DISINTEGRATING THE VISUOMOTOR DEPENDENCY IN WORLD-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) integrate world modeling and action generation, offering a promising paradigm for robotic policy. While WAMs show strong potential, their robustness under adversarial attack remains underexplored. A defining characteristic of WAMs is the coupling between world modeling and action generation. How this coupling shapes action responses to semantic changes remains insufficiently characterized. We investigate how changes in visual observations propagate to downstream action generation. However, existing task-level metrics, e.g., success rate, primarily reflect final task outcomes, offering limited insight into the strength of this visuomotor dependency. We therefore introduce **Visuomotor Sensitivity (VMS)**, which quantifies the magnitude of action responses to controlled semantic transformations. Using VMS, we compare attention-pathway interventions in WAMs and identify visual-to-action attention as an internal attack target. Based on these findings, we propose **World-Action Disintegration (WAD)**, which combines visual-to-action attention suppression with a VMS-inspired surrogate targeting paired action differences across consecutive observations from the existing training dataset. Experiments show locally consistent VMS measurements and reproducibility across inference seeds. In simulation, we observe near-static behavior when minimizing the VMS-based surrogate and large-amplitude unstable motion when maximizing it; both conditions fail the task. WAD substantially degrades task performance across WAM architectures in simulation and in real-world trials. These findings characterize action responsiveness and its associated execution behaviors beyond aggregate action deviation and task success.

1 INTRODUCTION

World-Action Models (WAMs) are emerging as a promising paradigm for robotic policy by explicitly coupling world prediction and action generation (Bi et al., 2026; Li et al., 2026a; Ye et al., 2026; Yuan et al., 2026), establishing a tight interaction between world modeling and robotic actions. Despite this defining coupling, how world prediction actually influences action generation remains poorly understood. Recent studies further reveal that future prediction alone provides an incomplete characterization of action behavior: failures in world imagination do not necessarily translate into erroneous actions, while action failures can arise without correspondingly severe degradation of predicted futures (Chen et al., 2026; Li et al., 2026b). These observations suggest that understanding WAMs requires looking beyond world prediction itself and examining how world information is translated into robotic actions. As WAMs move toward real world deployment, such an insufficiently understood world-to-action process may also expose a critical attack surface, making its robustness particularly important in safety-sensitive robotic settings.

In WAMs, visual observations guide action generation, forming a visuomotor dependency (Levine et al., 2016) from visual perception to action prediction. However, existing evaluations mainly focus on task-level success rate and provide limited insight into how this dependency behaves under changes in visual observations. Fig. 1 contrasts appropriate, insufficient, and excessive responses to a target shift. Motivated by these differences, we introduce **Visuomotor Sensitivity (VMS)** to quantify action variation under controlled semantic transformations. This measurement characterizes differences in action responsiveness between well-functioning and impaired policies. Our experiments

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.

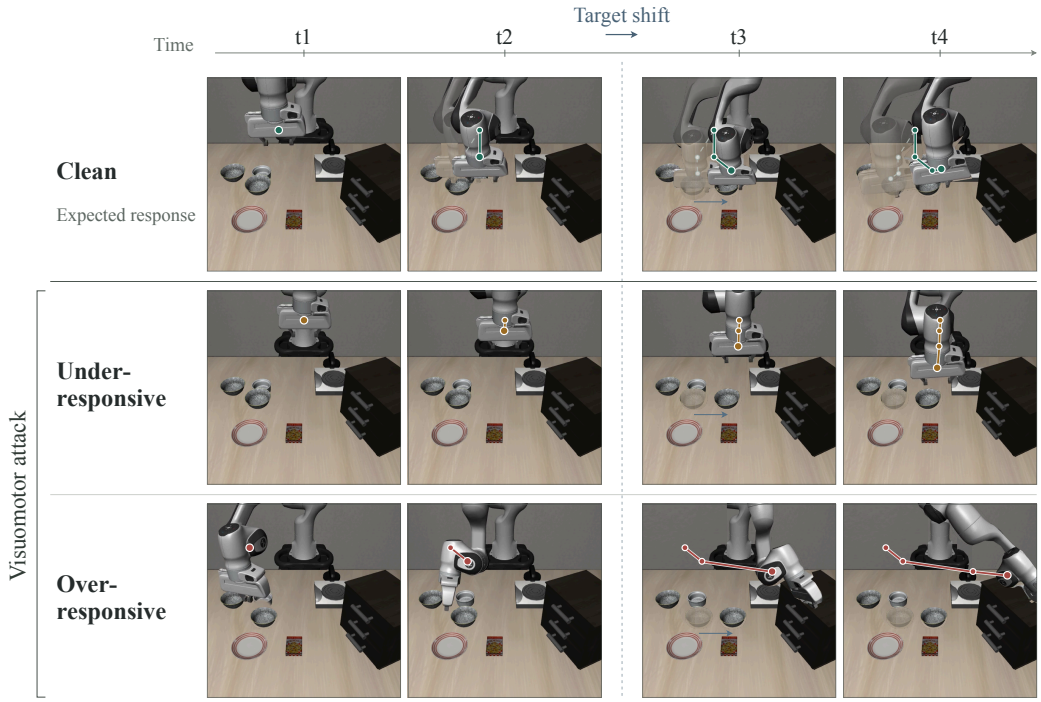

Figure 1: **Contrasting visuomotor responses to a target shift.** The clean policy adjusts to the shifted target, while under-responsive and over-responsive behaviors exhibit insufficient adjustment and excessive motion, respectively

show locally stable VMS under small semantic transformations and reproducible measurements across inference seeds. These results support VMS as a characterization of action response.

To further understand how the visuomotor dependency is instantiated within WAMs, we investigate the internal interactions between visual and action representations–the self-attention in WAMs. We decompose shared self-attention into visual-to-visual $A_{V\leftarrow V}$, action-to-action $A_{A\leftarrow A}$, visual-to-action $A_{A\leftarrow V}$ and action-to-visual attention $A_{V\leftarrow A}$ interactions. By examining the inference-time dynamics of these interactions, we find that $A_{A\leftarrow V}$ exhibits the most pronounced variation. Furthermore, we scale the logits of each available attention block separately across all layers, followed by the model's native masked softmax. VMS generally increases with the AV-logit scaling factor in the evaluated models, showing sensitivity to this intervention. These observations motivate visual-to-action attention as an internal attack surface. We evaluate this choice through attack ablations that compare the task success rates obtained by optimizing different attention objectives.

Building on these findings, we introduce **World-Action Disintegration (WAD)**, an adversarial attack on WAMs that combines two complementary objectives. At the output level, WAD minimizes a VMS-based surrogate, $\mathcal{L}_{\mathrm{VMS}}$, computed from paired action differences across consecutive observations drawn from the training dataset. At the attention level, WAD minimizes the mass allocated by action queries to visual tokens in $A_{A\leftarrow V}$. Our simulation experiments exhibit two failure modes: near-static behavior when minimizing the VMS-based surrogate and large-amplitude unstable motion when maximizing it. Extensive experiments across multiple WAM architectures show that WAD consistently degrades task performance by at least **30%** on LIBERO and RoboTwin, as well as in real-world deployments. Our main contributions are summarized as follows:

- We characterize action responsiveness in WAMs using **VMS**, finding consistency under semantic transformations and reproducibility across seeds. Simulations reveal near-static behavior when minimizing the VMS-based surrogate and large-amplitude unstable motion when maximizing it, characterizing execution behavior beyond task success.

- We propose **WAD**, which combines the VMS-based surrogate with suppression of visual-to-action attention $A_{A \leftarrow V}$. WAD reduces task success across the evaluated WAM architectures and benchmarks, attaining lower SR than a full-image action-deviation attack under the same perturbation constraints and number of optimization iterations. Real-world trials further demonstrate task degradation under WAD.

2 Related Work

World-Action Models. World-Action Models (WAMs) (Yuan et al., 2026; Bi et al., 2026; Li et al., 2026a; Ye et al., 2026; Guo et al., 2026; Cai et al., 2026; Kim et al., 2026; Qian et al., 2026; Zhou et al., 2026; Li et al., 2026c) extend Vision-Language-Action(VLA) (Kim et al., 2024; Wu et al., 2026; Wang et al., 2026; Black et al., 2025; Song et al., 2026) policies by incorporating visual dynamics or future prediction into learning, enabling world modeling and action generation (Chi et al., 2023) to be optimized within a unified policy framework. Recent WAMs adopt world-action formulations (Wan et al., 2025; NVIDIA, 2025; Ma et al., 2026). DreamZero (Ye et al., 2026) jointly generates future visual dynamics and actions, while LingBot-VA (Li et al., 2026a) couples visual dynamics modeling with action prediction. FastWAM (Yuan et al., 2026) shows that video modeling benefits action generation without future prediction at inference time. Despite architectural differences, these models share a causal dependence of action generation on visual or world representations. Existing studies prioritize policy performance and efficiency, leaving visuomotor responsiveness insufficiently characterized. We address this gap by characterizing action responsiveness with VMS and examining the associated visual-to-action interactions.

