SAY THE SAME, ACT DIFFERENTLY: Diagnosing and Steering Text–Action Divergence in Reasoning VLAs

Zihao Feng[1] **Qingzhao Zhang**[2] **Chunyu Xia**[1]
Bo Yu[3] **Z. Morley Mao**[1] **Ramesh Govindan**[1]
[1]University of Southern California [2]University of Arizona [3]General Motors
{zihaofen, chunyuxi, morley.mao, ramesh}@usc.edu
qzzhang@arizona.edu bo.3.yu@gm.com

Abstract

Reasoning vision-language-action (VLA) models pair physical control with natural-language rationales that supervisors can read to monitor embodied safety. Prior attacks on VLA reasoning corrupt the rationale itself and leave visible evidence in the text. We show that visual perturbations can instead alter the physical action while keeping the generated rationale word-for-word identical. At the shared text/action interface of Alpamayo for driving and InstructVLA for manipulation, our text-neutral action-subspace (TNAS) diagnostic shows that the locally most action-sensitive direction is nearly orthogonal to the active text-loss gradients, with 1.87% and 5.28% norm overlap. At equal hidden-state perturbation norms across both models, interventions along this text-neutral residual shift actions $4.64\times$ to $47.8\times$ more than text-gradient perturbations, whereas text-gradient perturbations induce $21.2\times$ to $277\times$ more text drift. Steering visual perturbations along this direction reaches the same decoupling from the image. Across 154 obstacle-annotated driving scenes in Alpamayo, TNAS-guided attacks induce a new collision under a word-for-word identical rationale in 52.6% of scenes under full-image perturbations and 35.1% under a localized forward-camera patch, compared with 4.55% for unguided action attacks. Relative to a text-preserving baseline, TNAS guidance raises exact rationale preservation from 83.0% to 89.5% and cuts teacher-forced text drift by 68.4% at a comparable collision rate. In InstructVLA, TNAS guidance cuts text drift by 40.5% and preserves the exact response on 85.0% of examples alongside a 1.73-unit action-chunk shift. An unchanged rationale therefore cannot certify a safe action, so monitors for reasoning VLAs must verify the predicted action channel directly.

1 Introduction

Vision-language-action (VLA) models translate visual observations and language instructions into embodied actions. Reasoning VLAs additionally produce natural-language scene descriptions, rationales, or plans alongside continuous action predictions [1–5]. Unlike latent representations or raw motor commands, this language channel gives human supervisors an explicit, readable account of what the model perceives and intends to do. A supervisor can inspect that account for a missed obstacle, an incorrect target object, or an intention that conflicts with the instruction, potentially before the action is executed. The same idea drives chain-of-thought monitoring for language agents [6, 7], and for embodied models the stakes are even higher because an erroneous physical motion can cause collisions or physical damage. Whether this interface can be trusted for safety hinges on one question: *can the action become unsafe without any change in what the model says?*

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

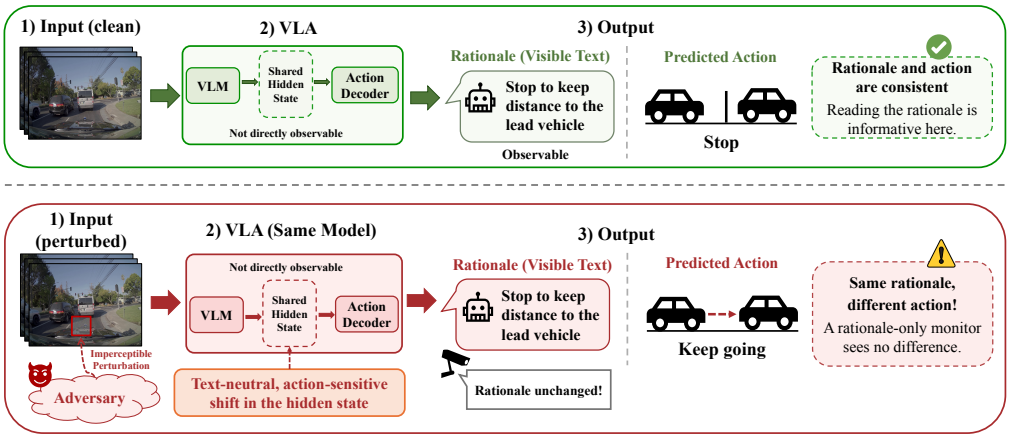


Figure 1: **Rationale-preserving action divergence in reasoning VLAs.** On a clean driving input, the VLM and action decoder read the same unobservable hidden state to produce a matching rationale and stopping action. Adding an imperceptible perturbation to the input shifts that shared hidden state along a text-neutral, action-sensitive direction. The model still outputs "Stop to keep distance to the lead vehicle" word for word, but its predicted action flips from stopping to continuing into the lead vehicle.

Recent attacks on reasoning VLAs redirect behavior by changing the reasoning itself. TRAP steers chain-of-thought and downstream actions toward an adversary's target with adversarial patches [8], while Trinh et al. [9] corrupt the intermediate reasoning trace directly, for example by substituting object names. Both leave evidence in the channel a monitor reads. Because the reasoning text has changed, an incorrect object reference or an instruction-inconsistent plan is visible to a supervisor. In contrast, as illustrated in Figure 1, we study *rationale-preserving action divergence*, where a perturbed input changes the predicted action while leaving the generated rationale word-for-word identical to the clean run. When a perturbed driving input produces a colliding plan under the exact same rationale as the clean input, a monitor restricted to that rationale receives identical evidence in both cases.

This failure mode is rooted in how reasoning and action share a representation. In the reasoning VLAs studied here, action prediction reads continuous hidden states that also produce the text [4, 5]. Because the rationale is a discrete, many-to-one readout of those continuous states, many distinct hidden states can map to the same generated text. Under ordinary inputs the text and action co-vary, which makes rationales informative in typical operation. For a rationale monitor to certify safety, however, any unsafe change in the action must also change the rationale. Prior reasoning-hijacking attacks show that altering the text alters the action, leaving open the converse question: does the shared hidden state contain action-sensitive variation that leaves the text readout unchanged, and can visual inputs reach it?

We answer this question with the TEXT-NEUTRAL ACTION-SUBSPACE (TNAS) diagnostic and a corresponding input-level attack. At the shared text–action interface, TNAS identifies the locally most action-sensitive direction in the hidden state, measures its overlap with the active text-loss gradients, and projects out that text-gradient component to isolate a text-neutral action direction (d_{TNAS}). We first perturb the clean hidden state directly along d_{TNAS} and control directions, under both fixed text-drift budgets and equal perturbation norms, to verify that this local direction shifts the predicted action while leaving the token distribution nearly unchanged. We then align image perturbations with d_{TNAS} during input-level optimization to steer the model toward silent action divergence under paired free generation. Across both Alpamayo for autonomous driving and InstructVLA for robotic manipulation, we find the same geometric decoupling and summarize our main contributions as follows:

Contributions.

- **Rationale-preserving action divergence through visual inputs.** We demonstrate that visual perturbations can substantially alter a reasoning VLA's action while preserving the exact generated rationale. In Alpamayo, TNAS-guided perturbations introduce a geometric collision absent from the clean plan while leaving the rationale word-for-word unchanged in 52.6% of annotated scenes under full-image perturbations and 35.1% under a localized forward-camera patch, and in InstructVLA, they preserve the exact response in 85.0% of examples with a mean action-chunk shift of 1.73 normalized units.

- **A local geometric mechanism for text–action decoupling.** We introduce TNAS and show that the action-sensitive direction at the shared interface overlaps the active text-loss span by only 1.87% of its norm in Alpamayo and 5.28% in InstructVLA. Under equal perturbation norms across both models, perturbing the hidden state along the projected text-neutral direction shifts actions $4.64\times$ to $47.8\times$ more than text-gradient perturbations, whereas text-gradient perturbations induce $21.2\times$ to $277\times$ more teacher-forced KL.
- **Mechanism-guided attacks with stronger text preservation at the same level of action deviation.** Directly combining an action-deviation loss and a text-preservation penalty creates competing gradients that interfere with each other during optimization. Adding TNAS alignment to the attack objective resolves this conflict by steering hidden-state updates along the text-neutral direction: at the strongest guidance weights, exact rationale preservation rises from 83.0% to 89.5% in Alpamayo and from 77.5% to 85.0% in InstructVLA, while teacher-forced text drift falls by 68.4% and 40.5% at an unchanged driving collision rate.

2 TEXT-NEUTRAL ACTION-SUBSPACE Diagnostic

2.1 Setup and probe construction

Models and cohorts. We evaluate Alpamayo, an autonomous driving VLA, on 200 clips and InstructVLA, a robotic manipulation VLA, on 200 Bridge-distribution examples, both fixed as described in Appendix A.1. Alpamayo is our primary safety case study because its predicted trajectory can be scored against logged traffic actors on 154 obstacle-annotated clips. With InstructVLA, we test whether the same geometry holds in a second embodiment that uses a flow-matching action expert and concise task-intention responses.

Shared-state hook and intervention coordinates. For a block of K transformer layers of hidden width d and an active reasoning- or response-token span R, an additive intervention $\delta \in \mathbb{R}^{Kd}$ broadcasts one vector per layer over R. A usable block must expose both a nonzero text basis and a strong action interface in the same state. As detailed in Appendix A.4.1, selecting the qualifying block with the strongest action response gives reasoning-token layers 32–35 in Alpamayo and response-token layers 14–17 in InstructVLA. The pooled state $h \in \mathbb{R}^{Kd}$ concatenates per-layer token means on R and supplies the input-level alignment readout.

Action and text readouts. As detailed in Appendix A.2.1, under the hooked forward pass, both action heads decode with a fixed 10-step Euler flow-matching solver and paired initial noise: Alpamayo integrates acceleration–curvature controls into 2D planar waypoints $\tau_t(\delta) \in \mathbb{R}^2$, and InstructVLA predicts normalized 7-DoF end-effector actions $a_t(\delta) \in \mathbb{R}^7$ (3D position, 3D rotation, and gripper). We measure action change as the mean displacement from the clean output ($\delta = 0$), over $T = 64$ waypoints at 10 Hz (6.4 s, in meters) for Alpamayo and over $H = 16$ chunk steps for InstructVLA:

$$\Delta_{\mathrm{ADE}}(\delta) = \frac{1}{T}\sum_{t=1}^{T} \|\tau_t(\delta) - \tau_t(0)\|_2 , \qquad \Delta_{\mathrm{IVLA}}(\delta) = \frac{1}{H}\sum_{t=1}^{H} \|a_t(\delta) - a_t(0)\|_2 . \tag{1}$$

These *fixed-prefix readouts* hold the clean rationale or response fixed as the text prefix, so the paired clean and perturbed passes differ only through the intervention. As defined in Appendix A.2.2, text drift is the forward teacher-forced KL divergence, from the clean to the perturbed next-token distribution, averaged over R.

Action-Jacobian direction. The action-Jacobian direction d_{act} is a hidden-state direction to which the action output is locally sensitive. We do not take the gradient of Δ_{ADE} or Δ_{IVLA} directly. Both are norms of a displacement that is zero at $\delta = 0$, so neither has a well-defined gradient at the clean point. We instead work with the Jacobian of the action output itself. Let $A_{\mathrm{hook}}(\delta)$ be the flattened action output of the hooked pass, namely the 64 XY waypoints for Alpamayo and the 16×7 action chunk for InstructVLA, with local Jacobian

$$J_A = \left.\frac{\partial A_{\mathrm{hook}}(\delta)}{\partial \delta}\right|_{\delta=0} . \tag{2}$$

As detailed in Appendix A.4, we obtain d_{act} by backpropagating through the action readout to the hooked state and normalizing the resulting row-space direction of J_A. The same direction then goes through text-span residualization and the hidden-state intervention tests.

