STAGE: Diagnosing Semantic Transfer at Grounded Execution in Embodied Agents

Anonymous ACL submission

Abstract

Embodied language grounding requires more than identifying the referent of an instruction: recovered semantics must also control the action an agent exposes. We study this missing link as a **semantic–action gap**, where instruction semantics are available but weakly expressed in native continuous actions. We introduce **SAT-Bench**, a fixed-observation counterfactual framework that holds the visual scene and agent state fixed while changing only instruction semantics. Across two LIBERO-based counterfactual splits, Qwen2.5-VL reliably recovers the intended target, but OpenVLA rarely changes its action. Target recovery reaches 100.0% and 95.8%, whereas action sensitivity remains only 6.8% and 7.7%. Hidden-state, threshold-free, cross-policy, and rollout diagnostics show that the gap is grounded, behavior-relevant, and not a metric artifact. To expose this boundary, we introduce **VISA**, a lightweight semantic-action interface that translates recovered semantics into ALLOW, DEFER, and target-consistency decisions. VISA reduces invalid-instruction blind execution from 92.7% to 2.8% while preserving 94.0% of normal commands. These findings suggest that embodied language evaluation should test whether parsed instruction semantics reach the action interface.

1 Introduction

Multimodal language-grounding evaluations often ask whether a model identifies the correct object, relation, or instruction state (Goyal et al., 2017; Agrawal et al., 2018; Thrush et al., 2022; Su et al., 2024). Embodied language agents add an action-interface requirement: recovered semantics should affect whether and how an action is exposed. However, a vision-language-action (VLA) policy (Kim et al., 2025; Octo Model Team et al., 2024) may recover the requested target and still move toward a plausible object from scene priors, or may parse a prohibited command and still expose an executable motion. We therefore ask an action-boundary question: *when the world is fixed, does changing instruction semantics change the native continuous action accordingly?*

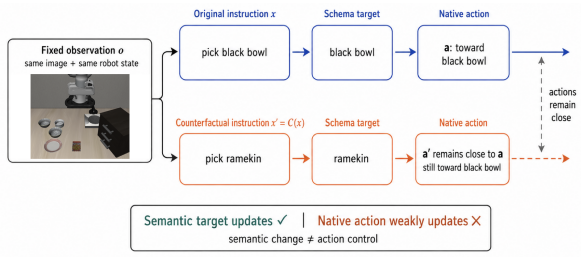

Figure 1: Fixed-observation counterfactual protocol. The observation and agent state are fixed while only the instruction semantics changes.

We call this interface-level requirement *semantic–action transfer* (SAT): recovered instruction semantics should affect whether and how a native action is exposed. When semantics are recoverable but remain weakly expressed in the policy's native seven-degree-of-freedom (7-DoF) action, we say the policy exhibits a *semantic–action gap* (SAG). To measure this gap, we use a *fixed-observation counterfactual protocol*: holding the observation and agent state fixed, we change only instruction semantics and test whether the exposed native action changes accordingly. Figure 1 illustrates this protocol, which separates *semantic availability* from *semantic control of action*. This distinction is the central thesis of the paper: semantic availability is not semantic action control, so embodied language evaluation should measure semantic-to-action transfer rather than only semantic parsing or task success.

This paper makes this action-boundary problem measurable in two steps: first through a benchmark protocol for diagnosing semantic–action transfer, and then through an interface that exposes action-boundary decisions. We introduce SAT-BENCH (*Semantic–Action Transfer Benchmark*), a fixed-

observation benchmark protocol spanning target-name changes, relation-defined targets, invalid commands, prohibitions, and paraphrase controls. Across these interventions, we observe a consistent semantic–action gap. On LIBERO (Liu et al., 2023) target-name and pixel-grounded relation swaps, Qwen2.5-VL (Bai et al., 2025) recovers the intended target in 100.0% and 95.8% of cases, yet OpenVLA action sensitivity, measured by paraphrase-calibrated native action changes, is only 6.8% and 7.7%, respectively. Additional probes, threshold-free analyses, attribution tests, and short-horizon rollouts rule out external-probe, threshold, and grounding-only explanations. The same boundary appears for non-executable language, where invalid or prohibited instructions can still trigger blind native execution.

These diagnostics motivate an explicit semantic-action interface: recovered semantics should not only be readable, but should determine whether a native action is allowed, deferred, or flagged before exposure. We therefore introduce VISA (*Verifiable Interface for Semantic Action*), a lightweight interface that converts recovered semantics into action-exposure decisions. VISA extracts a compact instruction-conditioned schema, checks admissibility, passes through valid native actions, defers invalid ones, and flags allowed actions that contradict the recovered target. The goal is not to build a new low-level controller, but to expose and regulate the semantic-action boundary. Empirically, VISA reduces invalid-instruction blind execution from 92.7% to 2.8%, preserves 94.0% of normal commands, and surfaces wrong-target or ambiguous native actions before execution. In short, VISA operationalizes the paper's thesis at the action boundary: recovered semantics should determine when to allow, defer, or flag native action exposure.

Our contributions are four-fold:

- a formulation of *semantic–action transfer* and *semantic–action gaps* as interface-level requirements for embodied language grounding;
- SAT-BENCH, a release-ready fixed-observation counterfactual benchmark protocol with prompt transformations, ActCheck labels, policy adapters, and evaluation scripts;
- diagnostic evidence that named and relation-defined targets can be recoverable externally and inside OpenVLA representations while remaining weakly transferred to native actions;
- VISA, a schema-based semantic-action interface that converts recovered semantics into allow, defer, and target-consistency decisions without replacing the low-level policy.

2 Related Work

Grounded language evaluation, counterfactuals, and probing. Multimodal grounding work tests whether models recover intended objects, relations, or scene states from language and perception. Counterfactual and contrastive evaluations further ask whether predictions rely on intended semantic variables rather than dataset priors or superficial correlations (Goyal et al., 2017; Agrawal et al., 2018; Thrush et al., 2022). Related embodied and procedural benchmarks study planning, grounding, scene consistency, and robustness under semantic perturbations (Su et al., 2024; Chakraborty et al., 2025). We target a narrower action-interface question: holding the observation and agent state fixed, does changing only the instruction induce the corresponding change in the exposed action? We also draw on representation probing, which tests whether task-relevant information is present in intermediate states (Alain and Bengio, 2016; Hewitt and Manning, 2019; Belinkov, 2022); in our setting, probing helps separate absent semantic recovery from failed transfer into action (Zhang et al., 2025).

Embodied language agents and VLA benchmarks. Recent embodied foundation models connect visual observations, language instructions, and actions through imitation learning, multimodal pretraining, and large robot datasets (Brohan et al., 2023; Driess et al., 2023; Zitkovich et al., 2023; Open X-Embodiment Collaboration et al., 2024). OpenVLA and Octo provide open-source VLA policies that enable controlled inspection of language-conditioned action behavior (Kim et al., 2025; Octo Model Team et al., 2024), while LIBERO offers language-conditioned manipulation tasks for transfer evaluation (Liu et al., 2023). These works primarily evaluate task success, generalization, or imitation quality. Our focus is complementary: we use native actions as a readout of grounded-language transfer rather than as the sole target of control optimization.

Invalid instructions, structured semantics, and action interfaces. A related line of work studies whether embodied agents can reject false-premise commands, infeasible requests, unsafe instructions, or scene-task inconsistencies (Hsieh et al.,

2025; Chakraborty et al., 2025; Son et al., 2025). High-level language-to-robot systems combine language reasoning with affordances, feedback, or programmatic primitives to improve executable planning (Ahn et al., 2023; Huang et al., 2023; Liang et al., 2022). Robotic language understanding has long used structured semantic representations to connect commands with objects, relations, actions, and constraints (Tellex et al., 2011; Thomason et al., 2016, 2019); modern VLMs such as Qwen2.5-VL provide image-conditioned parsing and grounding (Bai et al., 2025). VISA builds on this interface view, but uses schemas for evaluation and exposure decisions: whether recovered semantics should allow, defer, or flag an action.

3 Semantic–Action Transfer Framework

We formalize embodied instruction following as an action-interface problem with two separable requirements: the relevant semantic state must be recoverable, and the exposed continuous action must be controlled by that state. The goal is not to optimize robot motion, but to evaluate whether recovered semantics reach the native action interface. We first define semantic–action transfer under fixed observations, then introduce SAT-BENCH interventions, and finally define measurements for action sensitivity, non-execution, and target consistency.

3.1 Interface Requirement under Fixed Observations

Let o denote the visual observation together with the robot state, x a natural-language instruction, and $a = f(o, x)$ the native continuous action predicted by a VLA policy. We study an interface-level requirement: if the instruction semantics change while the world state is fixed, the exposed action should change in the corresponding semantic direction.

A counterfactual instruction $x' = C(x)$ changes the requested semantics while keeping o fixed, yielding $a' = f(o, x')$. The fixed-observation design is a measurement protocol rather than the definition of the failure itself. If both the image and the instruction change, an action difference could be caused by language, visual features, object layout, or scene priors. By holding o fixed, the protocol isolates the instruction-conditioned component of action selection.

This formulation separates two questions often conflated in embodied evaluation: whether the semantics are recoverable from (o, x), and whether the native action head expresses those semantics. We call the desired interface property semantic–action transfer (SAT). When semantic recovery is high but action control is weak, the policy exhibits a semantic–action gap (SAG). Thus, SAG is not a claim that the model lacks language understanding; it is a behavioral claim about where recovered semantics fail to reach the action boundary.

3.2 SAT-BENCH Interventions

SAT-BENCH instantiates the fixed-observation protocol through controlled semantic interventions over the instruction while the observation and robot state remain fixed. We use

$$C \in \{C_{\text{name}}, C_{\text{rel}}, C_{\text{invalid}}, C_{\text{neg}}, C_{\text{para}}\}, \quad (1)$$

where each intervention probes a distinct semantic-to-action property.

C_{name} replaces the explicitly named target while the scene is fixed, e.g., "pick up the black bowl" becomes "pick up the ramekin," testing whether target semantics control the next action. C_{rel} removes the target name and specifies the target through a spatial relation to an anchor object, testing whether grounded relational semantics reach the same action boundary. C_{invalid} converts executable commands into blank, impossible, ambiguous, or unsupported instructions, testing whether non-executable semantics inhibit action exposure. C_{neg} constructs prohibited commands such as `do_not`, `avoid`, `leave_alone`, and `except`, testing whether language can block an otherwise plausible action. C_{para} produces semantically equivalent paraphrases, which calibrate wording-induced action variation.