Adversarial Attacks on Robotic Policies. Adversarial attacks aim to induce erroneous model behaviors through crafted, imperceptible input perturbations (Goodfellow et al., 2015; Madry et al., 2018). Studying such attacks provides a way to expose model vulnerabilities and assess robustness under worst-case perturbations (Madry et al., 2018). Recent studies have extended adversarial attacks to robotic policies, showing that visual perturbations can manipulate VLA models to produce erroneous actions (Wang et al., 2025; Xu et al., 2025; Lu et al., 2026). Beyond directly perturbing action outputs, recent work has begun to target visual–action mechanisms. VLAGuard (Yin & Zhang, 2026) hijacks action-conditioned visual attention through physical adversarial patches, while SARF (Zhang et al., 2026) investigates policy-critical action-to-vision attention hijacking and its semantic disruption. BadWAM (Li et al., 2026b) targets world–action consistency by inducing drift between world prediction and executed actions. However, the sensitivity of action generation to visual changes remains underexplored as an attack surface. We characterize this sensitivity with VMS and exploit internal visual-to-action interactions to disrupt visuomotor responsiveness.

3 Method

3.1 Threat Model

Attack Goal. Given a visual observation $\mathbf{o}$ and non-visual conditions $\mathbf{c}$, a victim WAM policy π_θ predicts an action sequence $\mathbf{a} = \pi_\theta(\mathbf{o}, \mathbf{c})$. For each model–task setting, the adversary constructs an attack set from the training dataset. This set is disjoint from evaluation episodes. A single bounded perturbation $\boldsymbol{\delta}$ is optimized offline on these inputs. The perturbation is held fixed during evaluation, shared across evaluation episodes, and applied only to visual observations during execution.

Adversary Access. We assume white-box access to the victim policy, including its parameters, gradients, and intermediate attention representations. Offline attack preparation uses observation sequences and their non-visual inputs from the training dataset. The perturbation is optimized through forward and backward policy evaluations on these inputs, with model parameters frozen. Constructing the attack set and optimizing the perturbation require no closed-loop task rollouts.

Execution-Time Constraints. During attack execution, the adversary modifies only the visual input, producing $\mathbf{o}^{\mathrm{adv}} = \mathrm{clip}(\mathbf{o} + \boldsymbol{\delta})$ with $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$, where $\mathrm{clip}(\cdot)$ enforces the valid pixel range. The adversary does not directly modify the non-visual conditions, model parameters, or physical configuration of the target environment during execution.

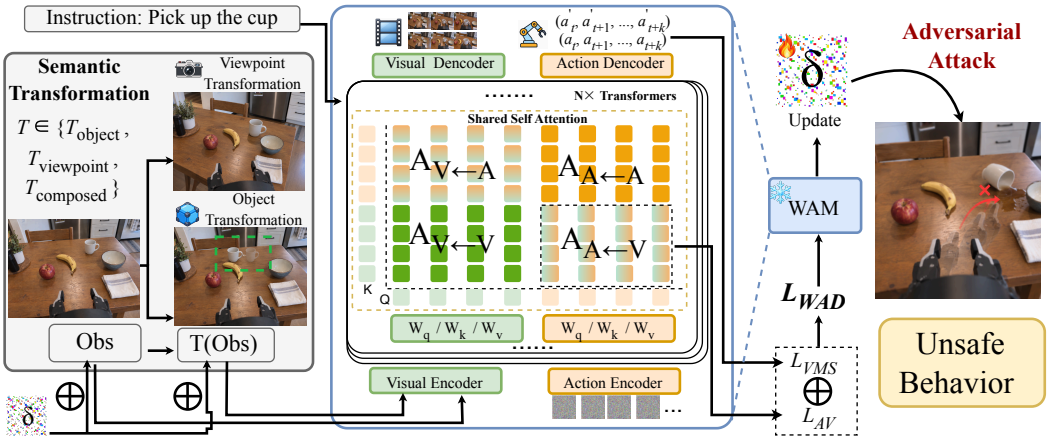

Figure 2: **Overview of World-Action Disintegration (WAD).** The semantic transformations illustrate the VMS analysis that motivates the paired-response objective. For offline WAD optimization, $\mathbf{o}$ and $\mathbf{o}'$ are consecutive observations drawn directly from the existing training dataset, with a shared bounded perturbation $\boldsymbol{\delta}$ applied to both. WAD combines visual-to-action attention suppression via $\mathcal{L}_{\text{AV}}$ with the VMS-based surrogate $\mathcal{L}_{\text{VMS}}$, which reduces paired action-prediction differences.

3.2 Visuomotor Sensitivity

For the well-trained robot policies, a fundamental property of robotic policies is their responsiveness to meaningful changes in visual observations. Meaningful variations in the visual scene should propagate through the policy and induce corresponding changes in action generation. We characterize this end-to-end visual-to-action responsiveness using VMS.

Given a visual observation $\mathbf{o}$ and its associated non-visual conditions $\mathbf{c}$, including the language instruction, proprioceptive state and other policy conditions, the policy produces the reference action $\mathbf{a} = \pi_\theta(\mathbf{o}, \mathbf{c})$. Throughout the VMS analysis and WAD optimization, π_θ denotes the predicted action chunk in the model's normalized action space, before action denormalization. We apply controlled semantic transformations while keeping the task instruction unchanged. We consider three types of semantic transformations: *object transformation* T_{object}, which changes the position of the task-relevant target object; *viewpoint transformation* $T_{\text{viewpoint}}$, which changes the pose of an arm-mounted camera; and *composed transformation* T_{composed}, which combines object and viewpoint transformations. The viewpoint condition concerns the arm-mounted camera, not the main-view camera. Given a transformation T, we write $\mathbf{o}' = T(\mathbf{o})$ for the resulting visual observation and $\mathbf{c}'$ for its associated non-visual conditions. The corresponding action is $\mathbf{a}' = \pi_\theta(T(\mathbf{o}), \mathbf{c}')$.

We quantify visuomotor responsiveness as the magnitude of the resulting action variation normalized by the magnitude of the corresponding visual variation. Accordingly, VMS is defined as

$$\text{VMS}(\mathbf{o}, T) = \frac{\sqrt{\frac{1}{d_a}\left\|\pi_\theta(T(\mathbf{o}), \mathbf{c}') - \pi_\theta(\mathbf{o}, \mathbf{c})\right\|_2^2}}{\sqrt{\frac{1}{d_o}\left\|T(\mathbf{o}) - \mathbf{o}\right\|_2^2}}, \tag{1}$$

where d_a and d_o denote the numbers of elements in the action output and visual observation.

VMS measures the magnitude of action response along the specified transformation, normalized by its visual change. When $\mathbf{c}' \neq \mathbf{c}$, the measurement includes the effects of the associated non-visual changes. A visual-only interpretation requires identical non-visual conditions and, for stochastic policies, matched inference randomness. In the camera-rotation tests in Table 1 and Appendix D.1, we vary camera orientation while holding the proprioceptive state fixed. These controlled viewpoint changes therefore do not introduce differences in proprioceptive state between the paired inputs. We examine whether well-trained WAMs exhibit locally consistent action responses as the magnitude of a semantic transformation varies, using a first-order approximation to characterize this behavior. For a fixed transformation type, we vary its magnitude η, with $T_0(\mathbf{o}) = \mathbf{o}$ and $\mathbf{c}_0 = \mathbf{c}$, while keeping

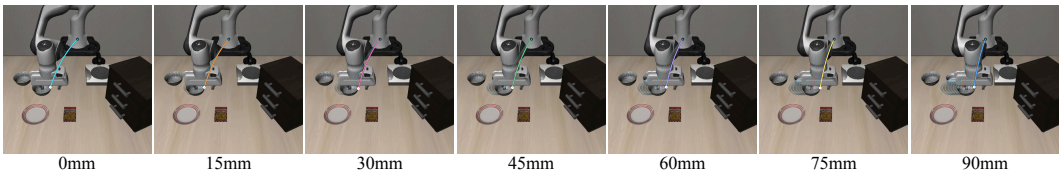

Figure 3: **Action-plan trajectories under same-direction semantic transformations.** We translate the target object from 0 to 90 mm while keeping other conditions fixed. The colored line in each panel denotes the planned end-effector trajectory predicted in a single inference. As the object shifts, the predicted action changes while preserving a consistent response direction.

Table 1: **Local response magnitude and consistency under semantic transformations.** Arm-mounted camera rotations are tested under fixed robot state and proprioceptive inputs. $\mathbf{d}_{\mathrm{VMS}}$ consistency is the cosine similarity between response vectors at neighboring transformation magnitudes.

Model	Metric	Camera view, θ				Object shift, s			
		2°	4°	6°	8°	1cm	2cm	3cm	4cm
Base model	VMS	0.051	0.038	0.033	0.031	0.079	0.048	0.045	0.041
	$\mathbf{d}_{\mathrm{VMS}}$ Consistency	—	0.865	0.565	0.842	—	0.487	0.698	0.913
Post-trained	VMS	0.524	0.514	0.516	0.522	0.295	0.281	0.301	0.284
	$\mathbf{d}_{\mathrm{VMS}}$ Consistency	—	0.919	0.942	0.986	—	0.905	0.917	0.952

the task instruction unchanged. Let $\mathbf{a}(\eta) = \pi_\theta(T_\eta(\mathbf{o}), \mathbf{c}_\eta)$ denote a predicted action chunk of length H. We define the **VMS direction** as the temporally aggregated action response:

$$\mathbf{d}_{\mathrm{VMS}}(\eta) = \sum_{t=1}^{H} \left(\mathbf{a}_t(\eta) - \mathbf{a}_t(0)\right). \tag{2}$$

This aggregation preserves the action coordinates while summarizing the net response across the chunk. Under local differentiability along the transformation path

$$\mathbf{d}_{\mathrm{VMS}}(\eta) \approx \eta\,\mathbf{g}, \qquad \mathbf{g} = \left.\frac{d\,\mathbf{d}_{\mathrm{VMS}}(\eta)}{d\eta}\right|_{\eta=0}, \tag{3}$$

where $\mathbf{g}$ captures the first-order response along the full input path, including changes in $\mathbf{c}_\eta$. Thus, small transformations should yield locally stable VMS magnitudes and high $\mathbf{d}_{\mathrm{VMS}}$ consistency.