Active text-loss span. The text side needs losses whose gradients with respect to δ are nonzero at the clean point $\delta = 0$. Teacher-forced KL and clipped hinge losses do not qualify, because the KL gradient vanishes when the clean and perturbed distributions coincide and hinge losses saturate with zero gradient whenever the clean token already clears the margin threshold. We therefore capture local text sensitivity through two complementary losses that remain active at $\delta = 0$, namely target negative log-likelihood (NLL), which tracks the probability mass assigned to the clean token across the vocabulary, and signed target-margin drop, which tracks the logit gap to the strongest competing token that governs top-1 token flips. For token $t \in R$ with clean target y_t and logits z_t, let $p_t = \mathrm{softmax}(z_t)$ and $m_t = z_{t,y_t} - \max_{v \neq y_t} z_{t,v}$. Writing superscripts cln and adv for the clean and perturbed passes, the two losses are

$$\mathcal{L}_{\mathrm{NLL}}(\delta) = \frac{1}{|R|} \sum_{t \in R} -\log p_{t,y_t}^{\mathrm{adv}}, \qquad \mathcal{L}_{\mathrm{margin}}(\delta) = \frac{1}{|R|} \sum_{t \in R} \big(m_t^{\mathrm{cln}} - m_t^{\mathrm{adv}}\big). \tag{3}$$

Gram–Schmidt on their clean-state gradients g_{NLL} and g_{margin} gives an orthonormal basis of the text-loss span $\mathcal{T} = \mathrm{span}\{g_{\mathrm{NLL}}, g_{\mathrm{margin}}\}$, which has rank 2 on every evaluated example.

TNAS residual. With $P_{\mathcal{T}}$ the orthogonal projector onto $\mathcal{T}$, the text-neutral action-subspace (TNAS) direction is

$$d_{\mathrm{TNAS}} = \mathrm{normalize}\big((I - P_{\mathcal{T}}) d_{\mathrm{act}}\big). \tag{4}$$

By construction, d_{TNAS} is orthogonal to both text-loss gradients at $\delta = 0$, so a small step along it leaves the NLL and margin of the clean rationale unchanged to first order while still moving the action.

2.2 Clean-state geometry

Probe 1. How much of the action does the text carry?
Rationale-only action prediction. To evaluate the consistency between the rationale and the action under ordinary inputs, we use a frozen Qwen3-14B model [10], configured as described in Appendix A.5, to predict from the clean rationale alone whether Alpamayo's planned trajectory comes to rest within the 6.4 s horizon. Across the 1,181 clips, where 24.5% (289/1,181) of planned trajectories actually come to rest, the reader's prediction matches the trajectory outcome on 84.3% of clips. Clips predicted to stop come to rest 63.6% of the time, compared with 5.76% for clips predicted to continue. Even so, the text does not pin down the action. The 1,181 clips share only 258 distinct rationales, 36.9% carry a rationale that also appears with the opposite stopping outcome, and the most frequent rationale ("Keep distance to the lead vehicle since it is directly ahead in our lane") alone covers 245 clips with 10.1 m to 208 m of forward displacement.
Hidden-state gradient projection. The hidden-state geometry locates this gap at the shared interface. To quantify how much of the action gradient lies in the text span, we project the action direction d_{act} onto $\mathcal{T}$ and divide by the norm of d_{act}, giving the projection-norm ratio $\rho = \|P_{\mathcal{T}} d_{\mathrm{act}}\| / \|d_{\mathrm{act}}\|$. Across the 200 Alpamayo clips, ρ averages 1.87% (versus $\rho = 1$ for text-gradient controls), roughly twice the 0.978% mean of isotropic random unit vectors reported in Appendix A.6, and across the 200 InstructVLA examples, ρ averages 5.28%. Because the text-span projection $P_{\mathcal{T}} d_{\mathrm{act}}$ and its orthogonal complement $(I - P_{\mathcal{T}}) d_{\mathrm{act}}$ are perpendicular, removing $\mathcal{T}$ leaves $\sqrt{1 - \rho^2} > 99.8\%$ of d_{act}'s norm intact on average in both models, meaning that almost the entire action-sensitive direction is orthogonal to the text span.

Takeaway 1. The rationale carries action information, but the text registers only a small part of what moves the action. Read alone, Alpamayo's rationale predicts whether the plan stops on 84.3% of clips, yet 36.9% of clips share their rationale with the opposite outcome. At the shared interface, the projection-norm ratio ρ of d_{act} onto the active text span averages 1.87% in Alpamayo and 5.28% in InstructVLA, so projecting out the text span retains on average more than 99.8% of the action direction's norm in both models.

2.3 Hidden-state intervention results

Probe 1 measures first-order gradients at $\delta = 0$. To test whether this local separation holds when finite hidden-state perturbations pass through the nonlinear language and action heads, Probe 2 and Probe 3 compare d_{TNAS} and the unprojected action direction d_{act} against three non-action control directions, namely the text-loss gradients (g_{NLL}, g_{margin}) and random unit vectors (u_{rand}), on the

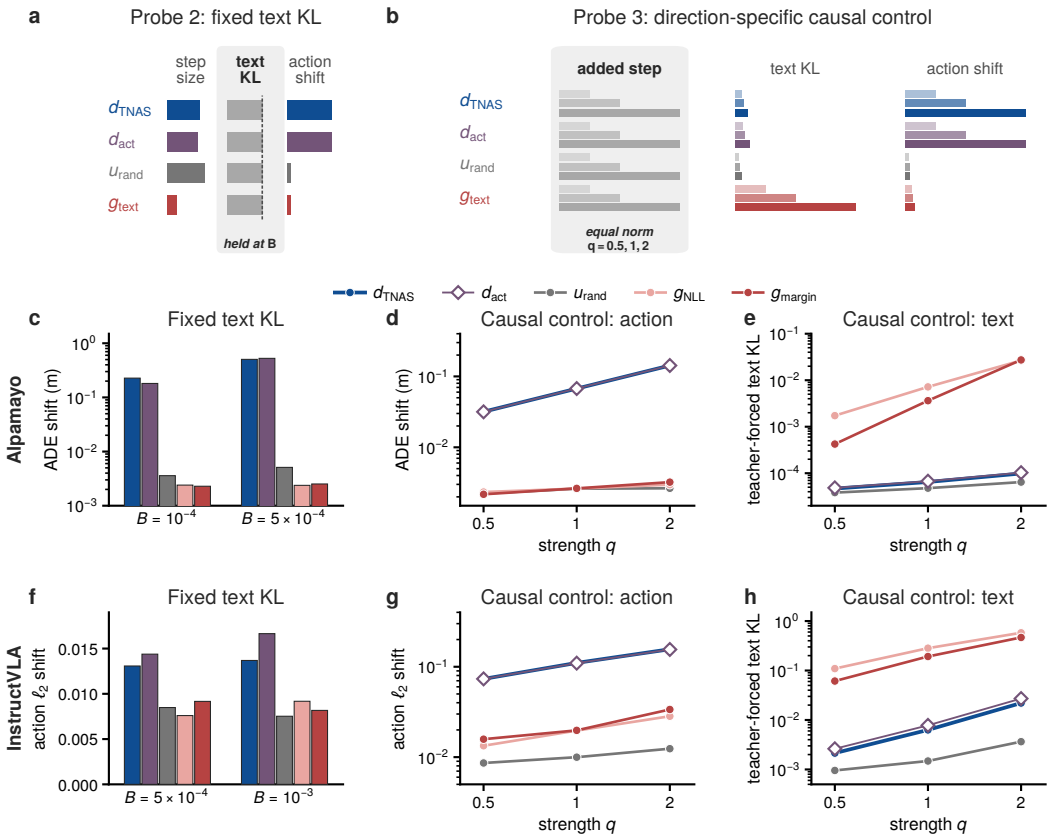


Figure 2: **Hidden-state intervention probes.** *Top:* schematic of the two protocols (g_{text} denotes g_{NLL} and g_{margin}). (a) Probe 2 adjusts each direction's step size so that all directions incur the same text-KL budget B (shaded) and compares the resulting action shift. (b) Probe 3 adds every direction at the same per-example norm $|q|S_{\text{clip}}$ for $q \in \{0.5, 1, 2\}$ (shaded, light to dark) and compares the resulting text KL and action shift. *Middle and bottom:* cohort means on 200 Alpamayo clips and 200 InstructVLA examples, with full values tabulated in Appendix A.7. (c,f) Action shift at fixed teacher-forced KL budgets B. (d,g) Action shift and (e,h) teacher-forced text KL across equal-norm strengths q, where d_{act} (hollow diamonds) nearly overlaps d_{TNAS}.

same 200 Alpamayo clips and 200 InstructVLA examples. Figure 2 summarizes both probes, and Appendices A.4 and A.7 provide the implementation details and full tables. As shown in Figure 2, Probe 2 compares these directions under a fixed text-drift budget in panel a, and Probe 3 compares them under a fixed hidden-state perturbation norm in panel b.

Probe 2. Fixed text KL. To test how much action change a hidden-state perturbation can induce before noticeably altering the rationale, we hold teacher-forced text KL fixed across directions and compare the resulting action shifts. As illustrated in Figure 2a,c,f, for each direction, we select the step size on the perturbation grid whose teacher-forced KL matches a target budget B.
Alpamayo. Perturbing along d_{TNAS} shifts the planned trajectory by 0.227 m at $B = 10^{-4}$ and 0.504 m at $B = 5 \times 10^{-4}$, matching d_{act} and exceeding the controls (≤ 5.10 mm) by $63.3\times$ to $212\times$.
InstructVLA. Perturbing along d_{TNAS} and d_{act} likewise produces the largest action shifts at both budgets, reaching 1.37×10^{-2} and 1.66×10^{-2} at $B = 10^{-3}$ versus at most 9.18×10^{-3} for the controls.

Probe 3. Direction-specific causal control. In Probe 2, matching a fixed text-KL budget allows directions that barely perturb the text to take larger hidden-state steps than text-sensitive directions, as Figure 2a illustrates. To verify that d_{TNAS} is intrinsically more action-selective per unit norm rather than only benefiting from a larger step size, Probe 3 fixes the hidden-state perturbation magnitude across all directions, as depicted in Figure 2b. Starting from the clean state, Probe 3 adds each direction with the same per-example norm $|q|S_{\text{clip}}$, where $S_{\text{clip}} = |\langle h_{\text{clean}}, d_{\text{TNAS}} \rangle|$ and $q \in \{0.5, 1, 2\}$, as

detailed in Appendix A.8. Figure 2d,e,g,h then shows how the action and text channels respond. Across both models, d_{TNAS} and the text-gradient controls show opposite selectivity, while d_{act} lies on top of d_{TNAS} across all q.

Alpamayo. The trajectory shift along d_{TNAS} grows with q to 0.142 m at $q = 2$, exceeding the text-gradient controls by $44.0\times$ to $47.8\times$ and the random control by $53.9\times$. Conversely, at $q = 2$, d_{TNAS} induces only 9.87×10^{-5} teacher-forced KL, whereas the text-gradient controls induce $269\times$ to $277\times$ higher KL.