Together, these interventions cover the three interface behaviors needed for semantic–action transfer: redirecting actions, inhibiting inadmissible actions, and calibrating action variation induced by wording alone.

3.3 Measurement Stack

We use a three-part measurement stack. **Action-sensitivity metrics** test whether a semantic intervention changes the native action beyond wording-induced variation. **Non-execution metrics** test whether non-executable semantics suppress action exposure. **Target-consistency metrics** test whether an allowed action points toward the recovered target rather than a competing target. Together, these measurements separate three interface

questions: whether actions change, whether invalid actions are withheld, and whether exposed actions are semantically directed.

Action-sensitivity metrics. For target-changing interventions, we measure native action sensitivity using a normalized action difference:

$$D(a, a') = \left\| (a - a') \oslash \sigma_a \right\|_2 , \tag{2}$$

where σ_a is the policy-specific action standard deviation. An action is counted as sensitive if $D(a, a') > \tau$, with τ calibrated as the 95th percentile of semantically equivalent paraphrase controls. We use $\tau_{\text{OpenVLA}} = 2.2851$ for OpenVLA and $\tau_{\text{Octo}} = 2.063$ for Octo-small-1.5.

Given a semantic availability signal SemSens, we quantify the semantic–action gap as the mismatch between semantic availability and action sensitivity:

$$\text{SAG} = \text{SemSens} - \text{ActionSens}. \tag{3}$$

For target-name swaps, SemSens is schema sensitivity; for oracle relation swaps, $\text{SemSens} = 1$; and for pixel-grounded relation swaps, SemSens is grounding accuracy over all examples or 1 on the grounding-correct subset. To avoid making the diagnosis depend on a single cutoff, we also report target-vs-control AUC over the full action-delta distribution.

Non-execution metrics. For inadmissible instructions, the desired interface behavior is non-execution rather than a different target-directed motion. Let $a_{\text{orig}} = f(o, x)$ be the action under the original valid instruction and $a_{\text{inv}} = f(o, x_{\text{invalid}})$ the action under an invalid perturbation. We define

$$\begin{aligned} \text{BlindExec} &= \mathbb{I}[D(a_{\text{orig}}, a_{\text{inv}}) < \tau] , \\ \text{ActionInhib} &= 1 - \text{BlindExec}. \end{aligned} \tag{4}$$

Blind execution measures whether an invalid instruction still exposes a native action close to the original executable command. Action inhibition measures whether non-executable semantics suppress that exposure. At the schema level, an invalid instruction is rejected if the execution decision is a safe deferral such as `ask`, `hold`, `abort`, or `target-not-found`. We report invalid safe deferral together with normal-command pass rate so that a trivial all-stop system cannot score well by refusing every instruction.

Target-consistency metrics. Action sensitivity only asks whether the action changes; it does not determine whether the new action is directed toward the intended target. We therefore define ActCheck, a target-consistency diagnostic. Let p_e be the end-effector position, p_t the intended target position, p_w a competing wrong or original target, and $v = a_{1:3}$ the translational action. We compute

$$c_t = \cos(v, p_t - p_e), \qquad c_w = \cos(v, p_w - p_e). \tag{5}$$

With margin δ_{ac}, the action is target-aligned if $c_t - c_w > \delta_{\text{ac}}$, wrong-target if $c_w - c_t > \delta_{\text{ac}}$, and ambiguous otherwise.

Crossing this label with schema correctness yields *schema–action attribution*, which separates grounding failures from semantic-to-action transfer failures. A wrong schema with a wrong action is primarily a recoverability failure; a correct schema with a wrong or ambiguous action is an interface-transfer failure; and a correct schema with an aligned action is the desired case.

3.4 From Diagnostics to VISA

These measurements localize failures at the action boundary. If semantics are unrecoverable, the bottleneck is perception or language understanding; if semantics are recoverable but actions do not change, the bottleneck is semantic-to-action transfer; if invalid instructions still expose actions, the bottleneck is admissibility; and if allowed actions point to the wrong target, the bottleneck is target consistency. This motivates VISA: an interface that converts recovered semantics into allow, defer, and target-consistency decisions without replacing the low-level policy.

4 Diagnostic Evidence for the Semantic–Action Gap

We evaluate the semantic–action gap through three diagnostic questions: whether recoverable semantics control native actions, whether the gap survives artifact checks, and whether the same boundary appears when semantics should inhibit action exposure.

Unless otherwise specified, the action policy is OpenVLA and the schema or grounding probe is Qwen2.5-VL-7B-Instruct. We additionally evaluate OpenVLA-LIBERO90 and Octo-small-1.5, and use BridgeData V2 as a real-robot instruction-change diagnostic. The semantic probe measures

Policy	Domain	Intervention	N	Sem.	Act.	Gap	ActCheck
OpenVLA base	LIBERO	target-name	600	100.0	6.8	**93.2**	–
OpenVLA-LIBERO90	LIBERO	target-name	600	100.0	5.7	94.3	–
Octo-small-1.5	LIBERO	target-name	600	100.0	42.7	57.3	–
OpenVLA base	LIBERO	pixel relation	600	95.8	7.7	**88.1**	45.5
Octo-small-1.5	LIBERO	pixel relation	600	95.8	37.5	58.3	42.8
Octo-small-1.5	BridgeData V2	target/task	165	100.0	66.1	33.9	limited

Table 1: Semantic-to-action transfer across policies and domains. Gap is Sem. minus Act.; values except N are percentages.

recoverability only; it does not modify native policy actions. The full diagnostic pipeline is shown in Appendix B.

4.1 Recoverable Semantics Weakly Control Native Actions

The first diagnostic question is whether target semantics, once recoverable, actually control the native action interface. In target-name swaps, the schema target changes in 100.0% of cases, but OpenVLA actions are sensitive in only 6.8%. This shows that even an explicit target change does not reliably reach the native action. In pixel-grounded relation swaps, Qwen2.5-VL grounds the target correctly in 95.8% of cases, while OpenVLA action sensitivity remains only 7.7%. This setting is especially diagnostic because the target name is absent from the instruction and must be recovered from the image and spatial relation.

Figure 2 visualizes the main OpenVLA gap, and Table 1 extends the comparison across policies and domains. OpenVLA-LIBERO90 shows the same pattern on LIBERO target-name swaps. Octo-small-1.5 and BridgeData V2 are more responsive, but still leave residual gaps between semantic signal and action sensitivity. These results show that semantic–action transfer is a graded axis rather than an OpenVLA-specific binary failure.

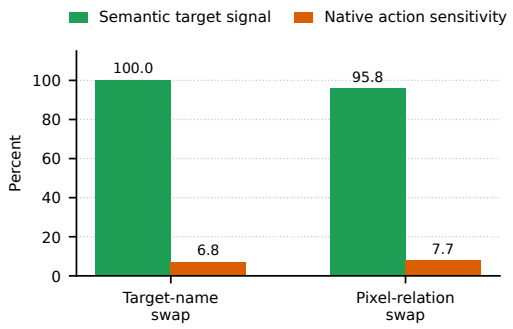

Figure 2: Main semantic–action gap on OpenVLA counterfactuals. Grounded target semantics are available, but the native action rarely changes accordingly.

Takeaway. Recovered target semantics are available, but they are weakly transferred to the native action interface. Thus, the failure is not semantic recovery alone, but semantic-to-action transfer.

4.2 Ruling Out Artifact Explanations

We next test whether the gap can be explained away by simpler alternatives: external-probe artifacts, threshold choice, grounding failure, or one-step diagnostics with no behavioral relevance.

Internal availability. A natural alternative explanation is that only the external Qwen probe recovers the target, while the action policy itself lacks the relevant information. We therefore train linear probes on OpenVLA action-prompt hidden states. Table 2 shows that named targets are readable from OpenVLA representations, and that the same pattern holds for relation-defined targets whose names are absent from the instruction. In the full pixel-relation condition, the hidden-state probe recovers the grounded target with schema sensitivity 1.000, while decoded action sensitivity remains 0.094. On an image-dependent subset where language-only or shuffled-image controls fail, schema sensitivity remains 1.000 but action sensitivity falls to 0.060. Thus, the relevant target information can be readable inside the action policy even when decoded actions remain weakly controlled by it. A preliminary activation patching check is reported in Appendix G.5; we treat the probes as semantic-availability evidence rather than as a causal repair.

Probe setting	Condition/subset	Sem. sens.	Act. sens.	Gap
Target-name	full	1.000	0.078	0.922
Target-name	mask-target	0.728	0.078	0.650
Target-name	random-label	0.061	0.078	-0.017
Pixel relation	full image+relation	1.000	0.094	0.906
Pixel relation	image-dependent	1.000	0.060	0.940
Pixel relation	language-only	0.594	0.094	0.500
Pixel relation	shuffled-image	0.656	0.094	0.561
Pixel relation	random-label	0.067	0.094	-0.028

Table 2: OpenVLA hidden-state probe results. Semantics are linearly readable, while decoded actions remain weakly sensitive.

Metric robustness. We test whether the gap is an artifact of the paraphrase-calibrated action threshold. Threshold-free comparisons show that OpenVLA target-name and pixel-relation swaps remain only weakly separable from paraphrase controls, with AUCs of 57.3 and 58.3. Spatially separated subsets lead to the same conclusion, with angle-separated results in Appendix D.4. In contrast, BridgeData V2 with Octo reaches AUC 92.9 (Table 1), showing that the metric detects instruction-

conditioned action changes when they exist.

Schema–action attribution. We also test whether the gap is merely a grounding failure. On the pixel-grounded relation split, Table 3 crosses schema correctness with ActCheck target-consistency labels. Only 4.2% of examples fall into schema-wrong regions, whereas 44.0% have a correct schema target but a wrong or ambiguous native action. This places many failures after grounding: the intended target has been recovered, but the exposed action is not target-consistent. Qualitative ActCheck examples are provided in Appendix E.

Schema/action case	Count	Rate
Correct target + aligned action	311	51.8
Correct target + wrong/ambig. action	**264**	**44.0**
Wrong target + aligned action	16	2.7
Wrong target + wrong/ambig. action	9	1.5

Table 3: Schema–action attribution on pixel-grounded relation examples.