Fig. 3 illustrates the resulting trajectory changes, while Table 1 evaluates VMS magnitude and $\mathbf{d}_{\mathrm{VMS}}$ consistency. We use cosine similarity to define the latter as

$$\mathrm{Cons}_{\mathbf{d}_{\mathrm{VMS}}}(\eta_i, \eta_{i-1}) = \cos(\mathbf{d}_{\mathrm{VMS}}(\eta_i), \mathbf{d}_{\mathrm{VMS}}(\eta_{i-1})), \tag{4}$$

for nonzero response vectors. For target-object translations of 1–4 cm and arm-mounted camera rotations of 2°–8°, VMS has a coefficient of variation below 10% within each transformation family, while the mean $\mathbf{d}_{\mathrm{VMS}}$ consistency exceeds 0.90, supporting local consistency over the tested range.

3.3 Analyzing Visual-to-Action Interaction

The preceding analysis characterizes visuomotor responsiveness at the output level. We next investigate how interventions on internal attention interactions affect this response. Specifically, we consider visual-to-action (AV), action-to-action (AA), visual-to-visual (VV), and, where available, action-to-visual (VA) attention. AV denotes the attention assigned by action queries to visual keys, providing a pathway through which visual information contributes to action-token updates.

We scale one selected logit block at a time across all attention layers. Let $\mathbf{Z}$ denote the attention logits before masking and softmax, $\mathbf{M}$ the native additive attention mask, and $\mathcal{B}$ the query–key index set of the selected block. At each layer, we apply

$$Z_{ij}^{(\gamma)} = \begin{cases} \gamma Z_{ij}, & (i,j) \in \mathcal{B}, \\ Z_{ij}, & \text{otherwise}, \end{cases} \qquad \mathbf{A}^{(\gamma)} = \mathrm{softmax}_{\mathrm{row}}\left(\mathbf{Z}^{(\gamma)} + \mathbf{M}\right).$$

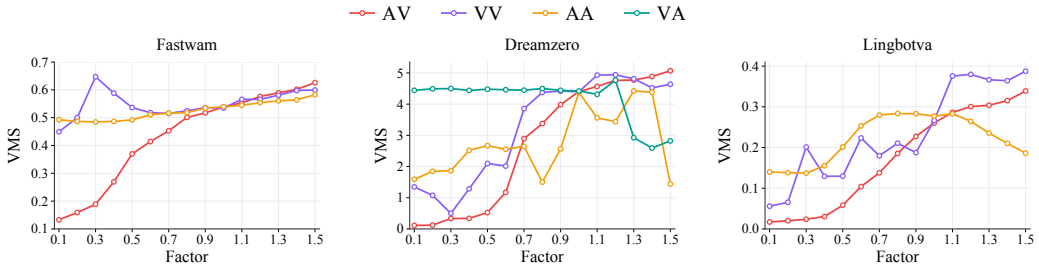

Figure 4: **Effect of attention-logit scaling on VMS.** We scale one selected logit block across all attention layers, followed by the native masked softmax without additional renormalization. VMS generally increases with the AV-logit scaling factor across FastWAM, DreamZero, and LingBot-VA, whereas other blocks exhibit weaker or non-monotonic responses.

The mask is preserved, and softmax normalizes each row over all permitted keys; no renormalization is applied. The setting $\gamma = 1$ recovers the attention operation. Although the operation leaves logits outside the selected block unchanged, softmax can change their attention probabilities. Thus, γ scales logits rather than attention mass, and increasing it need not increase that mass. As shown in Fig. 4, VMS increases with the AV-logit scaling factor in the models. The VA curve is reported only for DreamZero, since this pathway is masked in FastWAM and LingBot-VA. Together with the attention-activity measurements in Appendix E, these findings motivate AV as an attention-level attack target. The attack ablations in Sec. 4.4 assess its effectiveness through task success.

3.4 World-Action Disintegration

Based on our analysis in Secs 3.2 and 3.3, we propose **World-Action Disintegration (WAD)**, an adversarial attack that targets the visuomotor dependency underlying action generation in WAMs. The overview of WAD is illustrated in Fig. 2. WAD combines visual-to-action attention suppression with a VMS-inspired surrogate that reduces action-prediction differences across consecutive observations. We draw consecutive policy inputs $(\mathbf{o}_t, \mathbf{c}_t)$ and $(\mathbf{o}_{t+1}, \mathbf{c}_{t+1})$ directly from a trajectory in the existing training dataset. For brevity, let $(\mathbf{o}, \mathbf{c}) = (\mathbf{o}_t, \mathbf{c}_t)$ and $(\mathbf{o}', \mathbf{c}') = (\mathbf{o}_{t+1}, \mathbf{c}_{t+1})$. We apply the same perturbation $\boldsymbol{\delta}$ to both visual observations

$$\mathbf{o}_\delta = \mathrm{clip}(\mathbf{o} + \boldsymbol{\delta}), \qquad \mathbf{o}'_\delta = \mathrm{clip}(\mathbf{o}' + \boldsymbol{\delta}). \tag{5}$$

Each observation retains its non-visual conditions during optimization, including its proprioceptive state. These conditions are held fixed for each observation, but $\mathbf{c}$ and $\mathbf{c}'$ may differ across the pair.

Visual-to-action attention suppression. For shared self-attention over visual and action tokens, we partition the post-softmax attention matrix by query and key modality, ordered as (V, A)

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{V\leftarrow V} & \mathbf{A}_{V\leftarrow A} \\ \mathbf{A}_{A\leftarrow V} & \mathbf{A}_{A\leftarrow A} \end{bmatrix}, \tag{6}$$

all blocks respect the model's native attention masks. We target $\mathbf{A}_{A\leftarrow V}$, which quantifies the attention allocation from action queries to visual keys, motivated by the empirical experiments above.

We define the visual-to-action attention loss as the average AV attention mass across layers, heads, and policy sampling steps:

$$\mathcal{L}_{\mathrm{AV}}(\boldsymbol{\delta}) = \frac{1}{|\mathcal{S}|} \sum_{(l,h,s)\in\mathcal{S}} \frac{1}{|\mathcal{Q}_A|} \sum_{i\in\mathcal{Q}_A} \sum_{j\in\mathcal{K}_V} A_{ij}^{l,h,s}(\mathbf{o}_\delta, \mathbf{c}), \tag{7}$$

where $\mathcal{Q}_A$ and $\mathcal{K}_V$ are the sets of action queries and visual keys, respectively, and $\mathcal{S}$ indexes layer–head–step tuples. As attention weights sum to one over keys, $\mathcal{L}_{\mathrm{AV}}$ measures the average attention mass that action queries assign to visual tokens, to suppress visual-to-action attention.

VMS-based surrogate objective. For offline attack optimization, we use consecutive observations from the training dataset as natural transitions to construct an objective. Unlike the camera-rotation

tests with fixed robot state in Table 1 and Appendix D.1, these pairs may differ in proprioceptive state and task progress; thus, their action differences do not isolate visual sensitivity. We perturb only the visual inputs and minimize the paired action difference

$$\mathcal{L}_{\mathrm{VMS}}(\boldsymbol{\delta}) = \frac{\sqrt{\frac{1}{d_a}\left\|\pi_\theta(\mathbf{o}'_\delta, \mathbf{c}') - \pi_\theta(\mathbf{o}_\delta, \mathbf{c})\right\|_2^2}}{\sqrt{\frac{1}{d_o}\left\|\mathbf{o}_\delta{}' - \mathbf{o}_\delta\right\|_2^2}}. \tag{8}$$

The denominator is computed from the perturbed observation pair and kept fixed during optimization. We therefore use $\mathcal{L}_{\mathrm{VMS}}$ as a surrogate rather than a direct estimate of visual sensitivity.

Adversarial optimization. We combine attention suppression and the paired-response surrogate

$$\mathcal{L}_{\mathrm{WAD}}(\boldsymbol{\delta}) = \mathcal{L}_{\mathrm{VMS}}(\boldsymbol{\delta}) + \lambda \mathcal{L}_{\mathrm{AV}}(\boldsymbol{\delta}), \tag{9}$$

where λ controls the relative contribution of attention suppression and is set to 1 in all WAD experiments. We optimize the shared perturbation $\boldsymbol{\delta}$ under an ℓ_∞ constraint

$$\boldsymbol{\delta}^* = \arg\min_{\|\boldsymbol{\delta}\|_\infty \le \epsilon} \mathcal{L}_{\mathrm{WAD}}(\boldsymbol{\delta}). \tag{10}$$

We solve this constrained optimization using projected gradient descent (PGD) (Madry et al., 2018). Starting from $\boldsymbol{\delta}^0$ within the feasible region, the perturbation is iteratively updated as

$$\boldsymbol{\delta}^{k+1} = \Pi_{\mathcal{B}_\infty(\epsilon)}\left[\boldsymbol{\delta}^k - \alpha \operatorname{sign}\left(\nabla_{\boldsymbol{\delta}^k} \mathcal{L}_{\mathrm{WAD}}(\boldsymbol{\delta}^k)\right)\right], \tag{11}$$

where α is the step size and $\Pi_{\mathcal{B}_\infty(\epsilon)}$ denotes projection onto the ℓ_∞ ball of radius ϵ. After K iterations, we obtain the adversarial observation as $\mathbf{o}^{\mathrm{adv}} = \mathrm{clip}(\mathbf{o} + \boldsymbol{\delta}^K)$.