InstructVLA. At $q = 2$, d_{TNAS} shifts the action by 0.156, exceeding the text-gradient controls by $4.64\times$ to $5.51\times$ and the random control by $12.6\times$, while keeping teacher-forced KL at 2.20×10^{-2} compared with $21.2\times$ to $26.3\times$ higher KL for the text-gradient controls.

Takeaway 2. Hidden-state interventions along d_{TNAS} shift the action while barely moving the text. Under a fixed text-KL budget, d_{TNAS} shifts Alpamayo's trajectory $63.3\times$ to $212\times$ more than the controls. Under an equal perturbation norm across both models, d_{TNAS} shifts the action $4.64\times$ to $47.8\times$ more than the text-gradient controls and $12.6\times$ to $53.9\times$ more than the random control, while the text-gradient controls induce $21.2\times$ to $277\times$ more teacher-forced KL.

3 TNAS-Guided Visual Perturbations

Section 2 showed that direct hidden-state interventions along d_{TNAS} shift the predicted action while leaving the next-token distribution nearly unchanged. We now test whether an adversary modifying only the visual input can excite this same direction end-to-end in Alpamayo and InstructVLA, resolving the optimization conflict between altering the physical action and preserving the generated text.

3.1 Attack setup

Threat model. We consider white-box digital perturbations applied to the normalized, processed visual input tensor immediately before the visual encoder, with gradients backpropagated end-to-end through the vision encoder, language backbone, and action head. In Alpamayo, full-image attacks perturb all 16 camera-frame inputs across the four cameras and four temporal frames. In InstructVLA, both full-image and localized attacks perturb the VLM/SigLIP visual stream. Section 3.3 further restricts the attacker to a spatially localized patch in each domain to mimic real-world scenarios where perturbations are attached to local objects or surfaces in the scene.

Optimization. Every attack runs 200 steps of projected gradient descent (PGD) [11–14] on its maximization objective $\mathcal{L}$,

$$\delta^{(k+1)} = \Pi_{\mathcal{S}}\left[\delta^{(k)} + \alpha \operatorname{sign}\Big(\nabla_{\delta}\mathcal{L}(\delta^{(k)})\Big)\right], \tag{5}$$

where $\Pi_{\mathcal{S}}$ projects onto the ℓ_∞ ball of radius ε and the allowed spatial mask. Full-image attacks use $\varepsilon = 0.05$ and step size $\alpha = 5 \times 10^{-4}$ in normalized pixel coordinates, whereas localized patches use a larger local budget ($\varepsilon = 0.5$, $\alpha = 5 \times 10^{-3}$) to compensate for their restricted spatial support. Further implementation details on differentiable action rollouts are provided in Appendix A.2.

Evaluation metrics. Our evaluation covers both controlled fixed-prefix readouts and end-to-end free generation, along with open-loop safety checks in Alpamayo:

- **Fixed-prefix readouts.** Under the fixed clean text prefix, we report the Section 2.1 readouts: action shift (Δ_{ADE} or Δ_{IVLA}), teacher-forced KL ($\mathrm{KL}_{\mathrm{tf}}$), and hidden-state alignment $\cos(\Delta h, d_{\mathrm{TNAS}})$.
- **Free-generation exact match.** As detailed in Appendices A.1–A.2.3, under free generation, the model autoregressively generates its own text and decodes the corresponding action, and *Exact* reports the fraction of examples whose generated rationale or response matches the clean one word for word.
- **Silent collision rate.** To evaluate whether trajectory divergence creates concrete driving hazards, we score Alpamayo's free-generated 6.4 s (64-waypoint) plans against time-aligned logged traffic actors on the 154 clips with obstacle annotations [15, 16], following the protocol detailed in Appendix A.3. At each 0.1 s planning step, we project the ego vehicle and logged actors as oriented bounding boxes in bird's-eye view, inferring ego heading from consecutive waypoints, and test for overlap using the separating-axis theorem. A clip has an open-loop collision if the

Table 1: **Alpamayo input attacks.** Fixed-prefix and free-generation columns use the canonical 200 clips. The open-loop column uses the 154 of them with obstacle annotations, as described in Section 3.1.

Attack	Fixed-prefix			Free gen.	Open loop
	ADE (m) ↑	$\mathrm{KL}_{\mathrm{tf}}$ ↓	TNAS cos	Exact (%) ↑	Silent Coll. (%) ↑
Action-only PGD	70.1	2.78×10^{-1}	4.40×10^{-4}	9.5	4.55
Text-preserving PGD	69.2	3.26×10^{-1}	8.00×10^{-4}	83.0	48.7
TNAS Full image					
$\lambda_{\cos} = 5$	67.4	3.32×10^{-1}	1.64×10^{-2}	80.5	46.8
$\lambda_{\cos} = 10$	67.8	2.99×10^{-1}	3.93×10^{-2}	81.0	47.4
$\lambda_{\cos} = 15$	65.5	2.46×10^{-1}	7.57×10^{-2}	84.0	45.5
$\lambda_{\cos} = 20$	63.2	1.89×10^{-1}	1.33×10^{-1}	88.0	45.5
$\lambda_{\cos} = 30$	53.0	1.03×10^{-1}	2.46×10^{-1}	89.5	52.6
TNAS Patch					
$\lambda_{\cos} = 5$	38.7	1.39×10^{-1}	1.55×10^{-2}	70.5	33.8
$\lambda_{\cos} = 10$	39.8	1.24×10^{-1}	4.24×10^{-2}	70.5	35.1
$\lambda_{\cos} = 15$	37.2	1.07×10^{-1}	6.74×10^{-2}	71.0	31.8
$\lambda_{\cos} = 20$	34.8	9.00×10^{-2}	1.04×10^{-1}	76.0	34.4
$\lambda_{\cos} = 30$	30.0	5.70×10^{-2}	1.62×10^{-1}	80.5	34.4

ego box intersects any actor box at least once over the 6.4 s horizon, and the *silent collision rate* reports the fraction of the 154 clips where a collision-free clean plan is perturbed into a collision while the free-generated rationale remains word-for-word identical to the clean rationale.

Baselines. We compare against two baseline objectives that represent the intuitive output-level approaches to inducing action deviation alone and to inducing action deviation while preserving the generated text.

Action-only PGD only maximizes an action-displacement loss $\mathcal{L}_{\mathrm{act}}(\delta_{\mathrm{img}})$ between the perturbed and clean action predictions over the T-step horizon:

$$\mathcal{L}_{\mathrm{act}}(\delta_{\mathrm{img}}) = \frac{1}{T} \sum_{t=1}^{T} D_{\mathrm{act}}\big(a_t(\delta_{\mathrm{img}}),\, a_t(0)\big). \tag{6}$$

Following Equation 1, the per-step discrepancy D_{act} is the planar waypoint displacement $\|\tau_t(\delta_{\mathrm{img}}) - \tau_t(0)\|_2$ for Alpamayo and the mean squared coordinate difference across the 7-DoF action vector for InstructVLA.

Text-preserving PGD, detailed in Appendix A.2, subtracts an explicit output-level penalty on the two active text losses defined in Section 2.1, namely the clean-target NLL increase $\Delta\mathcal{L}_{\mathrm{NLL}}$ and the target-versus-runner-up margin drop Δm, following joint action–text formulations for reasoning VLAs [8]:

$$\mathcal{L}_{\mathrm{act+txt}}(\delta_{\mathrm{img}}) = \mathcal{L}_{\mathrm{act}}(\delta_{\mathrm{img}}) - \lambda_{\mathrm{txt}}\,[\Delta\mathcal{L}_{\mathrm{NLL}} + \Delta m] \tag{7}$$

3.2 TNAS-guided full-image attack

Output-level action and text objectives compete. In Table 1, Action-only PGD in Alpamayo shifts the planned trajectory by 70.1 m ADE, but it also rewrites the driving rationale on 90.5% of clips, leaving only 4.55% silent collisions. Adding the output-level text penalty in Text-preserving PGD keeps the clean rationale intact on 83.0% of clips and raises silent collisions to 48.7%, yet the two objectives work against each other during optimization. In Equation 7, both terms update the same input pixels: the action loss maximizes trajectory displacement, while the text penalty constrains only the clean target tokens' NLL and top-1 margin. Holding those target tokens in place against unguided action updates pushes vocabulary-wide teacher-forced KL from 2.78×10^{-1} under Action-only PGD to 3.26×10^{-1}, which lowers the chance that stochastic decoding reproduces the clean rationale.

Resolving the conflict with TNAS guidance. To resolve this conflict at its geometric source, **TNAS-guided PGD** augments the text-preserving objective with a term that rewards cosine alignment

Table 2: **InstructVLA input attacks.** Fixed-prefix and free-generation columns use the canonical 200 examples.

Attack	Fixed-prefix			Free gen.
	Action shift (norm.) ↑	$\mathrm{KL}_{\mathrm{tf}}$ ↓	TNAS cos	Exact (%) ↑
Action-only PGD	2.04	4.66×10^{-2}	1.20×10^{-3}	72.5
Text-preserving PGD	1.97	3.46×10^{-2}	2.90×10^{-3}	77.5
TNAS Full image				
$\lambda_{\cos} = 0.10$	1.85	2.67×10^{-2}	1.22×10^{-1}	81.0
$\lambda_{\cos} = 0.15$	1.80	2.35×10^{-2}	1.66×10^{-1}	81.5
$\lambda_{\cos} = 0.20$	1.73	2.06×10^{-2}	2.13×10^{-1}	85.0
TNAS Patch				
$\lambda_{\cos} = 0.20$	1.32	2.80×10^{-2}	1.13×10^{-1}	77.5

between the input-induced hidden-state shift $\Delta h(\delta_{\mathrm{img}})$ and each example's TNAS direction d_{TNAS}:

$$\mathcal{L}_{\mathrm{TNAS}}(\delta_{\mathrm{img}}) = \mathcal{L}_{\mathrm{act+txt}}(\delta_{\mathrm{img}}) + \lambda_{\cos} \cos\big(\Delta h(\delta_{\mathrm{img}}), d_{\mathrm{TNAS}}\big). \tag{8}$$

As shown in Section 2, shifting the hidden state along d_{TNAS} drives action displacement while leaving the two text losses in Equation 7 unchanged to first order. Maximizing cosine similarity in Equation 8 directly steers the direction of $\Delta h(\delta_{\mathrm{img}})$ toward d_{TNAS}. During optimization, this allows PGD updates to increase action displacement with minimal disturbance to the text logits and reduces the conflict between the action and text terms in $\mathcal{L}_{\mathrm{act+txt}}$.

TNAS guidance quiets the text channel. In Table 1, increasing $\lambda_{\cos}$ from 5 to 30 raises cosine alignment between Δh and d_{TNAS} at every step from 1.64×10^{-2} to 2.46×10^{-1}, compared with 8.00×10^{-4} under Text-preserving PGD. Even though the cosine term never accesses token logits or text losses, steering the hidden state toward d_{TNAS} pulls teacher-forced KL down monotonically across the sweep to 1.03×10^{-1} at $\lambda_{\cos} = 30$, a 68.4% reduction relative to Text-preserving PGD. Because lower distributional drift keeps the perturbed next-token distribution closer to the clean one during stochastic decoding, free-generation exact match also rises monotonically across the sweep and reaches 89.5% at $\lambda_{\cos} = 30$, up from 83.0% without TNAS guidance.