Short-horizon behavior relevance. A final concern is that one-step action diagnostics may not matter for later behavior. We evaluate 300 paired OpenVLA-LIBERO90 rollouts from identical initial states for $K = 20$ policy steps. Raw counterfactual instructions do not reliably redirect the trajectory by themselves; aggregate target-approach effects remain mixed (Appendix G). However, first-step ActCheck labels strongly stratify later behavior. Target-aligned first actions lead to significantly higher final and integrated counterfactual-target preference than ambiguous or wrong-target first actions, with full rollout statistics in Appendix G.8. We therefore do not claim that first-step actions determine task success; instead, the rollout study validates ActCheck as a short-horizon predictive diagnostic.

Takeaway. The semantic–action gap is not explained by external probing, threshold choice, or grounding failure alone. Target information can be readable inside OpenVLA and behavior-relevant under ActCheck, yet decoded actions remain weakly controlled by the semantics.

4.3 Action Inhibition Reveals the Same Boundary

The final diagnostic question is whether semantic–action transfer is limited to target redirection, or whether the same boundary appears when semantics should inhibit action exposure. For invalid or prohibited instructions, the relevant semantic requirement is not to move toward a different object, but to avoid exposing an executable native action.

The invalid-instruction diagnostic shows that OpenVLA often keeps exposing target-directed actions when execution should be deferred. OpenVLA blindly executes 92.7% of templated invalid instructions. Negation and prohibition show the same boundary under different wording: on 800 Octo negation/prohibition variants, Octo inhibits only 29.8% of prohibited commands, leaving 70.2% blind execution (Figure 3). The 300 non-template invalid examples are independently audited in Appendix C.5, with all examples labeled as requiring deferral. Direct binary VLM gating remains brittle to linguistic form, whereas schema-level admissibility checks provide a clearer interface signal.

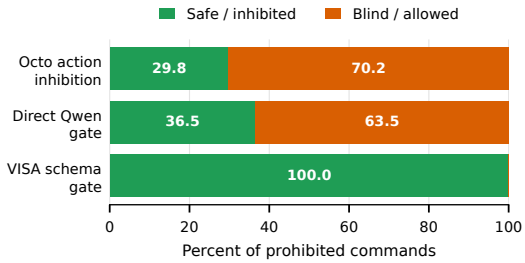

Figure 3: Non-executable language stress test on prohibited commands.

These results extend the semantic–action gap beyond target selection. In target-changing counterfactuals, recovered semantics fail to redirect the native action; in invalid and prohibited instructions, recovered non-executable semantics fail to inhibit action exposure.

Takeaway. Semantic–action failures also appear when the correct behavior is non-execution: invalid or prohibited instructions can still expose plausible native actions. This motivates an interface that can gate action exposure before execution.

Overall, these diagnostics localize failures at the action boundary: semantics can be recoverable externally and internally, yet fail to redirect native actions, inhibit invalid commands, or guarantee target-consistent exposure. Section 5 turns this diagnosis into an interface design that converts recovered semantics into allow, defer, and target-consistency decisions.

5 VISA: A Semantic-Action Interface for Action Exposure

Section 4 localizes semantic–action failures at the action boundary: recovered semantics may fail to redirect native actions, invalid instructions may

still expose executable motions, and target-correct schemas may still yield wrong-target or ambiguous actions. VISA responds to these failures with an explicit semantic-action interface. It exposes recovered instruction semantics, decides whether execution is admissible, passes through valid native actions, defers inadmissible instructions, and flags target-inconsistent actions before execution. Table 4 summarizes this mapping from diagnostics to interface decisions.

Diagnostic failure	VISA mechanism	Decision
Invalid or prohibited instruction exposes action	Admissibility gate	DEFER
Valid instruction with admissible native action	Native-action pass-through	ALLOW
Correct schema but wrong/ambiguous action	ActCheck target-consistency check	FLAG
Recoverable target with weak native transfer	Schema-level repair signal	Inspect / FLAG

Table 4: How VISA maps semantic–action diagnostics to action-exposure decisions.

VISA is lightweight because it does not replace the policy, synthesize grasps, or plan trajectories; it is interpretable because its decisions are mediated by an auditable schema and deterministic checks. Figure 4 summarizes the interface: schema extraction makes the recovered semantic state explicit, the admissibility gate decides whether a native action should be exposed, and ActCheck flags allowed actions that contradict the recovered target.

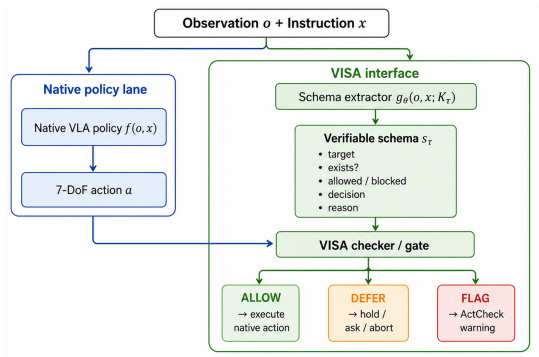

Figure 4: VISA semantic-action interface: schema extraction, admissibility checking, and action exposure.

5.1 Interface Components: Schema, Gate, and ActCheck

Schema extraction. For a task family τ, VISA extracts a task-specific schema

$$s_\tau = g_\theta(o, x; K_\tau), \qquad K_\tau \subseteq K, \tag{6}$$

where K_τ selects the keys needed by the task. We use a compact schema

$$s_\tau = (t, e, A^+, A^-, u, d, r), \tag{7}$$

where t is the target, e is target existence, A^+ and A^- are allowed and blocked symbolic action sets, u is the next symbolic action, d is the execution decision, and r is a reason. The action vocabulary is a small set,

$$A_\tau \subseteq \{\texttt{move_to_target}, \texttt{grasp}, \texttt{open}, \texttt{close}, \texttt{place}, \texttt{hold}, \texttt{ask}, \texttt{abort}\}. \tag{8}$$

Task families select different key subsets, but all schemas expose the interface variables needed for action exposure: target state, admissibility, blocked actions, next symbolic action, and reason.

Admissibility gate. The checker enforces instruction-conditioned constraints over the schema. Blank commands should not infer a task from scene priors, impossible-target commands should not execute target-directed actions, and prohibited commands should not execute blocked actions. The interface policy is

$$\pi_{\text{VISA}}(o, x) = \begin{cases} (f(o, x), \text{ALLOW}), & \text{if admissible}(s_\tau), \\ (\varnothing, \text{DEFER}(r)), & \text{otherwise.} \end{cases} \tag{9}$$

Thus, allowed native actions are passed through unchanged, while inadmissible instructions defer execution and expose a reason rather than an action.

ActCheck. The gate decides whether an action may be exposed; ActCheck decides whether an allowed action is target-consistent. Using the target recovered in s_τ, ActCheck applies the target-consistency test from Section 3.3. Wrong-target or ambiguous allowed actions are surfaced as FLAG decisions before execution. This makes target-inconsistent behavior inspectable rather than hiding it behind a single action-delta score.

5.2 Exposure Decisions, Not Control

VISA is an interface-level mechanism, not a low-level controller. It does not synthesize grasps, plan trajectories, verify collision risk, or certify long-horizon task success. Its intervention target is action exposure: whether to pass through a native action, defer execution, or flag a target-inconsistent action.

This scope matches the diagnostics in Section 4 and determines the evaluation in Section 6. We evaluate safe deferral, normal-command

pass-through, ActCheck flags, and repair signals as action-boundary decisions rather than task-completion claims. In this sense, VISA operationalizes semantic–action transfer as an auditable allow–defer–flag interface between recovered semantics and native actions.

6 Evaluating VISA as an Action-Exposure Interface

Section 5 defines VISA as an action-exposure interface rather than a low-level controller. We therefore evaluate whether it makes the intended interface decisions: defer inadmissible instructions, preserve admissible native actions, flag target-inconsistent actions, and expose repair signals without claiming task-level control.

6.1 Deferring Invalid Commands while Preserving Valid Ones

The non-execution diagnostics in Section 4.3 show that native policies can expose target-directed actions even when execution should be deferred. VISA directly addresses this failure by separating semantic extraction from admissibility checking. On templated invalid instructions, it raises safe deferral to 97.2%, reducing blind execution from 92.7% to 2.8%.

Negation and prohibition show the same pattern under different wording. Octo inhibits only 29.8% of prohibited commands, and a direct binary Qwen gate safely defers 36.5%. VISA instead extracts blocked-action and admissibility fields and applies deterministic checks, reaching 100.0% safe deferral with 0.0% blind allow.

Safe deferral alone is not sufficient: a trivial system could avoid blind execution by refusing every command. Figure 5 therefore reports a balanced semantic-interface score, defined as the worse of invalid safe deferral and normal-command pass-through. VISA scores 94.0, compared with 40.7 for Direct Qwen, 7.3 for Raw OpenVLA, and 0.0 for all-stop. Thus, the improvement is not an all-stop artifact: VISA converts invalid semantics into DEFER decisions while preserving useful native actions on admissible commands.

6.2 Flagging Target-Inconsistent Native Actions

The diagnostic results also show that action sensitivity does not imply target consistency. On pixel-grounded relation examples, OpenVLA is target-aligned in 54.5% of cases and wrong or ambiguous in 45.5%. Using the recovered schema target, ActCheck turns these wrong-target or ambiguous allowed actions into explicit pre-execution flags.

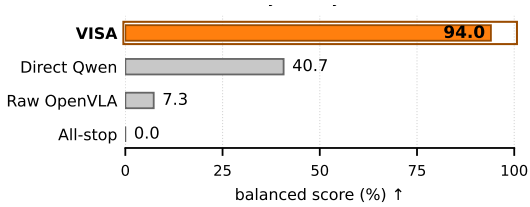

Figure 5: Deferral–utility balance. The score is the minimum of invalid safe deferral and normal pass-through.

This is the interface counterpart of schema–action attribution. When the schema is correct but the native action is inconsistent, the failure should not be hidden behind a single action-delta score; it should be surfaced before execution. ActCheck does not certify task success, but it exposes target-inconsistent native actions as FLAG decisions before execution.