4 Experiments

4.1 Experimental Setup

Models and benchmarks. We evaluate WAD across three WAM paradigms: action-only generation with FastWAM (Yuan et al., 2026), inverse-dynamics modeling(inverse-dynamics modeling) with LingBot-VA (Li et al., 2026a), and joint world-action modeling with DreamZero (Ye et al., 2026) and Motus (Bi et al., 2026). Experiments are conducted on LIBERO (Liu et al., 2023) and RoboTwin (Chen et al., 2025). Further model and benchmark details are provided in Appendix B.

Attack settings and controls. WAD is optimized offline on consecutive observations from the training dataset, disjoint from evaluation episodes, and the perturbation is fixed during evaluation. We compare clean observations, random noise, VLA-Patch adapted from UADA (Wang et al., 2025), a full-image white-box action-deviation baseline (Action-Dev), and our WAD. Random noise and Action-Dev use the model-specific ℓ_∞ budget as WAD. VLA-Patch uses a 5% main camera replacement patch with valid RGB values and no ℓ_∞ bound; WAD,Random noise and Action-Dev perturb the full images of all camera streams. Attack settings are described in Appendices B.3 and B.4.

Evaluation metrics. We report task success rate (SR, %) as the primary measure of task performance. For controlled semantic transformations, we use VMS to measure response magnitude and VMS direction to assess consistency between aggregate action-response directions. For the attack analysis, we separately report the VMS-based surrogate in Eq. 8 on fixed observation pairs.

Table 2: **Main attack results across WAM architectures.** We report task SR (%) under clean observations(Clean), random noise (Noise), VLA-Patch (Patch), Action-Dev (Action), and WAD. For the joint-model results, DreamZero is evaluated only on LIBERO and Motus only on RoboTwin.

WAM Type	LIBERO					RoboTwin				
	Clean	Noise	Patch	Action	WAD	Clean	Noise	Patch	Action	WAD
Action-only	97.6	71.4	95.2	30.2	**0.0**	91.9	70.4	89.1	20.7	**0.0**
IDM	98.6	95.6	97.4	92.4	**64.0**	92.9	90.1	91.4	89.1	**62.6**
Joint	96.8	90.6	95.2	71.4	**62.6**	88.7	83.3	80.1	73.1	**51.4**

4.2 Main Attack Results

Attack effectiveness across WAM architectures. Table 2 reports task success rate under clean observations, random noise, the adapted patch baseline, Action-Dev, and WAD. WAD attains lower SR than these controls in every model–benchmark setting. Action-Dev provides a comparison under matched attack data, perturbation constraints, initialization, minibatch sequence, policy-sampling randomness, step size, and iteration count. VLA-Patch leaves SR close to the clean level under its configuration. Because its spatial and camera scope differs from WAD's, this result neither isolates objective quality nor establishes that localized perturbations are generally ineffective against WAMs.

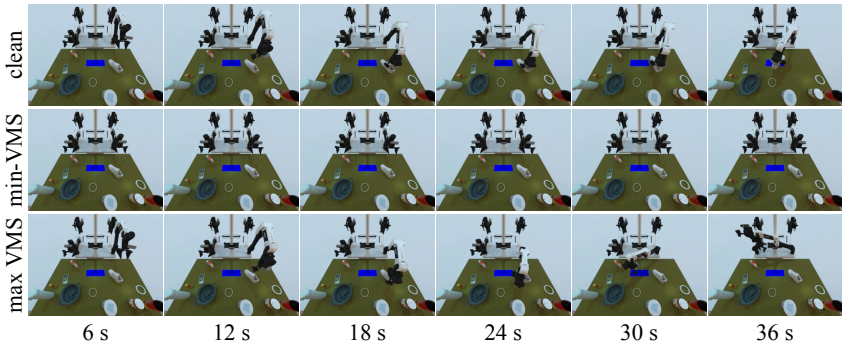

Figure 5: **Behaviors under paired-response suppression and amplification.** RoboTwin `place_shoe` rollouts under Clean, Min-VMS, and Max-VMS conditions, from top to bottom. Min-VMS and Max-VMS minimize and maximize the surrogate in Eq. 8, respectively.

Execution behaviors under paired-response suppression and amplification. Fig. 5 shows representative simulation rollouts under perturbations that minimize or maximize the VMS-based surrogate in Eq. 8, labeled Min-VMS and Max-VMS, respectively. Min-VMS leads the robot to little or no motion; Max-VMS causes large-amplitude, unstable motion. Both conditions fail the task. These rollouts reveal distinct behaviors associated with surrogate minimization and maximization, complementing the aggregate SR comparison. More examples are provided in Appendix C.

4.3 Empirical analysis of VMS

Under small semantic transformations, the post-trained policy exhibits locally stable VMS and aligned aggregate action-response directions, as shown in Table 1. For camera rotations in this analysis, we hold the robot state and proprioceptive input fixed. These results are consistent with the local first-order behavior described in Sec. 3.2. VMS and aggregate response directions also remain stable across stochastic inference seeds, as shown in Appendix D.1. Fig. 6 shows a non-monotonic association between the VMS-based surrogate and task SR along an interpolated perturbation path, with SR peaking at intermediate surrogate values among perturbed conditions. The horizontal axis reports the surrogate in Eq. 8; the blue and red curves show SR and the applied ℓ_2 perturbation norm, respectively. The clean reference has a surrogate value of 0.722 and achieves 98% SR. The experimental protocol and perturbation-norm analysis are provided in Appendix D.2.

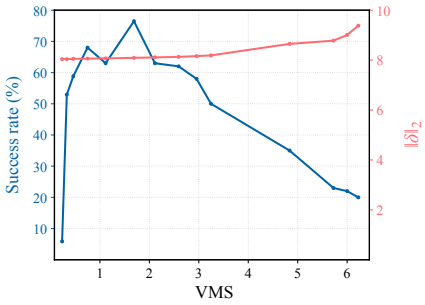

Figure 6: **Task success versus the VMS-based surrogate.**

4.4 Ablation Studies

Both ablations are conducted on LIBERO-Spatial under fixed checkpoints and evaluation protocols. Table 3 compares suppressing action-to-action attention $A_{A \leftarrow A}$ and visual-to-action attention $A_{A \leftarrow V}$ on DreamZero and FastWAM to identify which attention pathway is more critical to task execution. Visual-to-action suppression yields lower task success rates for both models, supporting

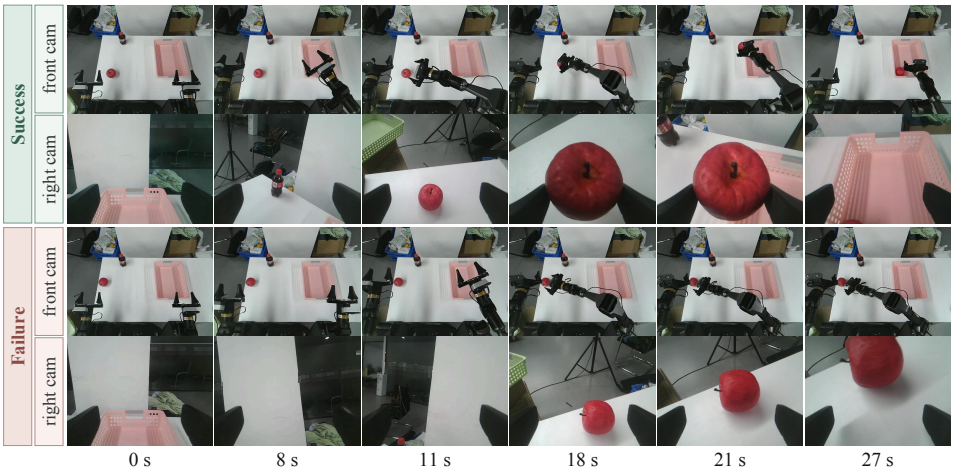

Figure 7: **Real-world robot execution before and after attack.** Within each timestamp, the upper and lower views correspond to the head and right-arm cameras. The top shows the clean rollout with smooth task execution, while the bottom shows the attacked rollout with less responsive motion.

Table 3: **Attention-objective analysis(SR,%).**

Objective	DreamZero	FastWAM
Clean	96.8	98.2
Minimize $A_{A\leftarrow A}$	87.6	37.0
Minimize $A_{V\leftarrow A}$	72.2	–
Minimize $A_{V\leftarrow V}$	76.4	–
Minimize $A_{A\leftarrow V}$	**68.2**	**0.00**

Table 4: **Component ablation.**

Method	$\mathcal{L}_{\text{AV}}$	$\mathcal{L}_{\text{VMS}}$	SR (%)
Clean	–	–	96.6
Random Noise	–	–	90.6
w/o AV	×	✓	78.2
w/o VMS	✓	×	68.2
WAD	✓	✓	**62.6**

its selection as the attention target. Table 4 further evaluates the individual contributions of attention suppression and the VMS-based surrogate, and compares them with the joint WAD objective to assess whether the two components provide complementary effects.

4.5 Real-World Evaluation

We evaluate WAD on FastWAM deployed on an ARX dual-arm robot performing a real-world pick-and-place task, with perturbations digitally applied to camera observations before policy inference. Over 20 trials per condition, the clean policy succeeds in $13/20$ trials (65% SR), whereas WAD reduces the success rate to 0%. As shown in Fig. 7, the clean policy produces smooth and responsive motions, while WAD substantially suppresses robot responses and prevents task completion. Additional setup details are provided in Appendix F.

