WHAT HAPPENS INSIDE VLAS FOR COMPOSITIONAL GENERALIZATION? A MECHANISTIC ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) policies show strong performance on many manipulation benchmarks, but they fail to generalize compositionally: even when objects and skills are familiar, visual shortcuts or isolated object words can dominate the full instruction. Prior work develops mitigation methods based on hypothesized failure mechanisms. We instead analyze internal representations to understand when and where task-relevant information supports action generation. We study how task information flows from text prompts to final actions and what happens internally when compositional generalization fails. We introduce VLA-PATCHLENS, an analysis framework based on activation patching and path patching for localizing task-critical information. The framework compares matched clean and corrupted runs that differ in task-critical instructions or task behavior. For each matched pair, we replace the corrupted run's activation at a layer-token group with the clean-run activation and measure whether the patch recovers action loss and rollout success. We construct three compositional generalization task suites, and for $\pi_{0.5}$, the analysis shows that task-critical information in text tokens is evenly distributed across layers, with early and late layers carrying complementary parts of the task; in later layers, early text-token information becomes concentrated in image tokens and reaches action tokens through cross-attention. We also find that action-loss recovery and success-rate recovery are highly consistent, making action-loss recovery a lightweight proxy for selecting intervention targets before closed-loop evaluation. In compositional generalization failures, VLA-PATCHLENS reveals clear breaks in this task-critical information flow, and finds simple single-layer activation patching improves success rate from 0% to 3.9–23.5% on average and up to 66.7% at the best layer across three task suites, while activation-similarity scores remain uninformative.

1 INTRODUCTION

Vision-Language-Action (VLA) models have driven recent progress in generalist robotic manipulation by mapping visual observations and language commands directly to robot actions (Driess et al., 2023; Brohan et al., 2023; Intelligence et al., 2025b; Kim et al., 2025; Intelligence et al., 2026). However, deployment in complex real-world environments still requires policies to generalize beyond training task configurations. Toward this goal, a more attainable subproblem is **compositional generalization** (Fei et al., 2026): embodied agents must recombine familiar objects and learned manipulation skills to execute novel task instructions under new task-target pairings.

This setting is challenging because policies can fail even when the objects and manipulation primitives are familiar. One common failure mode comes from vision shortcuts (Fang et al., 2026; Zhan et al., 2026; Zhang et al., 2026b; Fei et al., 2026): visual input can override the language instruction, causing the policy to ignore the specified target and bias actions toward the visual input. Spatial shortcuts (Li, 2025) can also arise when the policy takes the task language into account: an object word can dominate the full sentence, causing the policy to reach the object-related location rather than the location specified by the instruction. The policy can learn these shortcuts during training because each training episode provides one task-language instruction from a narrow distribution, while diverse image tokens carry richer, more redundant signals than sparse text tokens. This imbalance can push the policy toward visual priors or isolated object words instead of the full instruction (Lian et al., 2026; Zhan et al., 2026; Fang et al., 2026; Fei et al., 2026). Prior work

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

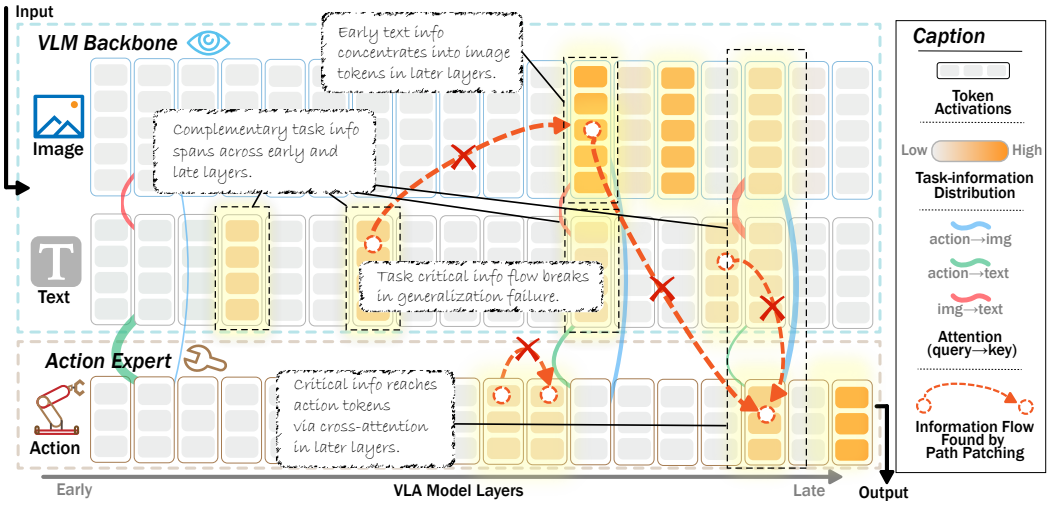


Figure 1: VLA-PATCHLENS localizes task-critical information distribution and flow across layers and token groups for the $\pi_{0.5}$ model; see Section 4 for more findings.

has mitigated these failures with reward penalties on vision-only actions (Lian et al., 2026), visual-affordance prior subtraction (Zhan et al., 2026), Counterfactual Action Guidance from a vision-only branch (Fang et al., 2026), or training-free attention recalibration under visual conflict (Zhang et al., 2026b). These approaches improve language grounding or robustness, but mostly intervene at the external behavior level and lack a direct internal account of why a policy succeeds or fails. At the same time, mechanistic interpretability methods are being used to identify behaviorally meaningful internal representations within VLA models, such as speed and movement height (Häon et al., 2025); sparse semantic and action features (Khan et al., 2025; Buurmeijer et al., 2026; Swann et al., 2026; Grant et al., 2026); event-grounded features (Jin et al., 2026); symbolic state signals (Lu et al., 2025) and state-transition information (Molinari et al., 2025); text latents (Li, 2025); and task-success subspaces (Miao et al., 2026). However, existing analyses usually focus on steering a named behavior or exposing a single task-related feature. They do not yet provide a systematic internal analysis for compositional generalization settings. This gap raises the central question of this paper: how does task information flow from the text prompt to the final actions? What happens internally when compositional generalization fails (e.g., what information is lost)?