6.3 Repair Signals Are Not Controller Claims

Correct schema targets can also define target-consistent directional candidates when the native action is wrong or ambiguous. We use this only as an interface-level repair signal: it shows that the recovered semantic state contains information that could support a target-consistent alternative, but it does not synthesize a full manipulation trajectory or certify task success (Appendix K.5).

Together, these results show that VISA improves action-boundary decisions: it can allow admissible native actions, defer inadmissible instructions, flag target-inconsistent actions, and expose repair signals without replacing the low-level policy.

7 Conclusion

Semantic availability is not action control. We identify the semantic–action gap as a transfer failure: instruction semantics can be recoverable while native actions remain weakly controlled. SAT-Bench measures this gap with fixed-observation counterfactuals that change only instruction semantics. Our diagnostics show that named and relation-defined targets can be recovered externally and read from OpenVLA representations, yet decoded actions remain weakly sensitive; invalid instructions reveal the same boundary through blind execution. VISA turns this distinction into ALLOW/DEFER/flag decisions, so embodied grounding should measure semantic-to-action transfer rather than semantic recovery alone.

Limitations

Our experiments focus on LIBERO-style tabletop manipulation, which provides a controlled setting for fixed-observation counterfactual evaluation but does not establish that the same quantitative gaps hold across all embodiments, real-world deployments, or non-manipulation domains. We include additional checks with OpenVLA-LIBERO90, Octo-small-1.5, and BridgeData V2, but BridgeData V2 lacks object-level metadata and should be interpreted as a real-robot instruction-change diagnostic rather than a strict object-grounded target-swap benchmark. Our action-sensitivity metrics use paraphrase-calibrated thresholds and threshold-free AUC analyses, but they do not replace full long-horizon instruction-following evaluation. The paired K = 20 rollouts and ActCheck analyses reduce the concern that the results are only first-action artifacts: first-step target-consistency labels significantly predict short-horizon target preference. However, aggregate raw counterfactual rollouts do not constitute a task-success evaluation; our claim is therefore about a measurable semantic-to-action transfer bottleneck, not full manipulation success.

Our hidden-state probes show that target information is readable from OpenVLA action-prompt representations, but probing is not a complete causal proof that the action head ignores semantics. Preliminary activation patching has limited effect, and broader causal interventions are needed to characterize how decoded actions use intermediate representations. VISA also depends on schema quality: incorrect grounding, ambiguous references, poor target-existence judgments, or inconsistent schema fields can lead to incorrect allow or defer decisions. Finally, ActCheck only verifies whether the translational action is directionally more consistent with the intended target than with a competing target; it does not check grasp feasibility, collision risk, contact dynamics, rotation, gripper state, phase transitions, or long-horizon completion. VISA should therefore be viewed as a post-hoc semantic-action interface, not an end-to-end controller or complete robot safety system.

Ethics Statement

This work studies failure modes in embodied vision-language-action agents and proposes diagnostic tools for identifying when instruction semantics are not faithfully reflected in robot actions. The intended use is to improve evaluation, transparency, and safety by exposing plausible but semantically inappropriate actions before they are executed.

VISA and ActCheck should not be interpreted as complete robot safety systems. They are post-hoc diagnostic and action-interface mechanisms that can block or flag some semantic–action failures, but they do not verify physical safety, grasp feasibility, collision risk, contact dynamics, long-horizon task completion, or recovery behavior. Real-world deployment would require additional safeguards, including conservative fail-safe control, hardware-level safety validation, calibrated uncertainty handling, human override mechanisms, and domain-specific risk assessment.

A potential risk of this work is false confidence: a system that passes VISA-style checks may still behave unsafely due to perception errors, distribution shift, ambiguous instructions, or low-level control failures. We therefore present VISA as a diagnostic and verification interface for studying semantic–action transfer, not as a deployable controller or a sufficient safeguard for real-world robots.

References

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. 2018. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4971–4980.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, and 26 others. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR.

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, and 32 others. 2023. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea.

Trishna Chakraborty, Udita Ghosh, Xiaopan Zhang, Fahim Faisal Niloy, Yue Dong, Jiachen Li, Amit Roy-Chowdhury, and Chengyu Song. 2025. HEAL: An empirical study on hallucinations in embodied agents driven by large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21226–21243, Suzhou, China. Association for Computational Linguistics.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, and 3 others. 2023. PaLM-E: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6904–6913.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Wen-Han Hsieh, Elvis Hsieh, Dantong Niu, Trevor Darrell, Roei Herzig, and David M. Chan. 2025. Do what? teaching vision-language-action models to reject the impossible. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11861–11869, Suzhou, China. Association for Computational Linguistics.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2023. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782. PMLR.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2025. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2022. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*.

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

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. 2024. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*. ArXiv:2405.12213.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, and 14 others. 2024. Open X-embodiment: Robotic learning datasets and RT-X models. In *Proceedings of the IEEE International Conference on Robotics and Automation*. ArXiv:2310.08864.

Yejin Son, Minseo Kim, Sungwoong Kim, Seungju Han, Jian Kim, Dongju Jang, Youngjae Yu, and Chan Young Park. 2025. Subtle risks, critical failures: A framework for diagnosing physical safety of LLMs for embodied decision making. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25692–25733, Suzhou, China. Association for Computational Linguistics.

Ying Su, Zhan Ling, Haochen Shi, Cheng Jiayang, Yauwai Yim, and Yangqiu Song. 2024. ActPlan-1K: Benchmarking the procedural planning ability of visual language models in household activities. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14953–14965, Miami, Florida, USA. Association for Computational Linguistics.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence*.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin Hart, Peter Stone, and Raymond J. Mooney. 2019. Improving grounded natural language understanding through human-robot dialog. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6934–6941.

Jesse Thomason, Jivko Sinapov, Maxwell Svetlik, Peter Stone, and Raymond J. Mooney. 2016. Learning multi-modal grounded linguistic semantics by playing I Spy. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 3477–3483.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. 2022. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5238–5248.

Chunhui Zhang, Zhongyu Ouyang, Xingjian Diao, Zheyuan Liu, and Soroush Vosoughi. 2025. Knowing more, acting better: Hierarchical representation for embodied decision-making. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 19144–19155, Suzhou, China. Association for Computational Linguistics.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, and 35 others. 2023. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR.

A Schema Examples

This appendix provides representative Verifiable Interaction Schemas used by VISA. The examples illustrate how the schema separates target reference, target existence, execution admissibility, and the next symbolic action.

Field	Executable instruction	Impossible target	Negated instruction
Instruction	`pick up the black bowl`	`pick up the red mug`	`do not pick up the black bowl`
Target	`black bowl`	`red mug`	`black bowl`
Target exists	`true`	`false`	`true`
Phase	`pre_grasp`	`defer`	`defer`
Allowed actions	`move_to_target, grasp`	`target_not_found`	`hold, ask`
Blocked actions	–	`move_to_target, grasp`	`move_to_target, grasp`
Next action	`move`	`target_not_found`	`hold`
Execution decision	`allow`	`defer`	`defer`

Table 5: Representative VISA schemas for executable, impossible-target, and negated instructions. Invalid instructions may still mention visible objects, but should not trigger target-directed execution.

Field	Pixel-grounded relation example
Instruction	`pick up the object to the right of the black bowl`
Anchor	`black bowl`
Relation	`right_of`
Target	`ramekin`
Target exists	`true`
Phase	`pre_grasp`
Allowed actions	`move_to_target, grasp`
Blocked actions	–
Next action	`move_to_target`
Execution decision	`allow`

Table 6: Example relation-defined schema. The target name is absent from the instruction and must be inferred from the image and spatial relation.

B Diagnostic Pipeline

Figure 6 summarizes the diagnostic pipeline used in the main paper.

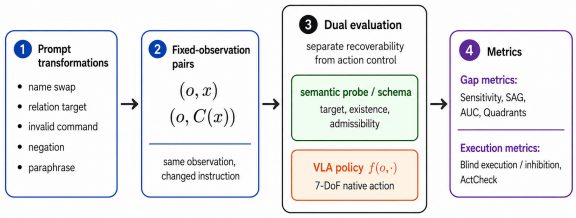

Figure 6: Diagnostic pipeline for semantic-to-action transfer under fixed-observation prompt transformations.

C SAT-Bench Construction Details

SAT-Bench is a fixed-observation semantic-to-action transfer benchmark protocol. It holds the visual observation and robot state fixed while changing only the instruction semantics through controlled prompt transformations. The benchmark is designed to test whether recovered semantics reach the action interface, rather than only whether they can be parsed or grounded.

C.1 Benchmark Statistics

SAT-Bench contains fixed-observation semantic interventions for target change, relation grounding, invalid execution, negation/prohibition, and paraphrase calibration. The splits are designed to support both semantic-recovery measurements and action-interface diagnostics.

Split	#Examples	Perturbations
Target-name swap	600	target replacement
Oracle relation-defined	600	relation-defined target replacement
Pixel-grounded relation	600	relation-defined target from image
Templated invalid instruction	1,800	blank, impossible, negation
Normal preservation	600	original valid instructions
Non-template invalid instruction	300	five invalid categories

Table 7: Benchmark statistics. The non-template invalid split contains 300 additional invalid instructions and is used for generalization and heuristic-baseline evaluation.

Artifact scope and intended use. The supplementary artifact is intended for research evaluation and benchmark reconstruction. It contains robot-manipulation metadata, synthetic or templated instruction transformations, schema labels, and audit sheets. It does not intentionally contain personally identifying information or offensive content. The artifact should not be used as a deployable robot-control system or as a complete safety benchmark.

Release plan and supplementary artifact. We include an anonymized supplementary artifact containing SAT-Bench metadata, counterfactual prompt transformations, schema labels, human-audit templates, documentation, and evaluation scripts. The artifact includes target-name swaps, oracle and pixel-grounded relation splits, templated invalid instructions, non-template invalid instructions, paraphrase controls, ActCheck labels, policy adapters, and metadata-limited BridgeData V2 diagnostics.

Raw images, simulator assets, model weights, raw model outputs, expert low-level actions, and absolute paths are excluded from the artifact. External datasets and models, including LIBERO, BridgeData V2, OpenVLA, Octo, and Qwen2.5-VL, are not redistributed and should be obtained from their original sources under their respective licenses and terms of use. BridgeData V2 is included only as metadata-limited diagnostic splits, not as a strict object-grounded benchmark.