5 Conclusion

In this work, we characterize action responsiveness in WAMs through **VMS**, observing local consistency under semantic transformations and reproducibility across inference seeds. Experiments show near-static behavior when minimizing the VMS-based surrogate and large-amplitude unstable motion when maximizing it, with task failure in both cases. Guided by VMS analyses of attention interventions, we propose **WAD**, which combines visual-to-action attention suppression with this surrogate on consecutive observations from the training dataset, reducing task success across WAM architectures and in real-world trials. These findings support examining paired action responses and execution behaviors alongside action deviation and task success when evaluating WAM robustness.

Ethics Statement

This work studies adversarial vulnerabilities in World-Action Models with the goal of improving the safety and robustness of embodied AI systems. Our experiments are conducted on public simulation benchmarks and controlled robotic platforms, without targeting deployed third-party systems or causing harm to humans or property. While the proposed attacks reveal failure modes that could potentially be misused, we believe that systematically characterizing these vulnerabilities is important for understanding the security risks of increasingly capable robotic policies and for motivating more robust model design and evaluation. We encourage future work to investigate effective defenses against the identified visuomotor vulnerabilities.

Reproducibility Statement

We document experimental settings to facilitate reproducibility. The evaluated models, benchmarks, checkpoint sources, and evaluation metrics are described in Sec. 4.1 and Appendix B. The method and appendix distinguish controlled VMS analysis from the VMS-inspired surrogate used for attack optimization on consecutive training observations. They describe normalized action-space computation, attention aggregation, and the WAD loss weight and projected sign-gradient update. Appendices B.3 and B.4 specify the data selection, batching, initialization, sampling noise, and continuous-action UADA adaptation used in our experiments. Appendix B.5 lists the evaluated RoboTwin tasks and benchmark evaluation protocols. The appendix also describes the VMS analyses and real-robot setup. All experiments are based on publicly available benchmarks and model checkpoints unless otherwise specified. Code and evaluation scripts will be released to facilitate reproduction of the reported 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*, 2026.

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

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, Ru Ying, Ran Zheng, and Yao Mu. Aha-wam: Asynchronous horizon-adaptive world-action modeling with observation-guided context routing. *arXiv preprint arXiv:2606.09811*, 2026.

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.

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

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

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4D world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, et al. Openvla: An open-source vision-language-action model. In *CoRL*, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations*, 2026.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Luyao Zhang, Mingrui Yu, Zelin Gao, Nan Xue, Boyu Zhou, Xing Zhu, Mingyu Ding, Yujun Shen, and Yinghao Xu. Causal World Modeling for Robot Control. In *RSS*, 2026a.

Qi Li, Xingyi Yang, and Xinchao Wang. Badwam: When world-action models dream right but act wrong. *arXiv preprint arXiv:2607.15207*, 2026b.

Ziang Li, Dongzhou Cheng, Yibin Wang, Shiyue Wang, Xiaoyang Xu, Lingxuan Weng, Juan Wang, and Jiaqi Wang. Light-wam: Efficient world action models with state-fusion action decoding. *arXiv preprint arXiv:2606.08242*, 2026c.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023.

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.

Shuailei Ma, Jiaqi Liao, Xinyang Wang, Jingjing Wang, Chaoran Feng, Zijing Hu, Chong Bao, Zichen Xi, Yuqi Gan, Weisen Wang, Yanhong Zeng, Qin Zhao, Zifan Shi, Wei Wu, Hao Ouyang, Qiuyu Wang, Shangzhan Zhang, Jiahao Shao, Yipengjing Sun, Liangxiao Hu, Lunke Pan, Nan Xue, Kecheng Zheng, Yinghao Xu, Xing Zhu, Yujun Shen, and Ka Leong Cheng. Scaling mixture-of-experts video pretraining for embodied intelligence. *arXiv preprint arXiv:2607.07675*, 2026.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

NVIDIA. Cosmos-predict2: World simulation model for physical ai, 2025. URL `https://github.com/nvidia-cosmos/cosmos-predict2`.

Zezhong Qian, Xiaowei Chi, Yu Qi, Haozhan Li, Zhi Yang Chen, and Shanghang Zhang. Wa-rl: World-action model reinforcement learning with reconstruction rewards and online video sft. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 4478–4485, 2026.

Wenxuan Song, Ziyang Zhou, Han Zhao, Jiayi Chen, Pengxiang Ding, Haodong Yan, Yuxin Huang, Feilong Tang, Donglin Wang, and Haoang Li. Reconvla: Reconstructive vision-language-action model as effective robot perceiver. In *AAAI*, 2026.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Taowen Wang, Cheng Han, James Liang, Wenhao Yang, Dongfang Liu, Luna Xinyu Zhang, Qifan Wang, Jiebo Luo, and Ruixiang Tang. Exploring the adversarial vulnerabilities of vision-language-action models in robotics. In *ICCV*, 2025.

Yihao Wang, Pengxiang Ding, Lingxiao Li, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. In *AAAI*, 2026.

Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, Bowen Yang, Zhe Li, Kai Zhu, Hongyu Wu, Yiheng Liu, et al. Robocoin: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Shuailei Ma, He Sun, Yong Wang, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Shuai Zhou, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Qian Zhu, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692v1*, 2026.

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.

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. In *CoRL*, 2026.

Dongfu Yin and Jinquan Zhang. Vlaguard: A framework for evaluating and mitigating physical attention hijacking in vision-language-action robots within wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, 2026. Accepted for publication.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Jiakun Du, Xiangyuan Wang, Xu Fu, Letong Shi, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. RLinf: Flexible and efficient Large-Scale reinforcement learning via Macro-to-Micro flow transformation. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*. USENIX Association, 2026. URL https://www.usenix.org/conference/osdi26/presentation/yu-chao.

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*, 2026.

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. In *IROS*, 2026.

Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, Xingyu Qiu, Chenhao Qiu, Pu Yang, Yunuo Cai, Jianxiong Gao, Yifan Li, Yanwei Fu, Xiangyu Yue, Zhi Chen, and Jianlan Luo. τ_0-wm: A unified video-action world model for robotic manipulation. *arXiv preprint arXiv:2606.01027*, 2026.

A LLM USAGE STATEMENT

We used ChatGPT to assist with language editing and drafting implementation descriptions.

B MODEL, INFERENCE, AND ATTACK CONFIGURATIONS

We evaluate four representative WAMs covering three modeling paradigms: FastWAM (Yuan et al., 2026) for action-only generation, LingBot-VA (Li et al., 2026a) for IDM, and DreamZero (Ye et al., 2026) and Motus (Bi et al., 2026) for joint world-action modeling. Experiments are conducted on LIBERO (Liu et al., 2023) and RoboTwin (Chen et al., 2025). Table 5 summarizes the benchmark coverage, perturbation budgets, and action-chunk configurations.

Table 5: **Model configurations used in our experiments.**

Model	Paradigm	Benchmark	ϵ	Action Chunk
FastWAM	Action-only	LIBERO/RoboTwin	$6/255$	32
LingBot-VA	IDM	LIBERO-Long/RoboTwin	$8/255$	16/32
DreamZero	Joint	LIBERO-Spatial	$8/255$	16
Motus	Joint	RoboTwin	$8/255$	16

B.1 MODEL-SPECIFIC CHECKPOINTS

FastWAM. FastWAM represents the action-only generation paradigm. For LIBERO, we use the released checkpoint `libero_uncond_2cam224` and evaluate all four suites: Spatial, Object, Goal, and Long. For RoboTwin, we use `robotwin_uncond_3cam_384`. For the VMS analysis in Sec. 3.2, we additionally use the RoboCOIN-pretrained FastWAM base checkpoint released by the AHA-WAM community (Cai et al., 2026; Wu et al., 2025). The real-world policy is initialized from `robotwin_uncond_3cam_384` and further fine-tuned on our ARX demonstrations. Real-world adaptation details are provided in Appendix F.

LingBot-VA. LingBot-VA represents the IDM paradigm. We use the released checkpoints `lingbot-va-posttrain-libero-long` and `lingbot-va-posttrain-robotwin` for LIBERO-Long and RoboTwin. Following the released inference pipeline, visual prediction precedes action decoding, and the action decoder conditions on the predicted visual representation.

DreamZero and Motus. DreamZero and Motus represent joint world-action modeling. Due to the availability of benchmark-specific checkpoints, we evaluate DreamZero on LIBERO-Spatial and Motus on RoboTwin. For DreamZero, we use the RLinf (Yu et al., 2026) checkpoint `RLinf-DreamZero-WAN2.2-5B-LIBERO-SFT-Step26000`. For Motus, we use the official RoboTwin checkpoint `Motus_robotwin2`.

B.2 INFERENCE AND PREPROCESSING

We retain the observation and action interfaces associated with each released checkpoint. Input images are converted to $[0, 1]$ and processed using the model-specific resizing and cropping operations before policy inference. Actions are normalized and denormalized using the statistics associated with each checkpoint.

Perturbations are applied after geometric image preprocessing and before model-specific channel normalization, followed by clipping to $[0, 1]$. For multi-camera policies, a separate perturbation is optimized for each camera stream and shared across its corresponding temporal frames. When WAD operates on paired observations, the same additive perturbation is applied to both observations.

Both VMS and the VMS-based attack surrogate are computed over the full predicted action chunk in normalized action space before action denormalization. The corresponding action-chunk lengths are reported in Table 5.

B.3 Attack Optimization

We construct the attack set from the existing training dataset, disjoint from evaluation episodes. Consecutive observations within selected trajectories and their associated non-visual inputs form the optimization pairs. WAD is optimized offline on these stored observations without additional policy rollouts, and the resulting perturbation remains fixed throughout evaluation. Only visual observations are modified; model parameters and non-visual inputs remain unchanged.

