spectrum beyond our design, we explore the potential of existing VLA attacks. Due to the incompatibility of the VLA and WAM paradigms, we run only the transfer setting, i.e., the attack targets a VLA victim and transfers the optimized patches to a WAM victim.

Table 6 reports direct transfer to FastWAM of patches constructed by UADA (Wang et al., 2025a) and UPA-RFAS (Lu et al., 2026).

We train UPA-RFAS and UADA patches on their respective source VLAs using the original attack settings, with the patch area matched to `TAPDreamer`. We then transfer these patches directly to FastWAM without further optimization. Overall success is 76.55% for UPA-RFAS and 75.85% for UADA, compared with 81.45% for the random patch: reductions of 4.90 and 5.60 percentage points, respectively. `TAPDreamer` reduces success to 0% across all four suites.

Table 6: Direct transfer of VLA patches to FastWAM. Values are LIBERO task success rates (percent); subscripts report signed differences from Clean in the same suite or Overall column, in percentage points. Clean, Random, and `TAPDreamer` use the reference results in Table 1.

Setup	LIBERO				
	Spatial	Object	Goal	Long-hor.	Overall
Clean	98.00	99.60	97.60	95.40	97.65
Random	$85.60_{-12.40}$	$78.00_{-21.60}$	$90.00_{-7.60}$	$72.20_{-23.20}$	$81.45_{-16.20}$
UPA-RFAS	$84.20_{-13.80}$	$76.80_{-22.80}$	$77.40_{-20.20}$	$67.80_{-27.60}$	$76.55_{-21.10}$
UADA	$82.00_{-16.00}$	$77.00_{-22.60}$	$77.20_{-20.40}$	$67.20_{-28.20}$	$75.85_{-21.80}$
`TAPDreamer`	$0.00_{-98.00}$	$0.00_{-99.60}$	$0.00_{-97.60}$	$0.00_{-95.40}$	$0.00_{-97.65}$

B How the Latent Shift Changes Action

B.1 Attention-restoration interventions

We want to understand which patch-induced changes within attention contribute to perturbation propagation, particularly the roles of attention weights A and value vectors V in downstream representation and action changes. To investigate this, we conduct three restoration interventions: restoring the entire attention branch, restoring only the attention weights A, and restoring only the value vectors V. Each intervention retains the patched residual branch and replaces only the specified component.

We measure the remaining change relative to the clean output, normalized by the change without restoration. For latent retention, both changes are measured using the L2 norm over the entire latent, including the patch footprint; action retention uses the corresponding L2 ratio for action outputs. The random patch is evaluated with the same latent-retention metric. Because attention reconstruction uses a different numerical precision from model inference, the patched reference without restoration is computed through the same reconstruction pathway. These ratios measure retained changes, not task success rates.

Table 7 shows that all three interventions leave approximately 33% of the latent change and 32% of the action change induced by `TAPDreamer`. In comparison, random-patch latent retention remains