Steering toward d_{TNAS} concentrates displacement into silent collisions. Under a fixed pixel budget, weighting the objective toward d_{TNAS} trades redundant fixed-prefix displacement, from 69.2 m under Text-preserving PGD to 53.0 m at $\lambda_{\cos} = 30$, for stronger TNAS alignment without reducing collisions. On the 154 obstacle-annotated scenes, the free-generated plans continue to collide on 61.7% of clips at $\lambda_{\cos} = 30$ versus 61.0% under Text-preserving PGD, so the higher exact rationale preservation translates directly into silent collisions. The final step of the sweep brings the largest gains in text stability. Between $\lambda_{\cos} = 20$ and $\lambda_{\cos} = 30$, TNAS alignment nearly doubles and teacher-forced KL falls the most, while silent collisions reach 52.6%, the highest in the sweep. TNAS guidance therefore buys text stability without giving up silent collisions.

Replication in InstructVLA. In Table 2, the same geometric guidance transfers to robotic manipulation in InstructVLA, where perturbations modify only the VLM visual stream while a single task-intention sentence averaging 9.60 words, such as "I will move the cup to the sink," covers an entire 16-step 7-DoF action chunk. Across the $\lambda_{\cos}$ sweep from 0.10 to 0.20, cosine alignment between Δh and d_{TNAS} rises at every step from 1.22×10^{-1} to 2.13×10^{-1}, whereas Action-only PGD at 1.20×10^{-3} and Text-preserving PGD at 2.90×10^{-3} leave the hidden-state shift unaligned with d_{TNAS}. Even the smallest guided weight $\lambda_{\cos} = 0.10$ already improves over Text-preserving PGD on every text metric, and steering further toward d_{TNAS} pulls teacher-forced KL down monotonically from 2.67×10^{-2} to 2.06×10^{-2} at $\lambda_{\cos} = 0.20$, a 40.5% reduction from 3.46×10^{-2} under Text-preserving PGD and 55.8% below 4.66×10^{-2} under Action-only PGD. This lower distributional drift raises free-generation exact match monotonically across the sweep from 81.0% to 85.0% at $\lambda_{\cos} = 0.20$, up from 77.5% under Text-preserving PGD. Even though this coarser many-to-one intention bottleneck already leaves 72.5% of responses unchanged under Action-only PGD without any text penalty, steering along d_{TNAS} still eliminates one-third of the response changes that remain under Text-preserving PGD. Under the fixed pixel budget, weighting the objective toward d_{TNAS} trades only a small part of

the fixed-prefix action shift, from 2.04 under Action-only PGD and 1.97 under Text-preserving PGD to 1.73 at $\lambda_{\cos} = 0.20$, so it retains 87.5% of Text-preserving PGD's action displacement alongside this stronger text stability. As in Alpamayo, the final step from $\lambda_{\cos} = 0.15$ to $\lambda_{\cos} = 0.20$ brings the largest exact-match gain within the guided sweep, jumping from 81.5% to 85.0% as TNAS alignment climbs from 1.66×10^{-1} to 2.13×10^{-1} and teacher-forced KL drops from 2.35×10^{-2} to 2.06×10^{-2}.

Takeaway 3. TNAS guidance resolves the action–text optimization conflict, suppressing text drift while increasing silent collisions. Output-level text penalties compete with the action gradient and leave hidden-state updates unaligned with d_{TNAS}. Steering along d_{TNAS} lowers teacher-forced KL by 68.4% in Alpamayo and 40.5% in InstructVLA, raises exact rationale preservation from 83.0% to 89.5% and from 77.5% to 85.0%, and produces silent collisions on 52.6% of annotated driving scenes alongside a 1.73-unit manipulation action shift.

3.3 Localized patch attack

Our full-image attacks assume a digital attacker who can add a perturbation to the preprocessed image tensor fed to the vision encoder, which in practice requires access to the model's software pipeline. Moving toward localized attacks on objects in the scene, such as a patch on the back of a front vehicle, requires the perturbation to operate within a restricted spatial region while remaining consistent across views and time. To test whether an attacker can still steer the shared representation along d_{TNAS} under these constraints, we confine the perturbation to a localized patch and retain the TNAS-guided objective.

Shared forward-camera patch in Alpamayo. In Alpamayo, we optimize a single clip-specific patch on the road surface ahead of the vehicle and tie it across the two forward-facing cameras and all four timestamps, covering 13.1% of visual tokens. The front-telephoto crop receives the base pattern, the front-wide crop receives a bilinearly resized version on the corresponding road region, and the two cross-view cameras remain clean. In Table 1, increasing $\lambda_{\cos}$ from 5 to 30 raises cosine alignment between Δh and d_{TNAS} at every step from 1.55×10^{-2} to 1.62×10^{-1} and pulls teacher-forced KL down monotonically from 1.39×10^{-1} to 5.70×10^{-2}, a 59.0% reduction that lifts free-generation exact match from 70.5% to 80.5%. With the surrounding road and cross-view cameras left untouched, fixed-prefix trajectory displacement reaches 38.7 m ADE at $\lambda_{\cos} = 5$ and 39.8 m at $\lambda_{\cos} = 10$, and weighting the objective toward $\lambda_{\cos} = 30$ trades part of this shift to 30.0 m for stronger text stability while keeping silent collisions between 31.8% and 35.1% of obstacle-annotated scenes.

VLM-branch patch in InstructVLA. In InstructVLA, we confine SigLIP updates to a 0.4×0.4 lower-center rectangle covering 16% of the image area. In Table 2, TNAS-guided PGD at $\lambda_{\cos} = 0.20$ steers the hidden-state shift from this patch to 1.13×10^{-1} cosine alignment with d_{TNAS} and lowers teacher-forced KL to 2.80×10^{-2}, 19.1% below the 3.46×10^{-2} of full-image Text-preserving PGD. Free-generation exact match reaches 77.5%, matching full-image Text-preserving PGD and exceeding the 72.5% of full-image Action-only PGD. With the remaining 84% of the image left untouched, the patch shifts the predicted action chunk by 1.32 and retains 76.4% of the 1.73 displacement from full-image TNAS-guided PGD at $\lambda_{\cos} = 0.20$.

Takeaway 4. TNAS-guided action divergence persists under localized patch constraints. In Alpamayo, a forward-camera patch on 13.1% of visual tokens cuts teacher-forced KL by 59.0% across the $\lambda_{\cos}$ sweep, preserves up to 80.5% of rationales word for word, and induces silent collisions on up to 35.1% of annotated scenes. In InstructVLA, a patch on 16% of the image shifts the action chunk by 1.32 units while matching the 77.5% exact preservation of full-image Text-preserving PGD with 19.1% lower teacher-forced KL.

4 Related Work

Reasoning vision-language-action models. VLA policies connect visual and language representations to embodied control [17–20], with autonomous driving models and benchmarks extending this paradigm to multi-camera road scenes [21–27]. Reasoning-conditioned VLAs expose natural-language scene descriptions, plans, or chain-of-thought traces alongside continuous action prediction [1, 2, 28, 3]. We study Alpamayo-1.5 [4] and InstructVLA [5], two representative embodiments

that pair autoregressive VLM text generation with continuous diffusion or flow-matching action heads reading from a shared hidden state.

Monitoring model reasoning. Chain-of-thought monitoring treats verbalized reasoning as a safety signal and asks when that signal stops reflecting behavior [6, 7]. In language models, faithfulness studies distinguish plausible post-hoc explanations from the causal drivers of an answer [29–32]. In embodied VLAs, prior work addresses unperturbed mismatches between textual plans and actions at the output level through RL consistency rewards [4] or runtime action verification [33]. Concurrent with our work, Foutter et al. [34] observe that Alpamayo-1.5 can state that it will yield while steering without decelerating, and post-train driving planners with a learned critic that rewards semantic consistency among observations, reasoning steps, and trajectories. We instead diagnose why text and action decouple at the shared hidden state, where TNAS shows that the locally most action-sensitive direction is nearly orthogonal to the active text-loss gradients, and steer visual inputs along this subspace to alter the physical action while keeping the generated rationale word-for-word identical. The TNAS geometry justifies pairing rationale monitors with execution-channel monitors [35].

Adversarial attacks on driving and embodied policies. Adversarial examples were first studied as imperceptible ℓ_p-bounded perturbations of the full digital image [36, 11, 37, 12]. Later work showed that such perturbations survive printing, camera capture, and viewpoint changes [38, 39], and that a localized patch such as a printed sticker or accessory can fool image classifiers and face recognition systems [40–43]. In autonomous driving, physical patches appear as dirt-like road patterns that steer automated lane centering out of its lane [44], roadside billboards that mislead steering models [45], and patches on the rear of a lead vehicle that corrupt monocular depth estimation [46]. Attacks on driving stacks also target trajectory prediction [13] and collaborative perception [14]. On VLAs, a small patch in the camera view can derail robotic manipulation [47] or hijack chain-of-thought reasoning [8], and other attacks corrupt the reasoning trace directly [9] or implant backdoors [48–50]. Reasoning-hijacking attacks visibly alter the text, and unconstrained action attacks corrupt it as a byproduct. Our TNAS-guided attack instead holds the generated rationale fixed while steering the action, under both full-image perturbations and a localized patch.

Representation steering and subspace projection. Representation engineering and activation steering change model behavior by adding a direction to intermediate hidden states at inference time [51, 52]. Linear concept erasure projects hidden states so that a linear readout can no longer recover a protected attribute [53, 54]. TNAS applies this projection to the steering direction itself at the shared text–action interface. Removing the active text-loss span from the locally most action-sensitive direction leaves a residual that moves the action readout while keeping the text readout unchanged to first order. Activation steering writes such directions into the hidden state, whereas our attack reaches the TNAS residual by perturbing only the visual input.

5 Conclusion

We studied rationale-preserving action divergence in reasoning VLAs, where a perturbed visual input alters the physical action while leaving the generated rationale word for word unchanged. To diagnose this vulnerability at the shared text–action interface, we introduced the text-neutral action-subspace (TNAS) probe, which projects active text-loss gradients out of the locally most action-sensitive hidden-state direction to isolate a text-neutral action residual. Across Alpamayo for autonomous driving and InstructVLA for robotic manipulation, the action-sensitive direction overlaps the active text span by only 1.87% and 5.28% of its norm, and equal-norm interventions along the TNAS residual shift actions $4.64\times$ to $47.8\times$ more than text-gradient perturbations while text-gradient perturbations induce $21.2\times$ to $277\times$ more text drift. Steering visual perturbations along this residual resolves the optimization conflict between action displacement and text preservation and cuts teacher-forced text drift by 68.4% in Alpamayo and 40.5% in InstructVLA relative to text-preserving baselines. Under free generation, TNAS-guided attacks preserve the rationale word for word while inducing new collisions on 52.6% of annotated driving scenes under full-image perturbations and 35.1% under a localized forward-camera patch, and preserve 85.0% of manipulation responses alongside a 1.73-unit action-chunk shift. Because an unchanged rationale cannot certify a safe action, embodied safety monitors must verify the predicted action channel directly alongside the text.

Acknowledgments and Disclosure of Funding

We thank the area chair and the anonymous reviewers for their valuable feedback and constructive suggestions. We also specifically thank Chufan Shi for his valuable help and insightful discussions throughout this work. This material is based upon work supported by the National Science Foundation under Grant No. 2333489. The authors declare no competing interests.

References

[1] Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

[2] Bo Jiang, Shaoyu Chen, Bencheng Liao, Xingyu Zhang, Wei Yin, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. Senna: Bridging large vision-language models and end-to-end autonomous driving. *arXiv preprint arXiv:2410.22313*, 2024.

[3] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

[4] Yan Wang, Wenjie Luo, Junjie Bai, Yulong Cao, Tong Che, Ke Chen, Yuxiao Chen, Jenna Diamond, Yifan Ding, Wenhao Ding, et al. Alpamayo-r1: Bridging reasoning and action prediction for generalizable autonomous driving in the long tail. *arXiv preprint arXiv:2511.00088*, 2025.