Attack data and batches. For each model and task, we select 10 eligible training trajectories uniformly without replacement and sample 32 valid consecutive observation pairs per trajectory uniformly without replacement, giving 320 pairs. We use fewer trajectories or pairs only when insufficient data are available and record the actual counts. A valid pair consists of adjacent stored steps within one trajectory, has a nonzero clean visual difference after preprocessing, and has a complete expert action chunk available at each endpoint; an eligible trajectory contains at least one such pair. We denote the nonempty pair collection by $\mathcal{P}_{\mathrm{attack}}$ and its endpoint multiset by $\mathcal{D}_{\mathrm{attack}}$; 320 pairs yield 640 endpoint entries, which need not be distinct observations. At each update, we independently sample $B = \min(4, |\mathcal{P}_{\mathrm{attack}}|)$ pairs uniformly without replacement within the batch. WAD averages its joint loss over these pairs, with the attention term evaluated at the first endpoint as in Eq. 7. Action-Dev uses the same batch's $2B$ endpoints, retaining multiplicities. The 120 iterations are minibatch updates, not epochs.

WAD jointly optimizes the visual-to-action attention objective and the VMS-based surrogate described in Sec. 3.4, with the loss weight fixed to $\lambda = 1$. Each observation retains its associated non-visual condition, which may differ across consecutive observations. Visual-to-action attention is aggregated over all layers and attention heads used during policy inference.

Unless otherwise specified, we use 120 projected sign-gradient descent updates with a step size of $\alpha = 1/255$, as defined in Eq. 11. The model-specific ℓ_∞ perturbation budgets follow Table 5. Each update is projected onto the corresponding ℓ_∞ ball, and perturbed observations are clipped to $[0, 1]$. The optimization objective is shared across models, while attention extraction and forward-pass implementations are adapted to each architecture.

Initialization and perturbation sharing. We initialize each perturbation entry independently from $\mathcal{U}[-\epsilon, \epsilon]$ and use one optimization run with no restarts, momentum, or learning-rate schedule. We use seed 42 with separate random-number streams for data selection, batch sampling, initialization, and policy sampling. We learn a separate perturbation for each model, task, and camera stream, shared across that task's attack observations, temporal frames, and evaluation episodes. This yields a task-specific universal perturbation. We retain the final iterate for evaluation.

Sampling noise. For each pair i, we draw and store policy-sampling randomness ξ_i. We replay the same randomness for its two observations and across optimization updates: the paired action predictions are $\pi_\theta(\mathbf{o}_{i,\delta}, \mathbf{c}_i; \xi_i)$ and $\pi_\theta(\mathbf{o}'_{i,\delta}, \mathbf{c}'_i; \xi_i)$. This includes initial noise and any additional stochastic sampler draws; inference runs in evaluation mode. Different pairs receive independent randomness. Shared perturbations and shared sampling randomness are separate choices: the former alone does not remove sampling variability from the paired action difference. The method equations suppress the noise arguments for readability.

We additionally examine the effects of perturbation magnitude and optimization length. As shown in Fig. 8, task success decreases as ϵ increases. Fig. 9 shows a similar trend with additional PGD updates, with SR approaching zero after sufficient optimization. These results support the default choice of 120 optimization iterations.

B.4 Attack Baselines and Comparison Conditions

We compare five conditions: clean observations, random noise, VLA-Patch (Wang et al., 2025), the full-image white-box action-deviation baseline (Action-Dev), and WAD. All conditions use the same model checkpoints and evaluation protocols. Random noise follows the same perturbation scope and model-specific ℓ_∞ budget as WAD.

Action-deviation baseline. Action-Dev directly maximizes the deviation between attacked and clean action predictions over the full normalized action chunk. Let $\bar{\mathbf{a}}_{\boldsymbol{\delta}}(\mathbf{o}, \mathbf{c}; \xi)$ denote the vectorized normalized action chunk predicted from $\mathrm{clip}(\mathbf{o}+\boldsymbol{\delta})$ with sampling randomness ξ, and let $\bar{\mathbf{a}}_{\mathbf{0}}(\mathbf{o}, \mathbf{c}; \xi)$

denote its clean counterpart. Index attack inputs by r and denote their assigned randomness by ζ_r. The optimization objective is

$$\max_{\|\boldsymbol{\delta}\|_\infty \le \epsilon} \mathcal{J}_{\mathrm{AD}}(\boldsymbol{\delta}),$$
$$\mathcal{J}_{\mathrm{AD}}(\boldsymbol{\delta}) = \frac{1}{|\mathcal{D}_{\mathrm{attack}}|\, d_a} \sum_{r=1}^{|\mathcal{D}_{\mathrm{attack}}|} \|\bar{\mathbf{a}}_{\boldsymbol{\delta}}(\mathbf{o}_r, \mathbf{c}_r; \zeta_r) - \bar{\mathbf{a}}_{\mathbf{0}}(\mathbf{o}_r, \mathbf{c}_r; \zeta_r)\|_2^2 . \tag{12}$$

where d_a denotes the number of elements in the predicted action chunk.

Each attacked–clean comparison uses identical non-visual inputs and sampling noise. Clean action predictions are computed and cached before optimization, and attacked predictions use the same sampling noise as their corresponding clean references. Action-Dev matches WAD in attack data, perturbation-sharing scheme, image preprocessing, model-specific ℓ_∞ budget, and the number of optimization iterations.

Action-Dev optimizer. We use projected sign-gradient ascent with $K = 120$ updates and a constant step size $\alpha_{\mathrm{AD}} = 1/255$, without momentum, scheduling, or restarts. Action-Dev shares WAD's nonzero random initialization tensor, minibatch sequence, and per-pair sampling randomness: for either endpoint r of pair i, we set $\zeta_r = \xi_i$. We ascend the mean squared deviation over the $2B$ batch endpoints and project each update to $[-\epsilon, \epsilon]$. We cache clean predictions by pair identity and endpoint, since an observation appearing in multiple pairs can have different assigned randomness, and treat cached predictions as constants during differentiation.

Action-Dev and the VMS-based surrogate optimize different quantities. Action-Dev maximizes deviation from the clean prediction at each observation, whereas Eq. 8 compares predictions across consecutive observations under a shared perturbation. The baseline provides a control matched in attack data, perturbation constraints, initialization, minibatch sequence, policy-sampling randomness, step size, and iteration count. These matched settings support comparison under the specified optimization protocol without implying that either objective reaches its global optimum.

Patch baseline. VLA-Patch adapts the Untargeted Action Discrepancy Attack (UADA) of Wang et al. (2025) to each WAM's visual input interface. Original UADA forms a differentiable soft action from action-bin probabilities and minimizes its squared distance to an action-space boundary chosen to be farthest from the ground-truth action. Its reference is therefore the expert action, not a clean policy prediction. Our continuous-action adaptation uses the WAM's predicted action chunk and the expert-action-based targets defined below.

Continuous-action UADA. Let $\bar{\mathbf{a}}_r^E \in \mathbb{R}^{H \times D}$ be the normalized expert action chunk aligned with input r, and let $\bar{\mathbf{a}}_{p,r}(\tau; \zeta_r)$ be the WAM's continuous predicted chunk when a transformed patch $\tau(p)$ replaces the selected main-camera region. Here τ denotes the sampled patch transformation; all other camera streams remain unchanged. We use the same checkpoint normalization for predictions and expert actions. For each action coordinate j, we define ℓ_j and u_j as the minimum and maximum normalized expert values over the selected attack trajectories. These are empirical target bounds, not assumed hard limits on the model's outputs. We fix the expert reference, bounds, and far-end targets before optimization

$$b_{r,h,j} = \begin{cases} u_j, & |u_j - \bar{a}^E_{r,h,j}| \ge |\ell_j - \bar{a}^E_{r,h,j}|, \\ \ell_j, & \text{otherwise}, \end{cases} \tag{13}$$

$$\mathcal{L}_{\mathrm{UADA}}(p) = \frac{1}{|\mathcal{D}_{\mathrm{attack}}| HD} \sum_{r=1}^{|\mathcal{D}_{\mathrm{attack}}|} \mathbb{E}_\tau \Big[\|\bar{\mathbf{a}}_{p,r}(\tau; \zeta_r) - \mathbf{b}_r\|_F^2 \Big] . \tag{14}$$

We minimize $\mathcal{L}_{\mathrm{UADA}}(p)$ over $p \in [0,1]^{d_p}$, where d_p is the number of patch entries. The Frobenius norm covers all $HD = d_a$ action-chunk entries. Ties select the upper bound; a constant coordinate with $\ell_j = u_j$ retains that fixed target. We sample 6 endpoint entries uniformly with replacement per patch update, keep their assigned policy randomness ζ_r fixed, and sample fresh patch transformations to estimate the expectation. The continuous prediction replaces the original action-bin soft output, with gradients propagated through the WAM inference computation to the patch. Each coordinate of $\mathbf{b}_r$ is the farther of ℓ_j and u_j. These bounds are pooled over all selected attack trajectories, so different coordinates may come from different timesteps or trajectories. Thus $\mathbf{b}_r$ is not

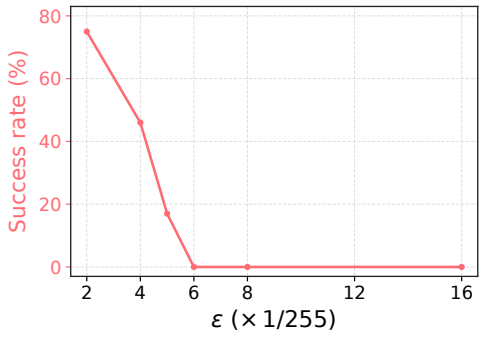

Figure 8: **Effect of perturbation budget.** Increasing ϵ leads to lower task success rates.