C.2 Target-Name Swap Split

For each original LIBERO observation and instruction, we construct a counterfactual instruction by replacing the target object with another visible or semantically valid target while holding the observation and robot state fixed. The main target-pair families are:

- black bowl → ramekin,
- black bowl → cookie box,
- middle drawer → top drawer.

This split tests whether the native VLA action changes when the explicitly named semantic target changes.

C.3 Relation-Defined Splits

The oracle relation-defined split removes the target object name from the instruction and denotes the target through a spatial relation to an anchor object. For example, an instruction may refer to "the object to the right of the black bowl." The gold target is computed from simulator object positions and relation templates. This split should be interpreted as an oracle action-sensitivity test rather than a visual-grounding benchmark.

The pixel-grounded relation split uses the same relation-defined instruction style, but evaluates whether Qwen2.5-VL can infer the target from the image and relation instruction. The target object name is not present in the instruction. We report target grounding accuracy, target-existence accuracy, parse success, action sensitivity over all examples, and action sensitivity on the grounding-correct subset.

Metric	Value
Number of examples	600
Target grounding accuracy	0.958
Target-existence accuracy	1.000
Parse success	1.000
Grounding-correct examples	575
OpenVLA action sensitivity, all	0.077
OpenVLA action sensitivity, grounding-correct	0.077

Table 8: Pixel-grounded relation split summary. Qwen2.5-VL grounds the target from pixels in 95.8% of examples, but OpenVLA action sensitivity remains 7.7%.

C.4 Invalid-Instruction Splits

The invalid-instruction splits test whether non-executable instruction semantics inhibit action exposure. The templated invalid-instruction split contains blank commands, impossible targets, and negated commands derived from the same 600 LIBERO observations. Invalid examples are expected to produce safe deferral rather than native target-directed execution.

Because templated invalid instructions can favor surface-level rules, we additionally construct a 300-example non-template invalid split. This split con-

tains five categories with 60 examples each: paraphrased prohibitions, implicit impossible targets, ambiguous references, instruction conflicts, and distractor non-command mentions. All examples in this split have safe deferral as the gold execution decision.

Category	Examples	Description
`hard_paraphrased_prohibition`	60	Prohibitions without exact template phrases
`hard_implicit_impossible`	60	Instructions requiring absent objects or unsupported relations
`hard_ambiguous_reference`	60	Underspecified target references
`hard_instruction_conflict`	60	Commands with mutually incompatible constraints
`hard_distractor_noncommand`	60	Object mentions that should not trigger execution
Total	300	Five non-template invalid categories

Table 9: Non-template invalid split statistics. The split is designed to be difficult for template matching, but it is still finite and category-based rather than a comprehensive human-written invalid-instruction corpus.

C.5 Audit of Non-Template Invalid Instructions

We audit the 300-example non-template invalid split with two internal annotation passes. The internal audit records whether the instruction should be executed or deferred, whether a target exists, the preferred non-execution action, and optional notes. The audit is used only to validate the benchmark labels and is not used to tune VISA. No external annotators, crowdworkers, or recruited human subjects were used.

Category	N	Majority defer	Action agreement	Target-status agreement
Paraphrased prohibition	60	60/60	60/60	60/60
Implicit impossible	60	60/60	60/60	60/60
Ambiguous reference	60	60/60	60/60	60/60
Instruction conflict	60	60/60	60/60	60/60
Distractor non-command	60	60/60	60/60	60/60
Overall	300	300/300	300/300	300/300

Table 10: Audit results for the non-template invalid split. All examples are majority-labeled as requiring safe deferral, and the two annotation passes agree on deferral, preferred non-execution action, and target-status labels.

Because all examples are labeled as requiring deferral, chance-corrected agreement for the binary defer/allow decision is degenerate; we therefore report raw agreement for deferral, preferred action, and target-status labels.

D Additional SAT-Bench Target-Change Results

D.1 Main Semantic–Action Gap Results

Policy	Setting	Semantic signal	N	Act. Sens. ↑	Gap
OpenVLA	Target-name	Schema = 1.000	600	0.068	0.932
OpenVLA	Oracle relation	Gold target	600	0.077	0.923
OpenVLA	Pixel relation	Grounding = 0.958	600	0.077	0.882
OpenVLA	Pixel relation, correct	Grounding = 1.000	575	0.077	0.923
Octo	Pixel relation	Grounding = 0.958	600	0.375	0.583
Octo	Pixel relation, correct	Grounding = 1.000	575	0.350	0.650
Octo	Pixel relation, angle $\geq 45^\circ$	Grounding = 1.000	44	0.523	0.477

Table 11: Main semantic–action gap results. This table provides the exact values corresponding to the main target-change diagnostics.

D.2 Target-Pair Breakdown

Target Pair	N	Schema Sens. ↑	Action Sens. ↑	SAG
black bowl → ramekin	255	1.000	0.047	0.953
black bowl → cookie box	213	1.000	0.066	0.934
middle drawer → top drawer	132	1.000	0.114	0.886
Overall	600	1.000	0.068	0.932

Table 12: Target-pair breakdown for target-name counterfactuals.

D.3 Action-Delta Distribution

Distribution	Mean	Median	P75	P90	P95
Target swap	1.268	1.199	1.549	2.041	2.483
Paraphrase control	1.108	1.067	1.484	1.926	2.285

Table 13: Normalized action-delta distributions. The OpenVLA sensitivity threshold is set to the 95th percentile of paraphrase-control deltas.

D.4 Spatial Subset Robustness

Subset	N	Action Sens. ↑	SAG
All samples	600	0.068	0.932
Distance > median	300	0.057	0.943
Angle > median	300	0.080	0.920
Distance and angle > median	204	0.083	0.917
Angle > 30°	496	0.052	0.948
Angle > 45°	44	0.045	0.955

Table 14: Spatial robustness analysis for target-name swaps. The semantic–action gap remains large even when target directions are spatially separated.

D.5 Policy Robustness

Policy	Schema Sens. ↑	Action Sens. ↑	SAG
OpenVLA base	1.000	0.068	0.932
OpenVLA LIBERO-90	1.000	0.057	0.943
Octo-small-1.5	1.000	0.427	0.573

Table 15: Target-name swap robustness across action policies. Octo-small-1.5 is more action-sensitive than OpenVLA, but still remains below schema-level sensitivity.

D.6 Octo Relation Robustness

For Octo-small-1.5, we calibrate a policy-specific paraphrase-control threshold:

$$\tau_{\text{Octo}} = 2.063. \tag{10}$$

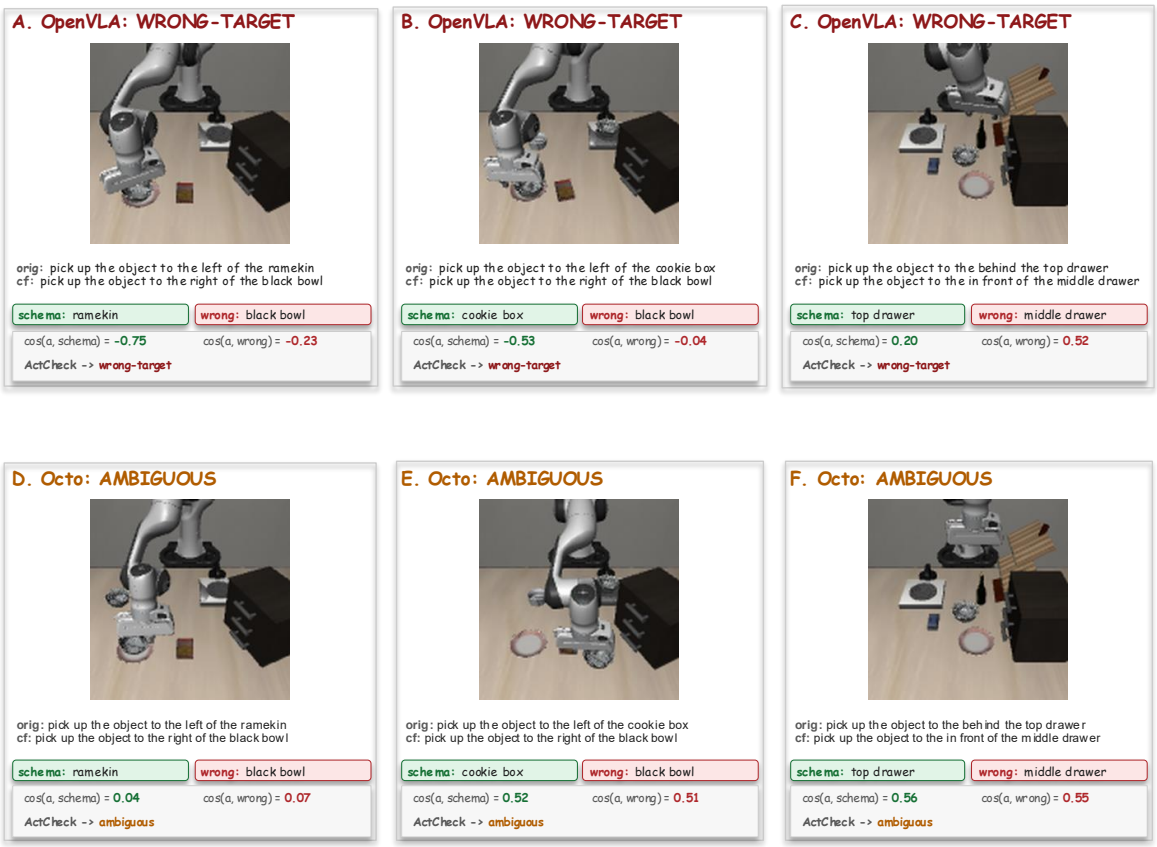

Figure 7: Qualitative ActCheck cases on pixel-grounded relation counterfactuals. In all cases, the schema probe recovers the counterfactual target, but the native policy action is classified as wrong-target or ambiguous. The examples illustrate that the semantic target can be available while the exposed action remains weakly aligned with the recovered target.

Table 16 reports Octo results on the pixel-grounded relation split. Octo reduces but does not eliminate the semantic–action gap.

Split	N	Action Sens. ↑	SAG
All	600	0.375	0.583
Grounding-correct	575	0.350	0.650
Angle $\geq 45^\circ$	44	0.523	0.477

Table 16: Octo-small-1.5 robustness on pixel-grounded relation-defined counterfactuals. Octo is more instruction-sensitive than OpenVLA, but the gap remains substantial.