[5] Shuai Yang, Hao Li, Bin Wang, Yilun Chen, Yang Tian, Tai Wang, Hanqing Wang, Feng Zhao, Yiyi Liao, and Jiangmiao Pang. Instructvla: Vision-language-action instruction tuning from understanding to manipulation. *arXiv preprint arXiv:2507.17520*, 2025.

[6] Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y. Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*, 2025.

[7] Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, Mark Chen, Alan Cooney, Allan Dafoe, Anca Dragan, et al. Chain of thought monitorability: A new and fragile opportunity for AI safety. *arXiv preprint arXiv:2507.11473*, 2025.

[8] Zhengxian Huang, Wenjun Zhu, Haoxuan Qiu, Xiaoyu Ji, and Wenyuan Xu. Trap: Hijacking vla cot-reasoning via adversarial patches. *arXiv preprint arXiv:2603.23117*, 2026.

[9] Tuan Duong Trinh, Naveed Akhtar, and Basim Azam. Altered thoughts, altered actions: Probing chain-of-thought vulnerabilities in vla robotic manipulation. *arXiv preprint arXiv:2603.12717*, 2026.

[10] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[12] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[13] Qingzhao Zhang, Shengtuo Hu, Jiachen Sun, Qi Alfred Chen, and Z Morley Mao. On adversarial robustness of trajectory prediction for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15159–15168, 2022.

[14] Qingzhao Zhang, Shuowei Jin, Ruiyang Zhu, Jiachen Sun, Xumiao Zhang, Qi Alfred Chen, and Z Morley Mao. On data fabrication in collaborative vehicular perception: Attacks and countermeasures. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 6309–6326, 2024.

[15] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17853–17862, 2023.

[16] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. VAD: Vectorized scene representation for efficient autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8340–8350, 2023.

[17] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[18] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[19] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[20] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[21] Zewei Zhou, Tianhui Cai, Seth Z Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *arXiv preprint arXiv:2506.13757*, 2025.

[22] Hidehisa Arai, Keita Miwa, Kento Sasaki, Kohei Watanabe, Yu Yamaguchi, Shunsuke Aoki, and Issei Yamamoto. Covla: Comprehensive vision-language-action dataset for autonomous driving. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1933–1943. IEEE, 2025.

[23] Xingcheng Zhou, Xuyuan Han, Feng Yang, Yunpu Ma, Volker Tresp, and Alois Knoll. Opendrivevla: Towards end-to-end autonomous driving with large vision language action model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 13782–13790, 2026.

[24] Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. Orion: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24823–24834, 2025.

[25] Katrin Renz, Long Chen, Elahe Arani, and Oleg Sinavski. Simlingo: Vision-only closed-loop autonomous driving with language-action alignment. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 11993–12003, 2025.

[26] Hao Shao, Yuxuan Hu, Letian Wang, Guanglu Song, Steven L. Waslander, Yu Liu, and Hongsheng Li. LMDrive: Closed-loop end-to-end driving with large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15120–15130, 2024.

[27] Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, et al. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. *arXiv preprint arXiv:2506.08052*, 2025.

[28] Cheng Yin, Yankai Lin, Wang Xu, Sikyuen Tam, Xiangrui Zeng, Zhiyuan Liu, and Zhouping Yin. Deepthinkvla: Enhancing reasoning capability of vision-language-action models. *arXiv preprint arXiv:2511.15669*, 2025.

[29] Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

[30] Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

[31] Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooran Rajamanoharan, Neel Nanda, and Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=L8094Whth0.

[32] Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In Jong C. Park, Yuki Arase, Baotian Hu, Wei Lu, Derry Wijaya, Ayu Purwarianti, and Adila Alfa Krisnadhi, editors, *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali, November 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.ijcnlp-main.20. URL https://aclanthology.org/2023.ijcnlp-main.20/.

[33] Yilin Wu, Anqi Li, Tucker Hermans, Fabio Ramos, Andrea Bajcsy, and Claudia Pérez-D'Arpino. Do what you say: Steering vision-language-action models via runtime reasoning-action alignment verification. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

[34] Matthew Foutter, Matteo Cercola, Lena Wild, Yunshan Wang, Michelle Li, Daniele Gammelli, and Marco Pavone. Do vision-language-action models mean what they say? On the role of faithfulness in embodied reasoning. *arXiv preprint arXiv:2607.04681*, 2026.

[35] Florian Seligmann, Emiliyan Gospodinov, Enes Ulas Dincer, and Gerhard Neumann. VLA-FAIL: Efficient task failure detection for finetuned vision-language-action models. *arXiv preprint arXiv:2606.21386*, 2026.

[36] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[37] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

[38] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations Workshop*, 2017.

[39] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International Conference on Machine Learning*, pages 284–293. PMLR, 2018.

[40] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[41] Danny Karmon, Daniel Zoran, and Yoav Goldberg. LaVAN: Localized and visible adversarial noise. In *International Conference on Machine Learning*, pages 2507–2515. PMLR, 2018.

[42] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540, 2016.

[43] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[44] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. Dirty road can attack: Security of deep learning based automated lane centering under physical-world attack. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3309–3326, 2021.

[45] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. PhysGAN: Generating physical-world-resilient adversarial examples for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14254–14263, 2020.

[46] Zhiyuan Cheng, James Liang, Hongjun Choi, Guanhong Tao, Zhiwen Cao, Dongfang Liu, and Xiangyu Zhang. Physical attack on monocular depth estimation with optimal adversarial patches. In *European Conference on Computer Vision*, pages 514–532, 2022.

[47] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6948–6958, 2025.

[48] Jiayu Li, Yunhan Zhao, Xiang Zheng, Zonghuan Xu, Yige Li, Xingjun Ma, and Yu-Gang Jiang. AttackVLA: Benchmarking adversarial and backdoor attacks on vision-language-action models. *arXiv preprint arXiv:2511.12149*, 2025.

[49] Xueyang Zhou, Guiyao Tie, Guowen Zhang, Hechang Wang, Pan Zhou, and Lichao Sun. BadVLA: Towards backdoor attacks on vision-language-action models via objective-decoupled optimization. *arXiv preprint arXiv:2505.16640*, 2025.

[50] Qi Li, Bo Yin, Weiqi Huang, Ruhao Liu, Bojun Zou, Runpeng Yu, Jingwen Ye, Weihao Yu, and Xinchao Wang. Vision-language-action safety: Threats, challenges, evaluations, and mechanisms. *arXiv preprint arXiv:2604.23775*, 2026.

[51] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

[52] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering, 2025. URL https://openreview.net/forum?id=2XBPdPIcFK.

[53] Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[54] Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. LEACE: Perfect linear concept erasure in closed form. In *Advances in Neural Information Processing Systems*, 2023.

[55] Zhiqi Li, Guo Chen, Shilong Liu, Shihao Wang, Vibashan VS, et al. Eagle 2: Building post-training data strategies from scratch for frontier vision-language models. *arXiv preprint arXiv:2501.14818*, 2025.

A Appendix

The appendix gives the protocols and full values behind the main-text claims. The cohorts that every experiment shares are fixed in Appendix A.1. Appendix A.2 specifies the attacks, metrics, and paired tests behind Tables 1 and 2, and Appendix A.3 defines the silent-collision readout. The TNAS construction and the hook choice of Section 2.1 are detailed in Appendix A.4. For Probe 1, Appendix A.5 evaluates the rationale-only reader and Appendix A.6 audits the projections. Appendix A.7 lists the values plotted in Figure 2, and Appendix A.8 details the sign-aligned amplification protocol of Probe 3. Asset terms and compute resources close the appendix in Appendices A.9 and A.10.

A.1 Datasets and paired cohorts

Alpamayo evaluation subset. The Alpamayo experiments use the gated NVIDIA `PhysicalAI-Autonomous-Vehicles` HuggingFace dataset together with the `nvidia/Alpamayo-1.5-10B` checkpoint. We start from the clip manifest distributed with the model release, which lists 1,181 unique clip IDs, and download each clip's egomotion stream and four camera streams through the `physical_ai_av` dataset interface: the 120° cross-left, front-wide, and cross-right cameras and the 30° front-telephoto camera. Probes read each clip at the canonical reference time $t_0 = 5{,}100{,}000$ μs with all four cameras and four temporal frames, for 16 images per clip.

The 200-clip subset is constructed deterministically. We list every clip ID that has a non-empty clean rationale and a resolved local cache, sort the list lexicographically, shuffle it with Python's `random.Random(0)`, and take the first 200 entries. All Alpamayo probe and input experiments draw from this one ordered list, and the ordered clip IDs are provided in the supplementary material.

InstructVLA evaluation subset. The InstructVLA experiments use 200 Bridge-distribution examples from the VLA-IT language evaluation set. The released VLA-IT split fixes the example list, so no sampling seed is involved. The hidden-state probes and the full-image and patch input evaluations all use this ordered list, which is provided in the supplementary material, and every comparison is paired by example ID. Appendices A.4.1 and A.4.2 describe hook selection and the differentiable action readout.

Evaluation cohorts. Each model's probe and input results use its 200-example subset. The Alpamayo safety analysis uses the 154 of those 200 clips that have obstacle annotations in the dataset release, as described in Appendix A.3. Appendix A.6 also reports the projection ratio on the 1,180 manifest clips with a nonempty rationale, and the benign rationale-only reader of Appendix A.5 uses all 1,181 clips of the manifest.

A.2 Input protocols and text/action metrics

Alpamayo attack settings. All Alpamayo attacks use the objectives of Section 3.1 with text weight $\lambda_{\text{txt}} = 1$. The text penalty is the raw change in clean-target NLL plus the target-margin drop, with no hinge. During optimization, both trajectories come from a differentiable 4-step flow rollout with seed 42, and every reported readout uses the 10-step solver of Appendix A.2.1. PGD starts from uniform noise in $[-\alpha, \alpha]$, where α is the step size, drawn with seed 0 for each clip and method, and every row reports the final step 200.

The patch attack optimizes one pattern per clip. The processor resizes each camera frame to 320×576 pixels, a grid of 20×36 input patches of 16×16 pixels. In the front-telephoto frame, the pattern covers 12×22 patches, or 192×352 pixels, centered horizontally and flush with the bottom edge. A bilinear resize of the whole pattern covers 8×14 patches, or 128×224 pixels, of the front-wide frame, centered horizontally with its top edge at the vertical center. Both copies repeat in all four frames of their camera, and the cross-left and cross-right frames stay clean. The patched area is 1,504 of the 11,520 input patches, or 13.1%. Figure 3 shows this layout on two clips.