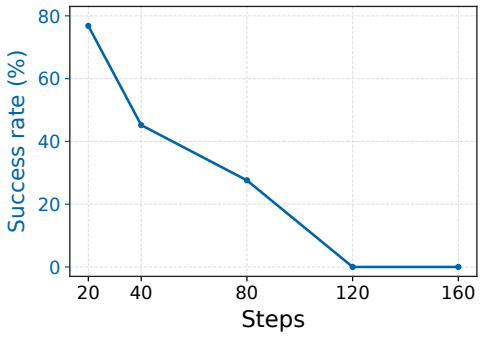

Figure 9: **Task success rate during attack optimization.** Task success decreases as the optimization progresses.

the endpoint's own expert future action and is not a cached clean prediction. This empirical-bound adaptation retains UADA's far-target squared-error construction.

The patch covers 5% of the main-camera image area and replaces pixels within that region. Patch values are constrained to the valid RGB range but are not subject to an additional ℓ_∞ bound. It is optimized on the same attack set $\mathcal{D}_{\text{attack}}$ as WAD, which is disjoint from evaluation episodes.

WAD can perturb the full image of each camera stream, whereas VLA-Patch changes only the selected main-camera region. These are distinct threat models. Their SR difference does not isolate the effect of the objective or establish a general limitation of localized attacks; imposing WAD's ℓ_∞ bound on the patch is not required for this comparison.

Following the optimization protocol of Wang et al. (2025), patch pixels are initialized independently from $\mathcal{U}[0, 1)$ and optimized using AdamW with an initial learning rate of 2×10^{-3}, cosine annealing, 20 warm-up iterations, a batch size of 6, and 2000 outer iterations with 50 inner-loop steps per iteration. Random shear and rotation transformations are applied during optimization, while the learned patch remains untransformed during evaluation.

For LIBERO, we follow the suite-specific patch-placement protocol of Wang et al. (2025). Under its 224×224 image setting, the reported anchor positions are $(120, 160)$, $(30, 150)$, $(15, 158)$, and $(5, 160)$ for Spatial, Object, Goal, and Long, respectively.

B.5 Benchmark Evaluation Protocols

For LIBERO, evaluation follows the official protocol. Each evaluated task is run for 50 episodes from the benchmark's fixed initial states. We use maximum episode lengths of 220, 280, 300, and 520 environment steps for the Spatial, Object, Goal, and Long suites, respectively. The evaluated suites for each model follow the checkpoint coverage described in Appendix B.1.

For RoboTwin, we evaluate seven tasks: `open_microwave`, `move_stapler_pad`, `place_can_basket`, `place_fan`, `open_laptop`, `place_a2b_right`, and `place_shoe`. Each task is evaluated under the benchmark's clean environment configuration using 100 episodes and a maximum episode length of 200 environment steps.

Across all conditions, clean, random-noise, VLA-Patch, Action-Dev, and WAD evaluations use matched task instances and initial states whenever applicable, enabling direct comparison of task success under different visual input modifications.

C Additional Qualitative Results

Fig. 10 presents additional LIBERO and RoboTwin simulation rollouts under clean observations and perturbations that minimize or maximize the VMS-based surrogate, supplementing Sec. 4.2. The clean rollouts show coordinated movements that advance the manipulation task. In contrast,

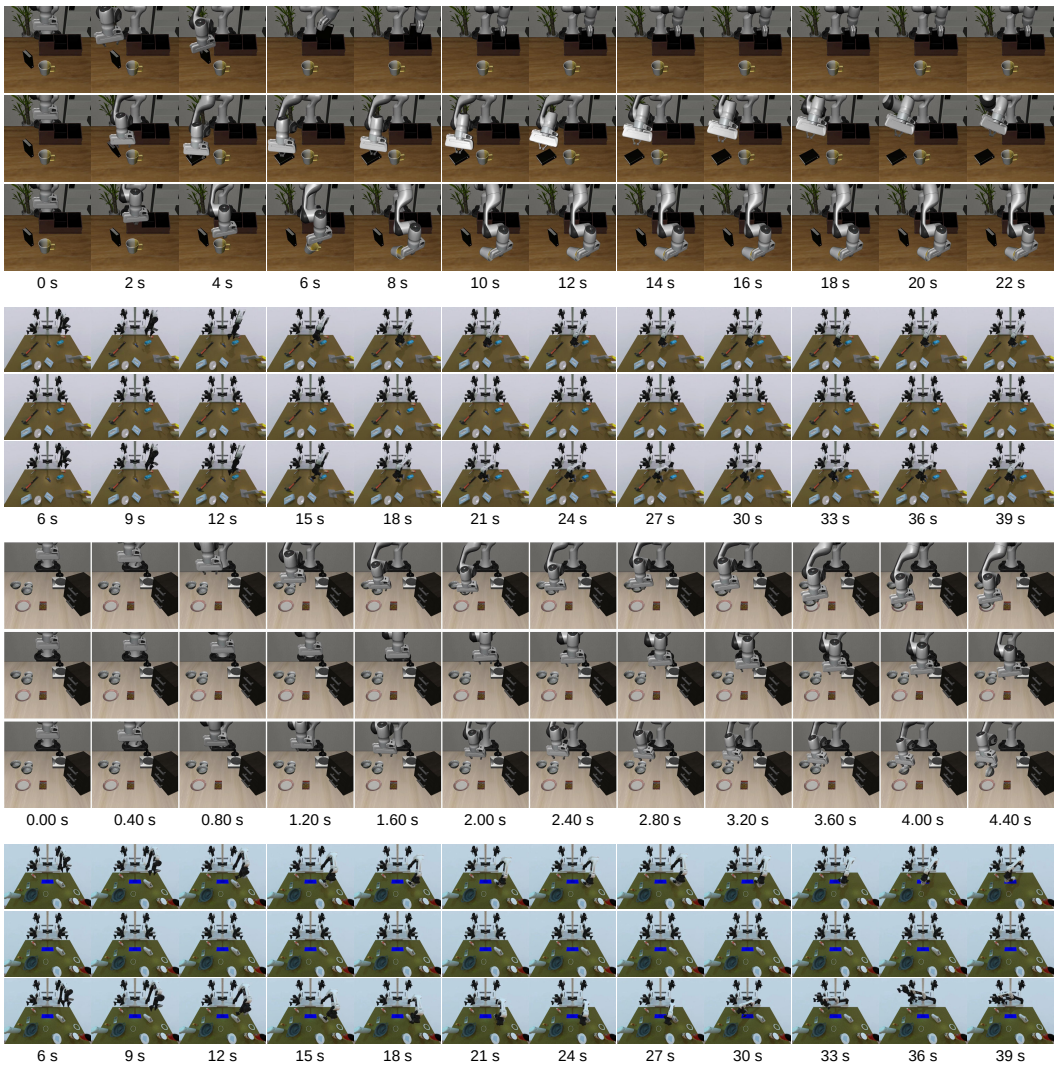

Figure 10: **Additional qualitative results under paired-response suppression and amplification.** We compare simulation executions on LIBERO and RoboTwin under clean observations and perturbations that minimize or maximize the VMS-based surrogate in Eq. 8. Within each group, the top, middle, and bottom rows correspond to Clean, Min-VMS, and Max-VMS, respectively. The clean rollouts complete the tasks, while the Min-VMS rollouts exhibit little or no motion and the Max-VMS rollouts exhibit large-amplitude, unstable movements; both attacked conditions fail.

the Min-VMS rollouts exhibit markedly suppressed motion: the robot remains nearly stationary or makes only small movements, failing to achieve meaningful task progress. The Max-VMS rollouts exhibit large-amplitude movements and unstable execution that prevent successful task completion. These examples illustrate two distinct failure patterns associated with minimizing and maximizing the surrogate, respectively. Videos of these rollouts are available in the supplementary material.

D Additional Experimental Details for VMS

D.1 Reproducibility Across Random Seeds

We assess whether VMS is reproducible across stochastic inference seeds using 50 held-out observations from successful demonstrations and 12 inference seeds. Each observation is paired with a fixed semantic transformation, either a $1\,\mathrm{cm}$ target-object translation or a 2° rotation of an arm-mounted

camera. As in the local-consistency test in Table 1, camera rotation is applied while holding the robot state and the policy's proprioceptive input fixed. The paired policy inputs are fixed across inference seeds. For each seed, the original and transformed observations share the same initial sampling noise, while the noise varies across seeds. We assess the reproducibility of VMS across stochastic inference seeds using FastWAM on LIBERO task 0, with 50 held-out observations from successful demonstrations and 12 inference seeds.

Let V_s denote the VMS measured under seed s. For each observation pair, we compute

$$\mu_V = \frac{1}{S}\sum_{s=1}^{S} V_s, \qquad \sigma_V = \sqrt{\frac{1}{S}\sum_{s=1}^{S}(V_s - \mu_V)^2}, \qquad \mathrm{CV} = \frac{\sigma_V}{\mu_V}, \tag{15}$$

where CV measures the relative variation across seeds. We additionally compare the aggregate response vectors defined in Eq. 2 across inference seeds using cosine similarity to evaluate directional consistency.