D.7 ActCheck Details

Policy	Split	N	Target-aligned ↑	Wrong-target ↓	Ambiguous ↓	Flag ↓
OpenVLA-LIBERO90	Pixel relation	600	0.545	0.285	0.170	0.455
Octo-small-1.5	Pixel relation	600	0.572	0.393	0.035	0.428
Octo-small-1.5	Grounding-correct	575	0.597	0.367	0.037	0.403
Octo-small-1.5	Angle $\geq 45^\circ$	44	0.795	0.159	0.045	0.205

Table 17: ActCheck results on pixel-grounded relation counterfactuals. OpenVLA and Octo both produce many actions that are wrong-target or ambiguous under the recovered schema target. Octo is more action-sensitive than OpenVLA, but increased sensitivity does not eliminate target-consistency failures.

E Qualitative ActCheck Cases

This appendix provides qualitative examples for the ActCheck target-consistency diagnostic.

Each case is selected from pixel-grounded relation counterfactuals where the Qwen2.5-VL schema probe correctly grounds the counterfactual target, but the native policy action is classified as wrong-target or ambiguous by the target-consistency check. Figure 7 visualizes these failure modes, and Table 18 lists the corresponding case metadata.

Policy	Original instruction	Counterfactual instruction	Example family	Schema target	ActCheck label
OpenVLA-LIBERO90	`pick up the object to the left of the ramekin`	`pick up the object to the right of the black bowl`	`black bowl / ramekin`	`ramekin`	wrong
OpenVLA-LIBERO90	`pick up the object to the left of the cookie box`	`pick up the object to the right of the black bowl`	`black bowl / cookie box`	`cookie box`	wrong
OpenVLA-LIBERO90	`pick up the object behind the top drawer`	`pick up the object in front of the middle drawer`	`middle / top drawer`	`top drawer`	wrong
Octo-small-1.5	`pick up the object to the left of the ramekin`	`pick up the object to the right of the black bowl`	`black bowl / ramekin`	`ramekin`	ambig.
Octo-small-1.5	`pick up the object to the left of the cookie box`	`pick up the object to the right of the black bowl`	`black bowl / cookie box`	`cookie box`	ambig.
Octo-small-1.5	`pick up the object behind the top drawer`	`pick up the object in front of the middle drawer`	`middle / top drawer`	`top drawer`	ambig.

Table 18: Representative qualitative ActCheck cases. In all cases, the VLM schema probe correctly grounds the counterfactual target, but the native policy action is either more aligned with the wrong or original target, or ambiguous under the recovered schema target.

F Additional SAT-Bench Invalid-Instruction and Gate Results

F.1 Main Gate Mitigation Results

Method	Blind ↓	Safe Def. ↑	Normal Pass ↑	False Block ↓
OpenVLA raw	0.927	0.000	1.000	0.000
All-stop	0.000	1.000	0.000	1.000
Direct Qwen	0.593	0.407	0.982	0.018
VISA gate	0.028	0.972	0.940	0.060
Hybrid	0.028	0.972	0.922	0.078

Table 19: Gate mitigation on templated invalid instructions and normal instructions. This table provides the exact values for the safety–utility comparison.

F.2 Invalid-Instruction Breakdown

Perturbation	Schema Rej. ↑	Action Inhib. ↑	Gap	Blind Exec. ↓
Blank	0.920	0.090	0.830	0.910
Impossible	1.000	0.060	0.940	0.940
Negation	1.000	0.068	0.932	0.932
Overall	0.973	0.073	0.901	0.927

Table 20: Invalid-instruction breakdown. Schema-level rejection is high, but native OpenVLA actions often remain close to the original task action.

F.3 Per-Perturbation Mitigation

Perturbation	N	OpenVLA Blind	Gated Blind	Safe Def. ↑
Blank	600	0.910	0.083	0.915
Impossible	600	0.940	0.000	1.000
Negation	600	0.932	0.000	1.000
Overall	1800	0.927	0.028	0.972

Table 21: Per-perturbation mitigation results for VISA checker and gate.

F.4 Prompt and Schema-Quality Ablation

Schema Prompt	Normal Pass	False Block	ASK/PROMPT	Self-Contrad.
naive/default	0.133	0.867	0.527	0.340
normal-preservation v2	0.765	0.235	0.150	0.085
v2 + repair	0.850	0.150	0.150	0.000
domain-aware v3	0.940	0.060	0.060	0.000
domain-aware v3 + repair	0.940	0.060	0.060	0.000

Table 22: Prompt and schema-quality ablation. The final domain-aware prompt substantially reduces over-deferral on normal LIBERO manipulation commands.

F.5 Checker and Gate Ablation

Method	Invalid Blind ↓	Safe Def. ↑	Normal Pass ↑	False Block ↓
OpenVLA raw	0.927	0.000	1.000	0.000
All-stop baseline	0.000	1.000	0.000	1.000
Schema monitor only	0.927	0.000	0.940	0.060
Checker without gate	0.927	0.000	0.940	0.060
Checker + Gate	0.028	0.972	0.940	0.060

Table 23: Checker/gate ablation. Detection alone does not reduce blind execution; execution gating is necessary.

F.6 Strict versus Execution-Aware Rejection

Perturbation	Exec-Aware	Strict	Gap	Gated Blind
Blank	0.915	0.915	0.000	0.083
Impossible	1.000	1.000	0.000	0.000
Negation	1.000	0.213	0.787	0.000

Table 24: Strict versus execution-aware rejection. Execution-aware safety can be high even when strict symbolic completeness, such as exhaustive blocked-action listing, is weaker.

F.7 Schema Field Quality

Field / Decision	Precision	Recall	F1 / Acc.
Target-swap `target_object`	1.000	1.000	1.000
Normal `target_object`	1.000	1.000	1.000
Invalid safe-deferral `next_action`	–	–	0.972
Invalid `blocked_actions` task	0.935	0.348	0.508

Table 25: Schema field quality. Target fields are reliable in the controlled target-swap and normal splits, and execution-aware safe deferral is strong. Strict blocked-action completeness is weaker, especially for invalid and negated commands.

F.8 Non-Template Invalid Generalization

Method	Safe Def. ↑	Normal Pass ↑	Invalid Allow ↓
Template heuristic	0.123	–	0.877
Direct Qwen	0.870	0.982	0.130
VISA schema gate	0.780	0.940	0.220
Hybrid	1.000	0.922	0.000

Table 26: Non-template invalid method comparison. The template heuristic fails on non-template invalid instructions, while Direct Qwen, VISA, and the hybrid expose different safety–utility trade-offs.

Category	Heuristic Correct ↑	VISA Correct ↑
`hard_paraphrased_prohibition`	0.000	1.000
`hard_implicit_impossible`	0.000	0.917
`hard_ambiguous_reference`	0.000	0.517
`hard_instruction_conflict`	0.317	0.517
`hard_distractor_noncommand`	0.300	0.950

Table 27: Non-template invalid category breakdown. VISA is strongest on paraphrased prohibitions, implicit impossible targets, and distractor non-command mentions, but remains weaker on ambiguous references and instruction conflicts.

Metric	Value
Completed examples	300/300
Schema validity	1.000
Action constraint consistency	1.000
Schema-internal blind-execution violation	0.000
Safety violation	0.000
Blocked-action violation	0.017

Table 28: Schema-internal diagnostics on the non-template invalid split. These constraint-level diagnostics are distinct from the gold-level invalid-allow metric.

Instruction	Heuristic	VISA	Gold
`leave the middle drawer alone`	`ALLOW`	`ASK`	Defer
`pick up the small tool under the cookie box`	`ALLOW`	`ASK`	Defer
`there is a cookie box in the scene, but no object should be moved now`	`ALLOW`	`WAIT`	Defer

Table 29: Qualitative non-template invalid examples. The template string heuristic allows execution because it does not recognize paraphrased prohibitions, implicit impossibilities, or distractor non-command mentions; the schema gate produces safe deferrals.

G Confidence Intervals and Probe Diagnostics

G.1 Bootstrap Confidence Intervals

We report bootstrap 95% confidence intervals over examples for the main relation-defined, ActCheck, grounding, and short-horizon diagnostics. Bootstrap intervals use 10,000 resamples with random seed 13.

Table	Policy	Metric	N	Mean [95% CI]
OpenVLA relation	OpenVLA-LIBERO90	Action sensitivity	600	0.077 [0.057, 0.098]
OpenVLA relation	OpenVLA-LIBERO90	Target-aligned action	600	0.545 [0.505, 0.585]
OpenVLA relation	OpenVLA-LIBERO90	Wrong-target action	600	0.285 [0.250, 0.322]
OpenVLA relation	OpenVLA-LIBERO90	Ambiguous alignment	600	0.170 [0.140, 0.200]
OpenVLA relation	OpenVLA-LIBERO90	ActCheck flag	600	0.455 [0.415, 0.495]
Octo relation	Octo-small-1.5	Action sensitivity	600	0.375 [0.338, 0.415]
Octo relation	Octo-small-1.5	Target-aligned action	600	0.572 [0.532, 0.610]
Octo relation	Octo-small-1.5	Wrong-target action	600	0.393 [0.353, 0.432]
Octo relation	Octo-small-1.5	Ambiguous alignment	600	0.035 [0.022, 0.050]
Octo relation	Octo-small-1.5	ActCheck flag	600	0.428 [0.388, 0.468]
Pixel-grounded relation	Qwen2.5-VL-7B	Grounding accuracy	600	0.958 [0.942, 0.973]
Short horizon	OpenVLA-LIBERO90	CF instruction, CF approach	25	0.560 [0.360, 0.760]
Short horizon	OpenVLA-LIBERO90	Original instruction, CF approach	25	0.840 [0.680, 0.960]

Table 30: Bootstrap 95% confidence intervals for the core relation, ActCheck, grounding, and short-horizon diagnostics. The intervals quantify sampling uncertainty over evaluated examples; they do not address benchmark-level distribution shift.