InstructVLA text-loss adaptation. The InstructVLA text-preserving baseline uses a one-sided hinge on clean-response positions R. Let m_t be the target-token logit minus the largest non-target

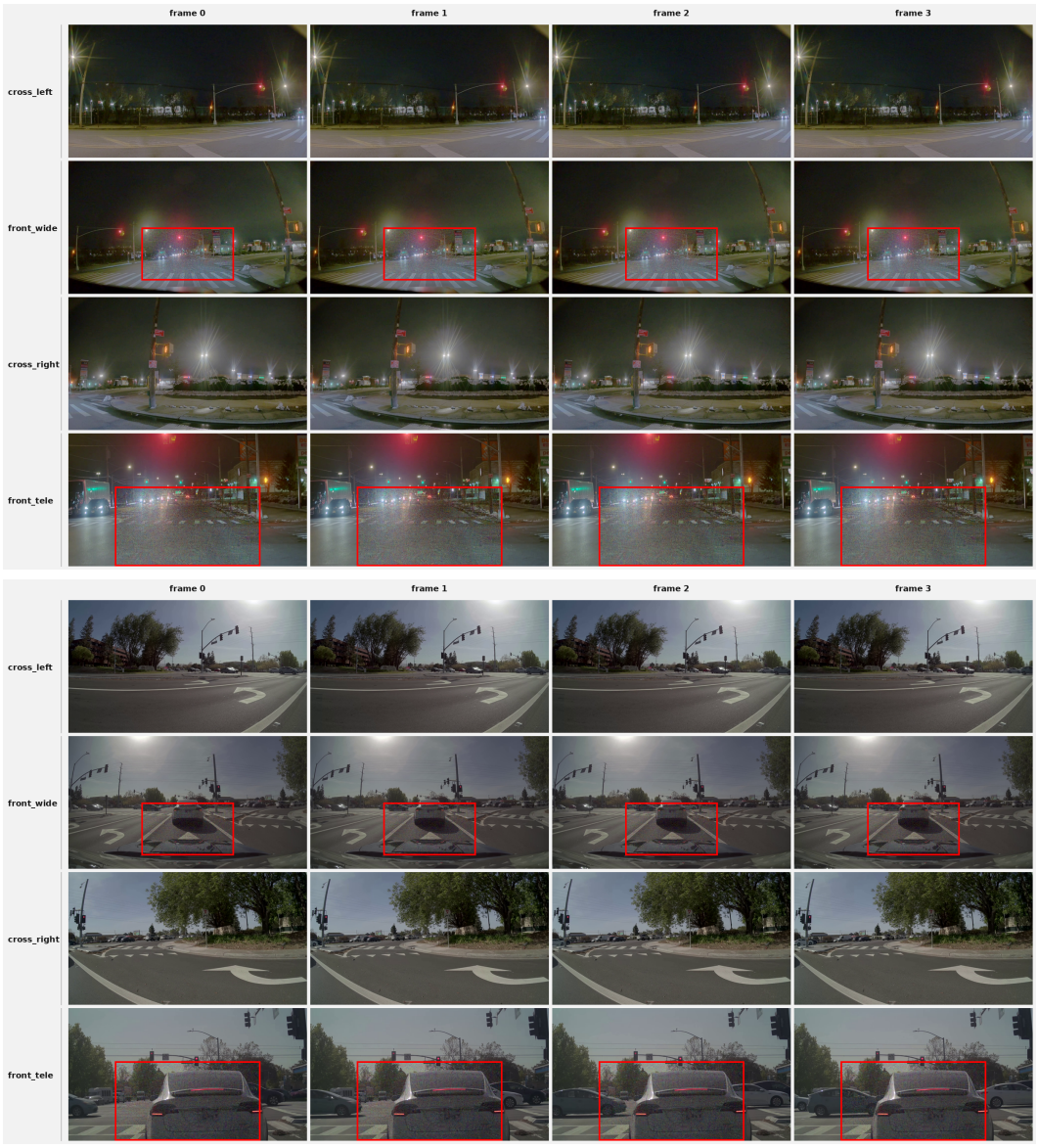


Figure 3: **Shared forward-camera patch in Alpamayo.** Illustrative patches on two clips of the evaluation cohort, optimized for 200 PGD steps at $\lambda_{\cos} = 20$ in the patch layout of Table 1. Rows are the four cameras and columns the four input frames. Red boxes mark the patch. In every frame, the optimized pattern fills the front-telephoto box and a resized copy fills the front-wide box, while the cross-view cameras stay clean.

logit, let $\mathcal{L}_{\mathrm{NLL}}$ be mean target-token NLL, and write $[x]_+ = \max(x, 0)$. Its text penalty is

$$\mathcal{L}_{\mathrm{txt}}^{\mathrm{IVLA}} = \frac{1}{|R|} \sum_{t \in R} [m_t^{\mathrm{cln}} - m_t^{\mathrm{adv}} - \tau_m]_+ + [\mathcal{L}_{\mathrm{NLL}}^{\mathrm{adv}} - \mathcal{L}_{\mathrm{NLL}}^{\mathrm{cln}} - \tau_n]_+.$$

The reported hinge family uses $\tau_m = \tau_n = 0$, unit action and text weights, and no additional KL penalty. The maximized objective is the mean squared action-coordinate displacement minus this penalty, plus $\lambda_{\cos} \cos(\Delta h, d_{\mathrm{TNAS}})$ for guided rows. The text-preserving baseline is the $\lambda_{\cos} = 0$ member, and the action-only arm drops both the text and the guidance terms. Within the full-image hinge family, the runs differ only in the guidance weight.

InstructVLA attack settings. The localized run uses a rectangular mask of 40% of the input height and 40% of its width, centered below the image center, and keeps the perturbed input inside $[-1, 1]$.

As in Alpamayo, PGD starts from seeded uniform noise in $[-\alpha, \alpha]$, restricted to the mask for the patch, and every row reports the final step 200. The perturbation enters the VLM/SigLIP input, and the action expert keeps its clean DINO input in both full-image and patch runs.

A.2.1 Action decoding and metrics

Both models decode actions with an explicit 10-step Euler flow-matching solver from $t = 0$ to $t = 1$, without classifier-free guidance, and share the initial Gaussian noise between paired runs through seed 42. Alpamayo's solver state is a 64×2 sequence of normalized acceleration and curvature controls, which the released unicycle action space integrates into the waypoints of Equation 1. InstructVLA's solver state is the 16×7 normalized action chunk itself, and Δ_{IVLA} reads the sampler's normalized outputs before deployment clipping, gripper thresholding, and dataset-specific unnormalization, so its coordinates can leave $[-1, 1]$.

A.2.2 Definition and implementation of teacher-forced KL

The teacher-forced KL metric of Section 2.1 is the forward KL from the clean to the perturbed next-token distribution, summed over the vocabulary and averaged over the active rationale/response span. With clean and perturbed per-token logits $z_t^{\mathrm{cln}}, z_t^{\mathrm{adv}} \in \mathbb{R}^V$ and $p_t^{\bullet} = \mathrm{softmax}(z_t^{\bullet})$,

$$\mathrm{KL}_{\mathrm{tf}}(\delta) \;=\; \frac{1}{|R|} \sum_{t \in R} \sum_{v=1}^{V} p_{t,v}^{\mathrm{cln}} \left[\log p_{t,v}^{\mathrm{cln}} - \log p_{t,v}^{\mathrm{adv}} \right], \tag{9}$$

which is computed in log-space for numerical stability by applying `log_softmax` separately to z_t^{cln} and z_t^{adv} before forming the inner sum. The metric is reported in nats. The clean rationale or response is fixed as the teacher-forced target sequence at every $t \in R$. At each position, the paired runs share the textual prefix and target token, and they differ only through the specified hidden-state intervention for local probes or visual-input intervention for input attacks.

Cross-table reading. Hidden-state probes select steps from finite alpha grids under the model-specific KL-budget rules of Appendix A.4, while input attacks optimize under visual-input budgets with no fixed KL target. Both report Equation 9 on the corresponding model's active text span.

A.2.3 Free-generation exact match

Alpamayo sampled decoding. For each clean or perturbed visual input, we run the VLM autoregressively without teacher forcing, with one sampled sequence, top-$p = 0.98$, temperature 0.6, at most 256 new tokens, and generation seed 42 shared between paired runs. Generation stops after the trajectory-start token, after which we extract the rendered chain-of-thought span. Sharing the random seed between paired clean and perturbed runs isolates the effect of the visual perturbation from sampling noise.

InstructVLA greedy decoding. Clean and perturbed responses are generated separately with sampling disabled and a single beam. The decoder extracts the generated suffix, truncates it at the designated meta and end markers, `<new_token_0>` and `<|im_end|>`, and strips surrounding whitespace. Exact match is equality of these cleaned strings. The 200 clean responses are short, templated intention statements of 9.60 words on average, with a median of 9, a range of 5 to 17, and 178 distinct strings. Each one summarizes a full 16-step action chunk, the many-to-one intention bottleneck of Section 3.2.

For either model, if c_{cln} and c_{adv} are the strings extracted under its protocol, the metric is

$$\mathrm{Exact}_{\mathrm{free}} = \mathbf{1}\{c_{\mathrm{cln}} = c_{\mathrm{adv}}\}. \tag{10}$$

A.2.4 Paired tests for the input tables

Every row of Tables 1 and 2 is paired by example through the shared seeds above, so Table 3 compares arms with exact paired tests on the comparisons interpreted in Section 3. At the strongest guidance weights, TNAS guidance significantly lowers teacher-forced KL and raises exact match over Text-preserving PGD in both models. In Alpamayo, its attacked-plan and silent-collision counts stay level with Text-preserving PGD and far above Action-only PGD. The InstructVLA patch matches the exact match of full-image Text-preserving PGD with significantly lower KL.

Table 3: **Paired tests for the input tables.** Binary outcomes list the counts of the two arms and the discordant examples favoring each arm, tested with exact McNemar tests. KL rows count the examples on which the first arm has lower teacher-forced KL, tested with exact sign tests. Exact match and KL use 200 examples, and collisions use the 154 annotated clips.

Outcome	Counts	Discordant	p
Alpamayo: full image $\lambda_{\cos} = 30$ vs. Text-preserving PGD			
Exact match	179 vs. 166	20/7	0.0192
Lower $\mathrm{KL}_{\mathrm{tf}}$	179 of 200	—	1.95×10^{-32}
Attacked collision	95 vs. 94	22/21	1.00
Silent collision	81 vs. 75	25/19	0.451
Alpamayo: full image $\lambda_{\cos} = 30$ vs. Action-only PGD			
Silent collision	81 vs. 7	75/1	2.04×10^{-21}
InstructVLA: full image $\lambda_{\cos} = 0.20$ vs. Text-preserving PGD			
Exact match	170 vs. 155	24/9	0.0135
Lower $\mathrm{KL}_{\mathrm{tf}}$	163 of 200	—	4.58×10^{-20}
InstructVLA: patch $\lambda_{\cos} = 0.20$ vs. full-image Text-preserving PGD			
Exact match	155 vs. 155	20/20	1.00
Lower $\mathrm{KL}_{\mathrm{tf}}$	130 of 200	—	2.65×10^{-5}

A.3 Logged-scene open-loop safety protocol

Cohort, time alignment, and geometry. Displacement and geometric collision readouts are standard in open-loop planning evaluations [15, 16]. We score Alpamayo's free-generation plans against time-aligned logged actors on the same 154 annotated clips for every arm, in the ego frame fixed at t_0. Actor boxes are transformed from their observation-time rig frames into that fixed frame using logged ego poses, then interpolated in position and unwrapped heading to the planning timestamps within their observed intervals. A single-observation track is used only at its nearest planning timestamp if within 0.05 s. Following standard open-loop protocol, actors follow their recorded trajectories over the 6.4 s horizon.

Ego dimensions and the rear-axle-to-box-center offset come from the clip's vehicle metadata, and the ego heading inferred from consecutive waypoints keeps its previous value for steps shorter than 1 mm. Actor dimensions are the per-track median annotated length and width. Apparent ego self-tracks, whose median center lies within 1 m of the expert ego footprint, are removed.