As shown in Table 6, VMS exhibits low variability across seeds, with CVs of 3.62% for object translation and 0.88% for camera rotation. The corresponding directional consistency scores are 0.9347 ± 0.0179 and 0.9993 ± 0.0005, respectively. These results show that VMS magnitude and aggregate response direction remain stable across inference seeds.

Table 6: Stability of VMS across 12 inference seeds under target-object translations and arm-mounted camera rotations. Camera rotations are tested with the proprioceptive state held fixed.

Transformation	**VMS**	**CV ↓**	**Direction Consistency ↑**
Object trans. 1 cm	0.4201 ± 0.0152	3.62%	0.9347 ± 0.0179
Camera rot. 2°	0.5200 ± 0.0046	0.88%	0.9993 ± 0.0005

D.2 Relationship between the VMS-based surrogate and task success

We examine the relationship between the VMS-based surrogate and task success using a fixed FastWAM checkpoint on task 0 of LIBERO-Spatial. Visual perturbations are optimized to decrease or increase paired action-prediction differences using the objective in Eq. 8. Each recorded observation retains its associated non-visual conditions, including proprioceptive state, which may differ between consecutive observations.

We obtain endpoint perturbations by solving

$$\min_{\|\boldsymbol{\delta}\|_\infty \le \epsilon} \mathcal{L}_{\mathrm{VMS}}(\boldsymbol{\delta}), \qquad \max_{\|\boldsymbol{\delta}\|_\infty \le \epsilon} \mathcal{L}_{\mathrm{VMS}}(\boldsymbol{\delta}), \tag{16}$$

using projected gradient descent and ascent, respectively. We denote the resulting perturbations by $\boldsymbol{\delta}_{\min}$ and $\boldsymbol{\delta}_{\max}$. The same perturbation is applied to both observations in each pair, while the clean visual difference used for normalization remains fixed. Optimization settings follow Sec. 4.1.

To probe intermediate surrogate values, we interpolate between the endpoint perturbations

$$\boldsymbol{\delta}_\lambda = \lambda\boldsymbol{\delta}_{\min} + (1-\lambda)\boldsymbol{\delta}_{\max}, \qquad \lambda \in [0,1]. \tag{17}$$

Every interpolated perturbation satisfies the same ℓ_∞ budget, although this shared upper bound does not imply equal perturbation magnitudes.

For all selected λ values and the clean condition, we evaluate the surrogate on the same fixed set of paired policy inputs, using clean visual differences for normalization. These measurements characterize paired responses across recorded states and are independent of the states visited during the corresponding rollouts. Task SR is evaluated using matched rollout initial states across conditions.

To assess variation in perturbation strength, we report the ℓ_2 norm of the perturbation actually applied to the RGB input. Let $\mathbf{o}$ denote an observation after geometric preprocessing and before channel normalization, with pixel values in $[0,1]$. The applied perturbation is

$$\boldsymbol{\delta}_{\mathrm{eff},\lambda}(\mathbf{o}) = \mathrm{clip}_{[0,1]}\big(\mathbf{o} + \boldsymbol{\delta}_\lambda\big) - \mathbf{o}. \tag{18}$$

Table 7: Mean episode-level relative variation of different attention blocks.

Model	AV	AA	VV	VA
FastWAM	**36.14%**	33.28%	24.90%	-
LingBotVA	**76.18%**	56.85%	59.92%	-
DreamZero	**74.75%**	34.70%	33.35%	71.02%

The red curve in Fig. 6 reports its ℓ_2 norm, accounting for pixel clipping, whereas $\boldsymbol{\delta}_\lambda$ denotes the additive optimization variable. The clean reference is evaluated separately from the interpolation path and has zero applied perturbation, a surrogate value of 0.722, and SR of 98%. The VMS axis in Fig. 6 reports the surrogate in Eq. 8, which need not vary monotonically with λ. Among perturbed conditions, SR peaks at intermediate scores without a perturbation-norm minimum, weakening a magnitude-only explanation. However, scores and perturbation strength covary, and clean SR remains high outside this peak region. Thus, the non-monotonic association is path-specific and does not establish a universal optimal surrogate range.

E Attention Dynamics Analysis

We analyze the temporal activity of three attention blocks under clean policy inference: visual-to-action $A_{A\leftarrow V}$ (AV), action-to-action $A_{A\leftarrow A}$ (AA), and visual-to-visual $A_{V\leftarrow V}$ (VV). We record the original post-softmax attention probabilities without modifying model inference.

E.1 Relative attention change

For consecutive measurements of an attention block, we compute the relative change as

$$r = \frac{\|P_t - P_{t-1}\|_1}{\max\left(\frac{1}{2}\left(\|P_t\|_1 + \|P_{t-1}\|_1\right), 10^{-12}\right)}, \tag{19}$$

where the ℓ_1 norm is taken over the query and key dimensions. We then average over batch elements, attention heads, measurement positions, temporal transitions, and 10 episodes, and report the final value as a percentage.

E.2 Model-specific measurement protocol

For FastWAM and LingBot-VA, we measure the AV, AA, and VV attention blocks at fixed intervals during policy execution. For LingBot-VA, which maintains a historical KV cache, we restrict comparisons to matched current-token regions while retaining historical keys in the softmax normalization. For DreamZero, each attention matrix contains multiple visual-action (V/A) token groups. We therefore extract the corresponding AV, AA, and VV sub-blocks for each group separately and average their measurements to obtain the layer-level statistics. For all models, measurements are computed over all accessible attention heads and layers, with comparisons restricted to token regions with matched shapes and consistent temporal positions.

E.3 Results and interpretation

As shown in Table 7, $A_{A\leftarrow V}$ consistently exhibits the largest temporal variation across FastWAM, LingBot-VA, and DreamZero. These models cover the three representative WAM paradigms described in Sec. 4.1, suggesting that the high temporal activity of visual-to-action interaction is not specific to a single architecture. Since the models use different measurement settings, we focus on the within-model ranking of attention blocks rather than directly comparing their absolute magnitudes. The consistent AV activity across these paradigms complements the attention-logit scaling experiment in Sec. 3.3 and motivates selecting visual-to-action attention as an attack target. The attack ablations evaluate the resulting task performance.

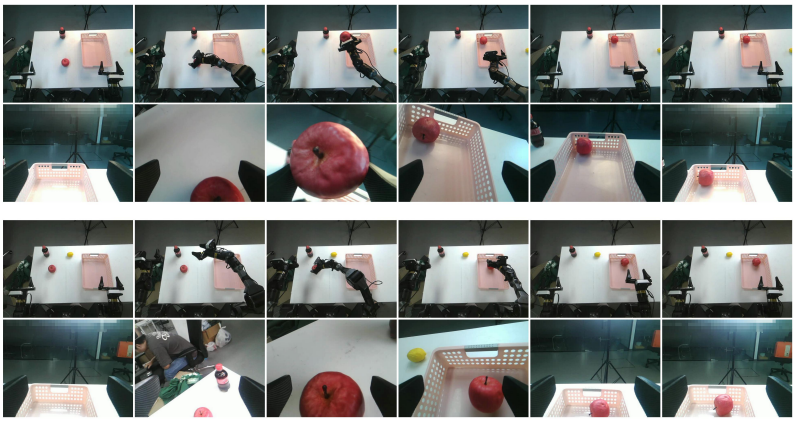

Figure 11: **Examples of real-world training demonstrations.** Two demonstration trajectories collected for the real-world pick-and-place task. Each row shows temporally ordered frames from one trajectory, including the head-camera and wrist-camera views used for policy training.

F Real-World Data Collection, Training, and Deployment

F.1 Robot and Task

We evaluate FastWAM on an ARX dual-arm robot in a real-world pick-and-place task, where the robot follows the instruction `Pick up the apple and put it in the basket`. Both arms are controlled with six joint and one gripper per arm, resulting in a 14-dimensional proprioceptive state and action space. The policy takes three RGB views, cam_head, cam_left, and cam_right, together with the current proprioceptive state, and predicts an action chunk $\mathbf{A}_t \in \mathbb{R}^{32\times 14}$.

F.2 Data Collection

We collect 121 teleoperated demonstrations comprising 50,581 synchronized control steps, corresponding to approximately 28.1 minutes of interaction at 30 Hz, with trajectory lengths ranging from 284 to 1,136 steps and averaging 418 steps (13.9 s). At each step, we record the three RGB frames, a 14-dimensional commanded action, and a 14-dimensional measured robot state, with both vectors ordered as $(l_0, \ldots, l_6, r_0, \ldots, r_6)$. All camera streams, robot records, and timestamps are aligned by frame index, and the data are split at the trajectory level to avoid overlap between training and validation trajectories. Examples of the training demonstrations are shown in Fig. 11.

F.3 Training

The policy is initialized from the RoboTwin FastWAM checkpoint. We fine-tune the video DiT, action DiT, and proprioceptive encoder, while keeping the VAE and text encoder frozen. Training is performed on 8 NVIDIA H20 GPUs with BF16 mixed precision and an effective batch size of 128. We use AdamW with a learning rate of 10^{-5}, weight decay of 0.01, a maximum gradient norm of 1.0, and a cosine learning-rate schedule with a 5% linear warmup. The random seed is fixed to 42.

F.4 Deployment and Perturbation Application

Inference is performed on a single NVIDIA RTX 4090 GPU with 24 GB memory using BF16 precision. The policy predicts a 32×14 action chunk, of which the first 16 actions are executed at 30 Hz before replanning. For WAD, we use an ℓ_∞ perturbation budget of $\epsilon = 8/255$ and apply the perturbation to the RGB observations after standard preprocessing, while keeping all other inputs and deployment settings unchanged.