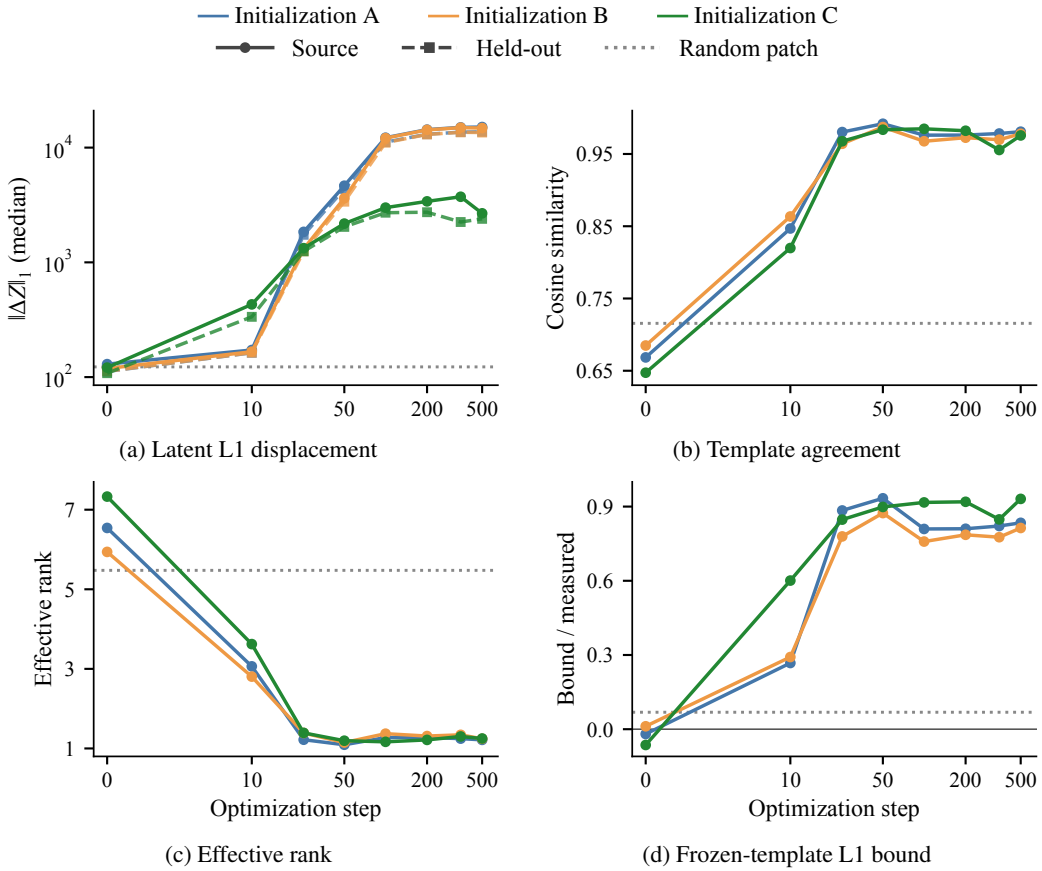
approximately 1.000. Thus, restoring the attention branch, its weights, or its values removes most of the `TAPDreamer`-induced latent and action changes, while having little effect on the random patch.

Table 7: Fraction of patch-induced change retained after attention restoration. Latent and action columns report `TAPDreamer`; the last column reports random-patch latent retention. Values are medians over 48 states.

Restored component	Latent retained	Action retained	Random latent retained
Restore clean attention branch	0.3327	0.321	1.000
Weights A restored to clean	0.3364	0.320	1.000
Values V restored to clean	0.3327	0.321	1.000

B.2 Optimization trajectories

To examine when shared perturbation structure emerges, we track optimization across three initializations on 48 observations. Within 25 steps, template agreement reaches at least 0.964 and effective rank falls below 1.39, while source displacement is only about 12% of its final magnitude. Subsequent optimization mainly amplifies displacement on both source and held-out observations while maintaining high alignment.



(a) Latent L1 displacement

(b) Template agreement

(c) Effective rank

(d) Frozen-template L1 bound

Figure 7: Evolution of perturbation structure across observations during optimization. For three initializations, the panels show displacement on source and held-out observations, source-template agreement, the effective rank of 48 perturbation fields, and the ratio of the frozen source-template lower bound to measured L1 displacement. In (a), solid and dashed curves denote source and held-out states, respectively. Dotted lines show results with random patches.

Figure 7(a) shows that displacement grows on both source and held-out observations, indicating that the amplified perturbation extends beyond the source states. The early rise in template agreement in (b) and the effective rank approaching one in (c) indicate that a common direction emerges early and becomes dominant across observations. The high lower-bound ratio in (d) shows that the source-derived template accounts for a substantial fraction of held-out displacement even after subtracting the residual. Together, these trajectories indicate that optimization first establishes a shared direction and then mainly increases its magnitude.

Metrics and legend. At each optimization checkpoint in Figure 7, we measure the final-latent change ΔZ_i on the 356 spatial cells outside the projected patch footprint. The source template $\overline{\Delta Z}_{\mathrm{src}}$ is the mean displacement field over 12 source states at that checkpoint and is held fixed when evaluating the 36 held-out states. Panel (a) reports the median $\|\Delta Z_i\|_1$ separately for the source and held-out states. Panel (b) reports the median cosine similarity $\langle \overline{\Delta Z}_{\mathrm{src}}, \Delta Z_i \rangle / (\|\overline{\Delta Z}_{\mathrm{src}}\|_2 \|\Delta Z_i\|_2)$ over held-out states, with displacement fields flattened into vectors.