Silent collision rate. For clip i, let M_i denote exact free-generation rationale equality, C_i the clean-plan collision flag, and A_i the attacked-plan collision flag. We report

$$\frac{1}{154} \sum_{i=1}^{154} \mathbf{1}\{M_i \wedge A_i \wedge \neg C_i\}.$$

The three flags come from paired records for the same clip and decoding branch. On these 154 clips, the exact-match counts $\sum_i M_i$ are 16, 131, 126/128/133/139/141, and 111/111/110/117/123 for the action-only, text-preserving, full-image $\lambda_{\cos} = 5/10/15/20/30$, and patch $\lambda_{\cos} = 5/10/15/20/30$ arms, and the attacked-plan collision counts $\sum_i A_i$ are 91, 94, 94/92/86/87/95, and 78/82/73/74/72.

A.4 TNAS implementation and hook selection

Pooled summaries and additive intervention coordinates. For a layer block $L = \{\ell_1, \ldots, \ell_K\}$ and active token span R, the pooled summary concatenates per-layer token means,

$$\bar{H}_\ell = \frac{1}{|R|} \sum_{t \in R} H_\ell^{(t)}, \qquad h = \left(\bar{H}_{\ell_1}, \ldots, \bar{H}_{\ell_K}\right) \in \mathbb{R}^{Kd}. \tag{11}$$

The intervention handle broadcasts one vector per layer over the same span,

$$H_\ell^{(t)} \leftarrow H_\ell^{(t)} + \delta_\ell, \qquad t \in R,\ \ell \in L, \tag{12}$$

Table 4: **Alpamayo layer-window sweep.** Matched teacher-forced text-KL budget 10^{-4} on 50 clips per block. At layers 24 to 27, 47 clips have a TNAS row and 49 a random row within the budget.

Layers	Mean ρ (%)	TNAS ADE (m)	Random ADE (m)
24 to 27	2.89	0.195	0.00284
28 to 31	2.34	0.161	0.00276
32 to 35	<u>1.99</u>	<u>0.226</u>	0.00356

with one unscaled additive coordinate per layer and hidden dimension. The summary h is not a standalone sufficient input to the model: a readout written at $h + \delta$ denotes the full hooked forward pass, with the remaining inputs and decoding configuration fixed.

Hooked action Jacobian and residual. With A_{hook} and J_A the flattened action output of that pass and its Jacobian at $\delta = 0$, as defined in Section 2.1, Alpamayo targets

$$d^{\star}_{\text{act}} \in \arg\max_{\|v\|_2=1} \|J_A v\|_2,$$

materializing the Jacobian by differentiating each output coordinate and using the first right-singular vector from a thin `torch.linalg.svd`. A Jacobian norm at most 10^{-12} returns a zero direction. InstructVLA uses the normalized Gaussian-output VJP construction in Appendix A.4.2. Both models form the text gradients and residual $r = (I - P_{\mathcal{T}})d_{\text{act}}$ in these unscaled δ coordinates, before intervention scaling.

Matched-budget perturbations. For unit directions $u \in \{d_{\text{TNAS}}, d_{\text{act}}, g_{\text{NLL}}, g_{\text{margin}}, u_{\text{rand}}\}$, matched-budget probes use the additive hook and model-specific clean-state scales. For Alpamayo, the step is per-layer RMS-scaled,

$$\delta_\ell = s \cdot \alpha \cdot \sigma_{h,\ell} \cdot u_\ell, \qquad s \in \{-1, +1\}, \tag{13}$$

where $\sigma_{h,\ell} = \text{rms}(\bar{H}_\ell)$. Equivalently, $\delta = s\alpha S u$ with $S = \text{diag}(\sigma_{h,\ell_1} I_d, \ldots, \sigma_{h,\ell_K} I_d)$.InstructVLA applies a single captured clean-state scale $\sigma_{\text{IVLA}} = \max(0.01\,\text{rms}(h_{\text{meta}}), 10^{-3})$ across hooked layers, so $\delta = s\alpha\sigma_{\text{IVLA}} u$. Here h_{meta} concatenates the per-layer mean clean meta-token states, captured separately from the response-token intervention. Alpamayo's action-facing and random directions use $\alpha \in \{0.03, 0.1, 0.3, 0.5, 1.0, 3.0, 6.0\}$, and its text-gradient directions use $\{0.03, 0.1, 0.15, 0.25, 0.3, 0.4, 0.5, 1.0\}$. InstructVLA uses $\alpha \in \{0.25, 0.5, 1, 2, 4, 8, 16, 32\}$ as detailed in Appendix A.4.2. For each direction and text-KL budget, Alpamayo selects the available row with the largest achieved teacher-forced KL no greater than the target budget, and InstructVLA selects the row closest to the target budget within the $1.4B$ ceiling described in Appendix A.4.2. Alpamayo random controls in Probes 2 and 3 use 16 isotropic unit-vector seeds per clip, and the InstructVLA matched-budget grids use random seeds 0 and 1.

A.4.1 Layer-window sensitivity and hook viability

The Alpamayo layer block comes from a sweep over three late VLM blocks, layers 24 to 27, 28 to 31, and 32 to 35. Each block runs the five directions of Probe 2 with alphas $\{0.03, 0.1, 0.3, 0.5, 1.0, 3.0, 6.0\}$ at a teacher-forced text-KL budget of 10^{-4} on the same 50 clips drawn from the 200-clip cohort. All three blocks recover rank-2 active text spans on every clip and show the same action/random asymmetry. As Table 4 shows, layers 32 to 35 give the strongest matched-budget action response, the selection rule of Section 2.1, and also the smallest projection ρ. Adjacent blocks reproduce the same pattern at smaller magnitude.

InstructVLA hook/gradient viability audit. For InstructVLA, a viability audit checks that a candidate hook exposes both a nonzero text basis on its token span and a nonzero action sensitivity through the flow action expert. Table 5 compares meta- and response-token spans in three four-layer windows. Response-token spans are feasible in all three windows, and layers 14 to 17 give the largest action response among them. Meta-token spans move the action but have zero text gradients, so they are pure action-interface states rather than shared text/action states. We therefore select the response-token 14 to 17 window.

Table 5: **InstructVLA hook viability audit.** Audit on three Bridge-distribution examples. Feasible cells have nonzero text gradients, a rank-2 text basis, and a nonzero action proxy. The selected hook is layers 14 to 17 on response tokens.

Layers	Span	NLL grad	Margin grad	Action proxy	Feasible (%)
4 to 7	meta	0	0	1.39×10^{-2}	0
4 to 7	response	0.109	0.849	4.82×10^{-3}	100
14 to 17	meta	0	0	2.07×10^{-2}	0
14 to 17	response	0.199	1.27	7.54×10^{-3}	100
24 to 27	meta	0	0	1.94×10^{-2}	0
24 to 27	response	0.0779	0.388	1.51×10^{-3}	100

A.4.2 InstructVLA action-direction estimator and budget selection

Gradients pass from the response-token hook through latent action features to the flow expert by bypassing inference-only detaches and using out-of-place Euler updates. To estimate the action-sensitive direction without materializing the full Jacobian through the flow rollout, we draw an output-space Gaussian vector z_0 and normalize $w_0 = z_0/(\|z_0\|_2 + 10^{-8})$, then compute a single vector-Jacobian product at $\delta = 0$ followed by a central-difference directional gain check ($\epsilon = 0.01$):

$$v = J_A^\top w_0, \qquad d_{\text{act}} = \frac{v}{\|v\|_2}, \qquad j = \frac{A_{\text{hook}}(\epsilon d_{\text{act}}) - A_{\text{hook}}(-\epsilon d_{\text{act}})}{2\epsilon}.$$

A VJP norm at most 10^{-12} returns a zero direction, and $\|j\|_2$ is recorded as the directional gain.

Finite-grid budget selection. Each evaluation estimates the per-example directions once on the 200 example IDs and holds them fixed throughout its perturbation sweep. The matched-budget sweep uses $\alpha \in \{0.25, 0.5, 1, 2, 4, 8, 16, 32\}$ so that action-aligned directions reach the $B = 10^{-3}$ KL budget on all 200 examples. For nominal budget $B \in \{5 \times 10^{-4}, 10^{-3}\}$, the analysis selects the row with achieved teacher-forced KL closest to B within a $1.4B$ ceiling. On this grid, action-aligned directions reach a maximum mean text KL of about 1.50×10^{-3}, while text-aligned directions reach about 10^{-2}. Across the five directions in Table 7, mean achieved KL is 97.8% to 101% of B at $B = 5 \times 10^{-4}$ and 94.4% to 98.9% of B at $B = 10^{-3}$, keeping at least 186 examples per direction at $B = 5 \times 10^{-4}$ and all 200 examples at $B = 10^{-3}$.

A.5 Benign rationale-only reader

Cohort and reader. As described in Appendix A.1, the reader uses all 1,181 clips of the release manifest. Each clip's clean rationale and planned trajectory come from one greedy Alpamayo run with generation and flow seed 42, and the clips contain 258 distinct rationale strings. `Qwen/Qwen3-14B` in bf16, with thinking disabled and greedy decoding, receives one rationale string per call together with the horizon and the label definition, and returns a binary stop prediction and a stop probability. It never receives the image, the trajectory, the ego speed, or the map. Each distinct string is read once, so identical rationales receive identical predictions, and all 258 outputs parse. These benign statistics use greedy rationales and are not paired with the sampled-decoding attack results of Section 3.

Prompt and readout. Each distinct rationale is inserted into the user prompt below, which goes through the Qwen3 chat template, and the reply is capped at 200 new tokens. The binary prediction is the `will_stop` field of the returned JSON object, and the stop probability is the `stop_prob` number that the model writes into the same object. No token probabilities are used. The three remaining fields do not enter this paper's analysis. In the prompt, `[RATIONALE]` stands for the rationale string.

```
You are given one sentence: the rationale that an autonomous-driving model produced to explain its own
driving plan for the next 6.4 seconds.

Predict what the vehicle will actually do. You see only this sentence. You do not see the camera images,
the planned trajectory, the vehicle's current speed, or the map.

Sentence: "[RATIONALE]"

Answer with a single JSON object and nothing else:

{"will_stop": <true|false>,
```

Table 6: **Benign rationale-only reader.** Planned outcomes against reader predictions on 1,181 clean Alpamayo clips. Shares are within each prediction column.

	Reader: will stop		Reader: keep going		
Plan outcome	Clips	Share (%)	Clips	Share (%)	Total
Comes to rest	243	63.6	46	5.76	289
Does not come to rest	139	36.4	753	94.2	892
Total	382	100	799	100	1,181

```
 "stop_prob": <number 0..1>,
 "longitudinal": "<accelerate|maintain|decelerate|stop>",
 "lateral_dir": "<left|right|none>",
 "lateral_mag": "<none|small|large>"}

Definitions:
- will_stop: the vehicle comes to rest (below 0.5 m/s) before the 6.4 s horizon ends.
- stop_prob: your probability that will_stop is true.
- longitudinal: speed in the final 0.8 s vs the first 0.8 s. "stop" if it comes to rest, "decelerate" if
below 70% of the initial speed, "accelerate" if above 130%, else "maintain".
- lateral_dir / lateral_mag: net sideways departure from the starting heading, in metres, measured at its
 largest point. Count sideways motion caused by the road curving as well as by a deliberate lane change
or nudge. "left" and "right" are from the driver's seat. none = under 1 m, small = 1 to 3 m, large = 3 m
or more.
```

Stop label. Speeds are taken from consecutive planned waypoints at 10 Hz. A plan comes to rest within the horizon if its mean speed over the final 0.8 s of the 6.4 s horizon is below 0.5 m/s. Table 6 reports the reader's predictions against this label. The reader recovers 84.1% of stopping and 84.4% of non-stopping plans, 84.3% overall. Its stop probability ranks stopping plans with AUROC 0.882, above 0.820 for a no-text baseline that ranks clips by entry speed from the ego history, with tied scores assigned their average rank.