Metric	Mean [95% CI]
Target-swap VLM schema sensitivity	1.000 [1.000, 1.000]
Target-swap OpenVLA action sensitivity	0.068 [0.048, 0.088]
Target-swap semantic-action gap case rate	0.932 [0.910, 0.952]
Invalid-instruction OpenVLA blind execution	0.927 [0.915, 0.939]
Checker/gate invalid blind execution	0.028 [0.020, 0.036]
Checker/gate safe deferral	0.972 [0.963, 0.979]
Checker/gate normal pass	0.940 [0.920, 0.958]
Checker/gate false block	0.060 [0.042, 0.080]

Table 31: Bootstrap 95% confidence intervals for target-name, invalid-instruction, and gate-mitigation metrics.

G.2 Schema-Backbone Robustness

We additionally test whether the schema target-extraction result depends on the Qwen2.5-VL-7B backbone. On the same 600 target-swap examples, Qwen2.5-VL-3B reaches the same schema sensitivity as Qwen2.5-VL-7B. This is a schema-backbone robustness check rather than a full gate-backbone ablation.

Schema Probe	Schema Sens. ↑	Action Sens. ↑	SAG
Qwen2.5-VL-7B	1.000	0.068	0.932
Qwen2.5-VL-3B	1.000	0.068	0.932

Table 32: Schema-backbone robustness on the same 600 target-swap examples. Both Qwen2.5-VL-7B and Qwen2.5-VL-3B recover the target change, while the native OpenVLA action remains weakly sensitive.

G.3 OpenVLA Hidden-State Probe Breakdown

The hidden-state probe trains linear classifiers on OpenVLA action-prompt representations to test whether counterfactual target identity is readable before action decoding. The probe is diagnostic: it measures semantic availability in the representation, while action sensitivity is still computed from the decoded native continuous action.

Model	Condition	Layer	Target Acc.	Schema Sens.	Action Sens.	Gap
OpenVLA base	full	8	1.000	1.000	0.078	0.922
OpenVLA base	full	16	1.000	1.000	0.078	0.922
OpenVLA base	full	final	1.000	1.000	0.078	0.922
OpenVLA base	language-only	final	1.000	1.000	0.078	0.922
OpenVLA base	image-only	final	0.500	0.000	0.078	-0.078
OpenVLA base	mask-target	final	0.864	0.728	0.078	0.650
OpenVLA base	random-label	final	0.283	0.061	0.078	-0.017
OpenVLA-LIBERO90	full	8	1.000	1.000	0.050	0.950
OpenVLA-LIBERO90	full	16	1.000	1.000	0.050	0.950
OpenVLA-LIBERO90	full	final	1.000	1.000	0.050	0.950
OpenVLA-LIBERO90	language-only	final	1.000	1.000	0.050	0.950
OpenVLA-LIBERO90	image-only	final	0.500	0.000	0.050	-0.050
OpenVLA-LIBERO90	mask-target	final	0.864	0.728	0.050	0.678
OpenVLA-LIBERO90	random-label	final	0.297	0.117	0.050	0.067

Table 33: Full hidden-state probe table. Target identity is linearly recoverable from OpenVLA action-prompt representations for both OpenVLA base and OpenVLA-LIBERO90, but decoded action sensitivity remains low.

G.4 Pixel-Relation Hidden-State Probe

We also probe relation-defined examples where the target name is absent from the instruction. The image-dependent subset contains examples where the full image+relation representation recovers the target but the language-only or shuffled-image con-

trol fails on the same test example. This subset checks that the result is not explained only by the text template or dataset priors.

Subset	N	Schema Sens.	Action Sens.	Gap
Full all test pairs	180	1.000 [1.000, 1.000]	0.094 [0.056, 0.139]	0.906 [0.861, 0.944]
Language-only	180	0.594 [0.522, 0.667]	0.094 [0.056, 0.139]	0.500 [0.417, 0.583]
Shuffled-image	180	0.656 [0.583, 0.722]	0.094 [0.056, 0.139]	0.561 [0.483, 0.639]
Image-dependent subset	100	1.000 [1.000, 1.000]	0.060 [0.020, 0.110]	0.940 [0.890, 0.980]

Table 34: Pixel-relation hidden-state probe with bootstrap confidence intervals. Even on examples where image information is needed to recover the relation-defined target, the decoded action remains weakly sensitive.

Activation-patching metric	Value
Native action sensitivity	0.080
Patched action sensitivity	0.070
Patch moves toward CF action	0.370
Mean native Δ(original, CF)	1.290
Mean patched Δ(original, patched)	1.135
Mean distance patched-to-CF action	1.344

Table 35: Preliminary layer-16 activation patching on 100 target-swap examples. Patching counterfactual hidden states into the original prompt has limited effect on decoded actions, supporting the interpretation that readable semantic information is not straightforwardly transferred to the action decoder.

G.5 Activation Patching

We include activation patching only as a preliminary causal check. The patching experiment uses 100 target-swap examples, OpenVLA base, layer 16, and $\alpha = 1.0$, with the native OpenVLA action decoder unchanged. The limited change in decoded action sensitivity means the hidden-state probe should be interpreted as evidence of semantic availability and a representation-to-action bottleneck, not as evidence that a simple hidden-state edit is sufficient to repair low-level control.

G.6 OpenVLA QA-Style Target Probe Negative Result

Metric	Value
Examples completed	20/20
Qwen schema sensitivity	1.000
OpenVLA QA-style schema sensitivity	0.000
Original target correct	0.500
Counterfactual target correct	0.050

Table 36: Negative diagnostic. Direct QA-style probing of the action-tuned OpenVLA checkpoint is unreliable and is not used as evidence of VLA-side semantic understanding.

G.7 Short-Horizon Target Approach

We expand the short-horizon study to 300 paired OpenVLA-LIBERO90 rollouts, corresponding to 600 actual rollouts. The evaluation covers six LIBERO spatial tasks with 50 trials per task and a horizon of $K = 20$ policy steps. Each pair starts from the same initial state and compares the original instruction rollout with the counterfactual target-instruction rollout.

Method	N	CF-target approach	Original-target approach	Mean target pref.
Original instruction	300	0.797	0.677	0.000247
Counterfactual instruction	300	0.697	0.640	0.000166

Table 37: Raw paired rollout results over 300 paired $K = 20$ rollouts. Counterfactual instructions do not reliably redirect raw OpenVLA trajectories by themselves, so we do not interpret this as a task-success result.

G.8 ActCheck Predicts Short-Horizon Target Preference

We stratify the same 300 paired $K = 20$ rollouts by the first-step ActCheck label. The results show that the one-step target-consistency label is predictive of trajectory-level behavior: aligned first actions lead to higher counterfactual-target approach rates and higher final and integrated target-preference scores.

First-step ActCheck label	N	CF approach	Final target pref.	Integrated target pref.
Aligned	86	0.872	0.0660	0.0659
Wrong-target	63	0.571	0.0306	0.0312
Ambiguous	151	0.649	-0.0095	-0.0097
Overall	300	0.697	0.0206	0.0206

Table 38: Trajectory outcomes stratified by first-step ActCheck label. Target-aligned first actions lead to the strongest short-horizon preference for the counterfactual target.

Metric	Comparison	Diff.	95% CI	Perm. p
Final target preference	aligned – ambiguous	0.0755	[0.0679, 0.0829]	0.0001
Final target preference	aligned – wrong-target	0.0354	[0.0272, 0.0431]	0.0001
Integrated target preference	aligned – ambiguous	0.0756	[0.0679, 0.0830]	0.0001
Integrated target preference	aligned – wrong-target	0.0348	[0.0266, 0.0425]	0.0001
CF approach rate	aligned – ambiguous	0.2231	[0.1186, 0.3260]	0.0005
CF approach rate	aligned – wrong-target	0.3007	[0.1589, 0.4393]	0.0001

Table 39: Permutation-test comparisons for ActCheck-stratified $K = 20$ rollouts. Aligned first-step actions significantly outperform both ambiguous and wrong-target first-step actions on final target preference, integrated target preference, and counterfactual-target approach rate.

G.9 Schema-Conditioned Projection Stress Test

The schema-conditioned projection study is a proof-of-concept action-interface intervention rather than a deployable controller. The target position is obtained from simulator-accessible object states. Given end-effector position p_e and target position p_t, the translational action is projected toward the target direction:

$$a'_{1:3} = \mathrm{dir}(p_t - p_e) \cdot \|a_{1:3}\|_2 \cdot \lambda. \qquad (11)$$

This intervention tests whether explicit schema-level target information can redirect the action stream when injected at the action interface. It does not solve grasping, placing, contact dynamics, or long-horizon task completion.

Intervention	N	Raw Dist. ↓	Projected Dist. ↓	Full Success
Always-project	5	0.336	0.038	0/5
Until-close	5	0.336	0.065	0/5
Phase-aware adapter	3	0.332	0.090	0/3

Table 40: Closed-loop target-swap summary. Schema-conditioned interventions redirect trajectories toward counterfactual schema targets, but full LIBERO task success remains zero.

Setting	Release Rate	Close-but-Fail	Success
d005	0.8	0.8	0
d008	1.0	1.0	0
d010	1.0	1.0	0

Table 41: Until-close projection. Projection reliably brings the robot close to the schema target, but full success remains zero.

Horizon	Raw Final Dist.	Projection Final Dist.	Redirection	Success
160	0.3434	0.0713	+0.2721	0
220	0.3357	0.0378	+0.2978	0

Table 42: Horizon extension. Longer rollouts bring the end-effector closer to the schema target, but full success remains zero.

G.10 Phase-Aware Adapter

Task	Target	Raw Dist.	Projected Dist.	Redirection	Lift Δz	Lifted
task1	ramekin	0.3890	0.0936	+0.2954	0.1872	yes
task3	cookie box	0.3799	0.0794	+0.3005	0.0606	yes
task5	ramekin	0.2264	0.0983	+0.1281	0.0707	yes

Table 43: Phase-aware adapter results. The adapter redirects the robot toward the schema target and lifts the target in all three tested counterfactual cases, although full pick-and-place success remains unsolved.

Metric	Value
Mean raw final distance	0.3318
Mean projected final distance	0.0904
Mean redirection score	+0.2413
Target lift rate	3/3
Full LIBERO success	0/3

Table 44: Phase-aware adapter summary.

Takeaway. The rollout redirection studies support a supplementary claim: schema-level semantics can causally affect the VLA action stream when injected at the action interface. Always-on and until-close projection reduce final distance to the counterfactual schema target, and a small phase-aware adapter can lift the selected target in three cases. However, we do not claim to solve full LIBERO manipulation. Full task success remains zero in these counterfactual rollouts. These results should therefore be interpreted as evidence for trajectory-level redirection and partial target interaction, not as evidence of a complete robot controller.