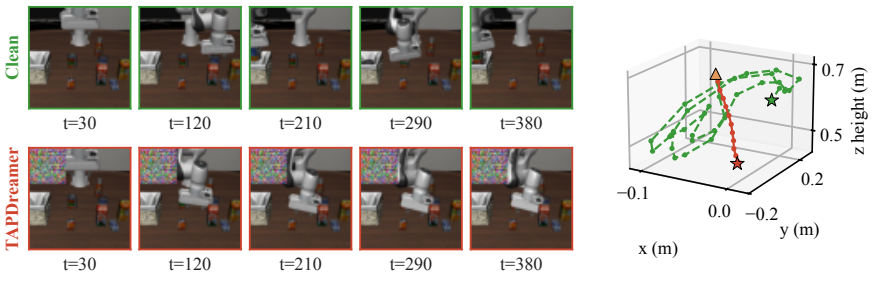
For panel (c), we stack the 48 flattened displacement fields as rows of a matrix. Its effective rank is $\exp(-\sum_j q_j \log q_j)$, where $q_j = \sigma_j^2 / \sum_k \sigma_k^2$ and σ_j are its singular values (with $0 \log 0 = 0$). A value near one indicates that the displacement fields largely share one direction. Panel (d) reports the median ratio $(\|\overline{\Delta Z}_{\mathrm{src}}\|_1 - \|\Delta Z_i - \overline{\Delta Z}_{\mathrm{src}}\|_1) / \|\Delta Z_i\|_1$ over held-out states: the frozen-template lower bound in Equation (8) divided by the measured L1 displacement.

Colors identify the three initializations in all panels. Only panel (a) uses solid lines with circles for source states and dashed lines with squares for held-out states; panels (b)–(d) show one summary curve per initialization. Gray dotted horizontal lines show the corresponding random-patch results, rather than an additional initialization.

B.3 Behavioral effects

The preceding analyses show how patch-induced perturbations propagate through attention and how optimization establishes a shared direction before amplifying its magnitude. We now ask how these representation changes affect action commands and, in turn, lead to task failure.

Visible execution failures. The same representation shift can affect different actions as a task unfolds. In Figure 8, the patch drives the end effector below its clean trajectory and prevents task completion. We use same-state action comparisons, internal restoration, and patch removal to connect the shift to commands and physical consequences.



(a) Put the alphabet soup and tomato sauce in the basket (b) End-effector trajectory

Figure 8: Example of closed-loop behavior under `TAPDreamer`. Left: observation sequences under clean and perturbed conditions. Right: corresponding three-dimensional end-effector position trajectories. Green dashed and red solid paths denote Clean and `TAPDreamer`, respectively. The triangle marks the starting position and stars mark the endpoints; t counts environment execution steps. Positions are in meters in the world coordinate frame.

Immediate action changes. Holding the physical state fixed, we compare actions from clean and patched observations before their trajectories diverge. In FastWAM, the patch suppresses lifting,

biases pose commands downward, and shifts gripper commands toward opening. A fit in which each patched six-dimensional pose command mixes its clean counterpart with a shared target explains 78.2% of the variation across 1,500 paired commands. These are state-level measurements, not independent episodes. The same bias can prevent a lift, lose an object after grasping, or move the end effector away from its intended position, depending on the task stage. These action patterns emerge from the encoder-level objective, which specifies no target action.

Action pathways across models. The disturbed visual representation also reaches actions through different downstream designs. In Motus, restoring the branch that represents visual world dynamics recovers about 93%–99% of the action change, whereas restoring the visual-language understanding branch recovers about 10%–27%. In DreamWAM-joint, restoring predicted future representations also substantially recovers actions. These interventions link encoder interference to control while showing that its route depends on the model's action pathway.

Continued exposure and recovery. Gripper commands shift toward closing at the first clean replan after removal, indicating that the immediate command bias depends on continued visual interference. Whether the task recovers then depends on the physical state: some pose deviations can be corrected, while dropped or misplaced objects and missed manipulation opportunities may persist. In the duration sweep, some tasks finish after removal following 20 corrupted replans, whereas continuous exposure maintains strong failure. Repeated exposure lets the fixed patch keep injecting the same latent change as execution accumulates physical consequences.