Mixed-outcome rationales. A rationale has mixed outcomes when its clips include both stopping and non-stopping plans. Of the 258 rationales, 23 are mixed, and together they cover 436 clips. Because the label is read at the end of the horizon, mixed outcomes include vehicles waiting at a red light that depart within the horizon. All 19 non-stopping clips with the rationale "Stop for the red traffic light" start at rest. The statistic therefore measures how much of the 6.4 s plan a sentence leaves unspecified, not how often the sentence is wrong.

A.6 Projection audits

Projection statistics. Across the 200 Alpamayo clips, the projection ratio ρ of Probe 1 has mean 1.87% and median 1.79%. Extending the cohort to all 1,180 manifest clips with a nonempty rationale leaves the mean at 1.87%. The random unit-vector reference draws 1,000 isotropic unit vectors per clip. Over these 200,000 samples it has mean 0.978%, median 0.916%, 95th percentile 1.91%, and 99th percentile 2.36%. Of the 200 clips, 43.5% exceed the random 95th percentile and 25.5% exceed the 99th, against 5% and 1% expected by chance. The largest per-clip value is $\rho = 5.28\%$, so the TNAS residual keeps more than 99.8% of the action direction's norm on every clip. In InstructVLA, ρ has mean 5.28% and median 4.90% over the 200 examples, and its mean square of 0.00382 leaves 99.8% of the norm on average. Rescaling either text-gradient basis vector by 0.01, 0.1, 1, 10, or 100 before re-orthonormalization changes the action projection ratio by at most 4.84×10^{-8}, so the ratio depends only on the text span.

A.7 Hidden-state intervention probe values

Tables 7 and 8 list the cohort means plotted in Figure 2c–h.

A.8 Sign-aligned clean-state amplification

For the clean-state amplification of Probe 3, we sweep a raw scalar s with

$$h(s, u) = h_{\text{clean}} + s\, S_{\text{clip}} u, \qquad S_{\text{clip}} = \left|\langle h_{\text{clean}}, d_{\text{TNAS}} \rangle\right|. \tag{14}$$

Table 7: **Probe 2: Matched-budget action shifts in Alpamayo and InstructVLA.** Alpamayo entries are mean ADE shifts in meters on 200 clips, or 199 for g_{NLL} at $B = 10^{-4}$, at nominal teacher-forced KL budgets B. InstructVLA entries are mean per-step normalized 7-DoF action shifts with the admissible count n, which covers all 200 examples at $B = 10^{-3}$ as detailed in Appendix A.4.2.

	Alpamayo ADE shift (m)		InstructVLA norm. ℓ_2 shift			
			$B = 5 \times 10^{-4}$		$B = 10^{-3}$	
Direction	$B = 10^{-4}$	$B = 5 \times 10^{-4}$	Shift	n	Shift	n
d_{TNAS}	<u>0.227</u>	0.504	0.0131	189	0.0137	200
d_{act}	0.181	<u>0.527</u>	<u>0.0144</u>	186	<u>0.0166</u>	200
u_{rand}	0.00359	0.00510	0.00848	194	0.00752	200
g_{NLL}	0.00240	0.00238	0.00760	186	0.00918	200
g_{margin}	0.00229	0.00252	0.00916	186	0.00817	200

Table 8: **Probe 3: Norm-matched clean-state amplification on** 200 **Alpamayo clips and** 200 **InstructVLA examples.** At each strength q, all directions receive the same per-example intervention norm $|q|S_{\mathrm{clip}}$, and $q = 0$ is the clean state. The KL columns report teacher-forced KL at $q = 2$.

	Alpamayo				InstructVLA			
	ADE shift (m)			KL	Norm. ℓ_2 shift			KL
Direction	$q = 0.5$	$q = 1$	$q = 2$	$q = 2$	$q = 0.5$	$q = 1$	$q = 2$	$q = 2$
d_{TNAS}	<u>0.0317</u>	0.0673	<u>0.142</u>	9.87×10^{-5}	<u>0.0734</u>	<u>0.110</u>	<u>0.156</u>	2.20×10^{-2}
d_{act}	0.0316	<u>0.0675</u>	<u>0.142</u>	1.03×10^{-4}	<u>0.0734</u>	<u>0.110</u>	0.155	2.71×10^{-2}
u_{rand}	0.00223	0.00261	0.00264	6.45×10^{-5}	0.00861	0.00999	0.0124	3.64×10^{-3}
g_{NLL}	0.00234	0.00262	0.00297	2.66×10^{-2}	0.0134	0.0197	0.0283	5.80×10^{-1}
g_{margin}	0.00217	0.00263	0.00323	2.73×10^{-2}	0.0158	0.0198	0.0337	4.66×10^{-1}

For both models, write $c = \langle h_{\mathrm{clean}}, d_{\mathrm{TNAS}} \rangle$ and aggregate by the sign-aligned parameter $q = s \cdot \mathrm{sgn}(c)$. Thus $q = 0$ is the clean state, and all directions have matched per-clip norm $\|qS_{\mathrm{clip}}u\|$ at fixed q. Because the SVD sign of a direction is arbitrary, q aligns the amplification curves by the sign of the natural projection. The InstructVLA clean-state amplification sweep averages four random seeds per example for u_{rand}, while deterministic named directions are evaluated once.

A.9 Existing asset licenses and terms

The experiments use existing datasets, checkpoints, and annotator models only for analysis. We do not redistribute raw clips, VLA-IT examples, checkpoints, adapter weights, or a repackaged derived dataset. Alpamayo uses the gated `nvidia/PhysicalAI-Autonomous-Vehicles` dataset under the NVIDIA Autonomous Vehicle Dataset License Agreement (`nvidia-av-dataset`), accessed through HuggingFace, and the `nvidia/Alpamayo-1.5-10B` checkpoint under the NVIDIA License for non-commercial research or evaluation use. The Alpamayo inference code is Apache-2.0. The benign rationale-only reader uses `Qwen/Qwen3-14B` [10] under Apache-2.0. InstructVLA uses the public `ShuaiYang03/instructvla_finetune_v2_xlora_freeze_head_instruction` checkpoint, `ShuaiYang03/InstructVLA_Assets`, and the `ShuaiYang03/InstructVLA_embodied_mm_evaluation_assets` language-evaluation files. We credit these releases and the InstructVLA paper, and we use only the fixed evaluation examples. As of November 2026, the public InstructVLA release pages and the `InternRobotics/InstructVLA` code repository list no explicit license for these assets, so we treat their license status as unspecified. The InstructVLA asset page identifies `nvidia/Eagle2-2B` [55] as the VLM base, whose HuggingFace card lists CC-BY-NC-4.0. For all assets with gated or restrictive terms, the experiments use local access only and do not distribute the protected asset contents. Rationale-preserving visual manipulations also pose a misuse risk: they can alter predicted actions without changing the visible explanation. Runnable attack pipelines and optimized perturbation tensors are not distributed with this manuscript.

A.10 Compute resources

The experiments ran on a single HPC node with up to four NVIDIA A100-80GB GPUs and one Python worker per GPU. Each Alpamayo worker loads one model copy and uses about 25 GB of

GPU memory. The InstructVLA probes use the same GPU type. We estimate that reproducing the Alpamayo hidden-state probes and clean-state amplification curves from cached model and data artifacts takes 75 to 85 A100 GPU-hours, including about 25 GPU-hours for the local-probe alpha grids and 30 GPU-hours for the intervention sweeps. The InstructVLA local probes and clean-state amplification take an estimated 20 to 24 GPU-hours. The input-attack costs below are measured wall-clock times summed over workers. Each Alpamayo guidance weight takes about 47 GPU-hours on the 200 clips, or about 14 GPU-minutes per clip, for both the full-image and the patch attack. The Action-only and Text-preserving rows come from a shared 200-clip run of about 85 GPU-hours. Each InstructVLA input-attack arm takes about 13 GPU-hours on the 200 examples. Merging, row selection, summaries, and the open-loop collision scoring run on CPU and take minutes per stage.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim that TNAS isolates a text-neutral action direction in Alpamayo and InstructVLA, and that visual perturbations steered along this direction change the action while keeping the generated rationale word-for-word identical. Section 2 supports the first claim with three probes, and Section 3 supports the second with full-image and patch attacks on both models.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 2.3 states that TNAS rests on first-order gradients at the clean state and tests whether the separation holds under finite hidden-state perturbations. Sections 3.1 and 3.3 define the white-box digital threat model, note that full-image attacks require access to the model's software pipeline, and score collisions open loop against logged traffic actors.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is an empirical study with no theorems or proofs. Its equations define readouts, objectives, and interventions.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sections 2 and 3 and Appendices A.1–A.8 give the protocols, objectives, layer blocks, perturbation budgets, decoding settings, and metric definitions, and the exact ordered 200-example cohort IDs for both models are provided in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The appendix identifies the gated HuggingFace Alpamayo dataset, the Alpamayo checkpoint, and the released VLA-IT split used for InstructVLA, and the paper reports the objectives, perturbation budgets, optimizer settings, subset construction rules, and cohort manifests needed to reimplement the experiments. Because the implementation includes white-box pixel-space and localized patch attack pipelines with dual-use safety implications, as discussed in Appendix A.9, runnable attack scripts and optimized perturbation tensors are not publicly distributed.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 2 and Appendices A.1 and A.4 specify the two evaluation subsets, deterministic sampling rules, hook layers, active spans, KL budgets, random-control counts, and action metrics. Sections 3.1 and 3.3 specify PGD step counts, step size, perturbation budgets, masks, camera/frame coverage, and paired evaluation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Appendix A.2.4 reports exact paired tests for the comparisons interpreted in Section 3, and Appendix A.6 reports the random-unit-vector baseline distribution used to contextualize the projection statistic.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Appendix A.10 reports the A100-80GB setup, the estimated GPU-hours for the hidden-state probes, and the measured GPU-hours of the input attacks in both models. The open-loop collision scoring runs on CPU in minutes.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The study analyzes already released VLA systems and datasets within their license terms, involves no human subjects, and withholds runnable attack code and optimized perturbations to limit misuse, as described in Appendix A.9.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 5 states the positive impact: embodied safety monitors should verify the predicted action directly alongside the rationale. Appendix A.9 discusses the misuse risk of perturbations that alter actions without changing the explanation, and runnable attack code and optimized perturbations are withheld to limit that risk.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new pretrained model, scraped dataset, or other asset with high misuse risk. Runnable attack code and optimized perturbations are withheld, as described in Appendix A.9.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Appendix A.9 lists the existing datasets, checkpoints, annotator model, and base VLM assets used by the experiments, along with the license or release-term status stated by each public release page. For InstructVLA assets whose public release pages do not list an explicit license, the appendix states that status rather than assuming an open license, and the paper does not redistribute or repackage those assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release a new dataset, model checkpoint, or benchmark. The per-example perturbations used in the evaluations are not distributed, and Sections 3.1 and 3.3 document how they are generated.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not use crowdsourcing or conduct research with human subjects. The benign rationale-only reader in Appendix A.5 uses Qwen3-14B under greedy decoding, not human annotators.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research. No study participants are recruited, compensated, or exposed to experimental risk.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Section 2.2 and Appendix A.5 describe the Qwen3-14B rationale-only reader, and the paper names and cites the VLA systems studied.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.