H Prompt Templates

H.1 Schema Prompt

The schema prompt asks the VLM to produce a structured JSON-like object with the following fields:

- `target`,
- `target_exists`,
- `phase`,
- `allowed_actions`,
- `blocked_actions`,
- `next_action`,
- `execution_decision`.

The prompt instructs the model not to invent missing targets, not to infer a task from a blank instruction, and not to execute prohibited actions.

H.2 Pixel-Grounded Relation Prompt

For pixel-grounded relation instructions, the prompt additionally asks the model to identify:

- the anchor object,
- the spatial relation,
- the object satisfying the relation,
- whether the target exists,
- the admissible next action.

The target object name is not present in the instruction; the model must infer it from the image.

H.3 Direct Qwen Gate Prompt

The Direct Qwen Gate prompt asks the VLM to output exactly one of:

- `allow`,
- `ask`,
- `hold`,
- `abort`,
- `target-not-found`.

Unlike VISA, this baseline does not expose target, target existence, allowed actions, blocked actions, or next-action schema fields.

FloatBarrier

I Reproducibility Artifacts

The following scripts and output directories were used to construct SAT-Bench, evaluate semantic-to-action transfer, run VISA diagnostics, and reproduce the main and supplementary analyses:

- Pixel-grounded relation schema outputs: `outputs/linguistic\_blindness/qwen25vl7b\_visual\_relation\_schema\_600`
- Pixel-grounded relation analysis: `outputs/linguistic\_blindness/pixel\_grounded\_relation\_analysis`
- ActCheck script: `scripts/lb\_visa\_actcheck.py`
- Target-name ActCheck summary: `outputs/linguistic\_blindness/visa\_actcheck\_exp2/actcheck\_summary.csv`
- Visual-relation ActCheck summary: `outputs/linguistic\_blindness/visual\_relation\_actcheck/actcheck\_summary.csv`
- Octo visual-relation action sensitivity: `outputs/linguistic\_blindness/octo\_visual\_relation\_action\_sensitivity\_600`
- Octo relation analysis: `outputs/linguistic\_blindness/octo\_visual\_relation\_analysis`
- Qualitative ActCheck and CI packet: `outputs/linguistic\_blindness/qualitative\_ci\_packet`
- Qualitative ActCheck candidate cases: `outputs/linguistic\_blindness/qualitative\_ci\_packet/qualitative\_actcheck\_case\_candidates.csv`
- Selected qualitative ActCheck cases: `outputs/linguistic\_blindness/qualitative\_ci\_packet/recommended\_qualitative\_cases.csv`
- Bootstrap CI supplement: `outputs/linguistic\_blindness/qualitative\_ci\_packet/bootstrap\_ci\_core\_supplement.csv`
- Templated invalid mitigation: `outputs/linguistic\_blindness/invalid\_instruction\_mitigation`
- Non-template invalid benchmark: `outputs/linguistic\_blindness/hard\_invalid\_set\_300/benchmark.jsonl`
- Direct Qwen / VISA / hybrid risk–utility report: `outputs/linguistic\_blindness/risk\_utility\_direct\_visa\_hybrid/risk\_utility\_report.md`
- Hidden-state probe outputs: `outputs/linguistic\_blindness/p0\_oral\_main\_outputs/hidden\_state\_probe\_main\_table.csv`
- Short-horizon target approach: `outputs/linguistic\_blindness/p0\_oral\_main\_outputs/paired\_rollout\_k20\_100\_bootstrap\_ci.csv`
- Schema-conditioned projection rollouts: `outputs/linguistic\_blindness/schema\_conditioned\_projection`

J BridgeData V2 Second-Domain Diagnostic

We construct targeted BridgeData V2 real-robot diagnostics from the first public TFDS shard. The invalid-instruction subset contains 52 trajectories, 165 sampled frames, 495 invalid or prohibited instructions, and 165 normal instructions. We also construct a 165-example fixed-frame target/task-change diagnostic by replacing the original Bridge instruction with a different target or task instruction from the same shard. Because this shard does not provide reliable object metadata, the Bridge target/task split should be interpreted as an instruction-change diagnostic rather than a strict object-grounded target-swap benchmark.

Dataset / policy	N	Sem. change	Act. sens. ↑	Gap
BridgeData V2 / Octo	165	1.000	0.661	0.339

Table 45: BridgeData V2 target/task-change diagnostic. Octo is more sensitive than OpenVLA on LIBERO, but 33.9% of instruction changes remain below the Bridge-specific paraphrase threshold.

Perturbation	N	Octo action inhib. ↑	Octo blind exec. ↓	VISA safe def. ↑
Blank	165	0.527	0.473	1.000
Impossible	165	0.576	0.424	1.000
Negation	165	0.133	0.867	1.000
Overall	495	0.412	0.588	1.000

Table 46: BridgeData V2 invalid/prohibited instruction results. Octo still exhibits substantial blind execution, especially under negation; VISA schema-gated execution consistently defers invalid commands.

Split	N	Direct Qwen safe def. ↑
Blank	165	1.000
Impossible	165	1.000
Negation	165	0.030
Invalid overall	495	0.677
Normal pass	165	0.515

Table 47: Direct Qwen Gate on the BridgeData V2 diagnostic subset. The direct gate handles blank and impossible commands but is brittle under negation.

K Additional Robustness Diagnostics

This section collects robustness diagnostics for SAT-Bench and VISA. The diagnostics are organized around alternative explanations for the main semantic–action gap result: a single threshold choice, grounding failure, OpenVLA-only behavior, spatial overlap between targets, first-step-only artifacts, all-stop utility loss, and surface-form brittleness under invalid or prohibited commands. Together, these checks support the interpretation of semantic-to-action transfer as a graded action-interface property rather than a single-model failure.

K.1 Threshold-Free Action Sensitivity

Setting	N	Mean target Δ	Mean control Δ	AUC
OpenVLA target-name swap	600	1.268	1.108	0.573
OpenVLA pixel-relation swap	600	1.454	1.237	0.583
BridgeData V2 Octo target/task change	165	1.912	0.220	0.929

Table 48: Threshold-free action sensitivity summary. OpenVLA target-change and paraphrase-control deltas are only weakly separated, while the same metric strongly separates target/task changes in the BridgeData V2 Octo diagnostic.

K.2 Schema–Action Quadrant Attribution

Case type	Count	Rate
Schema correct + action consistent	311	0.518
Schema correct + action wrong/ambiguous	264	0.440
Schema wrong + action consistent	16	0.027
Schema wrong + action wrong/ambiguous	9	0.015

Table 49: Schema–action quadrants for the OpenVLA pixel-grounded relation split. The dominant failure mode is correct schema grounding with a wrong or ambiguous native action, which localizes many errors to the semantic-to-action interface.

K.3 Policy and Domain Matrix

Policy	Setting	N	Sem./oracle	Act. sens.	SAG
OpenVLA base	LIBERO target-name	600	1.000	0.068	0.932
OpenVLA-LIBERO90	LIBERO target-name	600	1.000	0.057	0.943
Octo-small-1.5	LIBERO target-name	600	1.000	0.427	0.573
OpenVLA base	LIBERO pixel relation	600	0.958	0.077	0.882
Octo-small-1.5	LIBERO pixel relation	600	0.958	0.375	0.583
Octo-small-1.5	Bridge target/task change	165	1.000	0.661	0.339

Table 50: Policy/domain matrix for semantic–action gaps. The result supports a graded behavioral-axis interpretation: the gap is not equally severe everywhere, but it remains measurable across settings.

K.4 Normal-Command Utility Breakdown

Command type	N	Normal pass	False block	Target valid
Overall	600	0.940	0.060	1.000
Spatial relation	468	1.000	0.000	1.000
Drawer/cabinet	132	0.727	0.273	1.000

Table 51: VISA normal-command utility breakdown. VISA is not an all-stop gate: it preserves all spatial-relation normal commands, while false blocks are concentrated in drawer/cabinet commands.

K.5 Schema-Conditioned Repair Proxy

Method	N	Target-aligned	Wrong/ambiguous	Intervention rate
Native OpenVLA action	600	0.545	0.455	0.000
ActCheck directional candidate	600	1.000	0.000	0.455

Table 52: Schema-conditioned repair proxy. The directional candidate is a post-hoc target-consistency proxy rather than a deployable controller. It shows that recovered schema targets provide enough information to define an action-interface repair signal.

K.6 Negation and Prohibition Stress Test

The negation/prohibition stress split is constructed from 200 normal base examples and four variants per base example: `do_not`, `avoid`, `leave_alone`, and `except`, for 800 total prohibited commands.

Method	N	Safe/Inhibited ↑	Blind/Allow ↓
Octo action inhibition	800	0.298	0.703
Direct Qwen Gate	800	0.365	0.635
VISA / Qwen Schema Gate	800	1.000	0.000

Table 53: Overall negation/prohibition stress results. Octo continues executing most prohibited commands; direct binary gating allows most commands; the structured schema gate defers all prohibited variants in this controlled diagnostic split.

Variant	Octo inhibited ↑	Direct safe deferral ↑	VISA safe deferral ↑
`avoid`	0.205	1.000	1.000
`do_not`	0.150	0.000	1.000
`except`	0.150	0.460	1.000
`leave_alone`	0.685	0.000	1.000

Table 54: Variant-level negation/prohibition stress results. Direct Qwen Gate is highly sensitive to surface form: it succeeds on `avoid`, partly handles `except`, and fails on `do_not` and `leave_alone`.

Interpretation. These diagnostics make the main claim falsifiable rather than merely adding additional benchmark numbers. Threshold-free AUCs test whether the result is a single-cutoff artifact. Spatial subsets test whether low action sensitivity is caused by overlapping target directions. Schema–action quadrants separate grounding failures from semantic-to-action transfer failures. The policy/domain matrix shows that the gap is graded: more responsive policies reduce the gap but do not remove the need to report it. Normal-command utility prevents a trivial all-stop gate from appearing safe, and the non-template invalid and negation/prohibition splits test whether non-executable semantics inhibit motion beyond surface-form matching. Taken together, these results support the paper's interface-level interpretation: semantic recovery, native action sensitivity, target consistency, and non-execution are distinct properties and should be measured separately.