To this end, we introduce VLA-PATCHLENS, an analysis based on activation patching and path patching (Heimersheim & Nanda, 2024; Meng et al., 2022; Goldowsky-Dill et al., 2023) to analyze task-critical internal representations in VLA models, as shown in Figure 2. For each matched clean-corrupted run pair, where the runs differ only in task-critical instruction or task behavior, we replace the corrupted run's residual-stream activation at a layer-token group (e.g., the image-token residual stream at layer L_k) with the clean-run activation. Our operational assumption is that a layer-token site carries task-critical information when this replacement substantially recovers action loss and rollout success rate toward the clean run. Sweeping this intervention across layers and token groups yields a layer-wise map of task-relevant representations and identifies where and how task-language information is represented, associated with relevant token groups, and used for output.

We construct three compositional generalization task suites and use VLA-PATCHLENS to analyze successful task following and generalization failures. For the successful task-following setting, we use successful task-following rollouts as clean runs and blank-task-language rollouts as corrupt runs, letting us localize task-critical information and track how it shifts during execution. For the failure analysis setting, we treat runs that exhibit compositional generalization behavior, even when unsuccessful, as clean runs and runs that violate the task instruction as corrupted runs. These analyses localize internal causes of generalization failures, and we find that with correct localization, simple single-layer patching can improve observable success rate. We further make this localization and patching process cheap by identifying action loss as a proxy for the closed-loop rollout. Our findings are shown in Figure 1 and Section 4. We further validate the general applicability of VLA-PATCHLENS through experiments with StarVLA-OFT and real-robot manipulation in Section 4.4.

We make three contributions. First, we propose VLA-PATCHLENS, an activation-patching tool for locating the distribution and flow of task-critical information across layers and token groups. Second, for $\pi_{0.5}$, by analyzing successful task-following rollouts, we find that task-critical information in text tokens is evenly distributed across layers, with early and late layers carrying complementary parts of the task; in later layers, early text-token information becomes concentrated in image tokens and reaches action tokens through cross-attention. Third, for $\pi_{0.5}$ compositional generalization failure rollouts, we find breaks in the task-critical information flow identified above, and single-layer activation patching improves success rate from 0% to 3.9–23.5% on average and up to 66.7% at the best layer across three task suites, while activation-similarity scores remain uninformative.

2 RELATED WORK

Mechanistic Interpretability for VLA Models. Mechanistic interpretability has become a useful tool for analyzing (Meng et al., 2022; Goldowsky-Dill et al., 2023; Hanna et al., 2025) and guiding (Bricken et al., 2023; Liu et al., 2024) the internal behavior and final outputs of LLMs and VLMs. This line of work is moving into embodied policies, where VLA studies show that internal activations contain behaviorally meaningful structure (Häon et al., 2025). Intervention-oriented work steers speed or direction with value vectors (Häon et al., 2025), shifts action choices with sparse latent directions (Khan et al., 2025), controls desired behavior with linear features (Buurmeijer et al., 2026) or sparse-autoencoder features (Swann et al., 2026), steers policies with contrastive conceptors (Miao et al., 2026), and intervenes on event-grounded sparse features (Jin et al., 2026). Diagnosis-oriented work instead asks where task information is represented: it probes symbolic states (Lu et al., 2025), reconstructs policy behavior through text latents (Li, 2025), identifies visual dominance with motor/goal pathway specialization (Grant et al., 2026), probes state transitions (Molinari et al., 2025), and traces modality dependence and shortcuts (Shi et al., 2026), and diagnoses reliance on irrelevant visual cues (Zhang et al., 2026a). Tri-Info (Yang et al., 2026) addresses interpretable failure prediction using entropy and mutual information of state and action embeddings. In this work, we focus on task-information flow from text prompts to final actions. We use activation and path patching to trace this flow and identify where it is disrupted during compositional generalization failures. This mechanistic diagnosis guides targeted interventions to restore task-information flow and mitigate these failures.

Compositional VLA Generalization. VLA policies achieve strong in-domain manipulation in representative benchmark settings (Brohan et al., 2023; Intelligence et al., 2025b; Kim et al., 2025; Liu et al., 2023), yet compositional generalization remains fragile under robustness, paraphrase, and shortcut tests (Fei et al., 2026; Zhou et al., 2025; Fang et al., 2026; Kim et al., 2026; Lian et al., 2026; Zhan et al., 2026; Zhang et al., 2026b). One line improves overall generalization through knowledge and data scaling (Brohan et al., 2023; O'Neill et al., 2024; Tian et al., 2026; Bu et al., 2025), generalist or experience-based policy learning (Intelligence et al., 2025b;a; 2026), spatial reasoning (Chen et al., 2025; Zhang et al., 2025), and cross-embodiment or language-action transfer (O'Neill et al., 2024; Zheng et al., 2026; Zha et al., 2026). A second line targets language-following failures from modality imbalance using vision-only penalties (Lian et al., 2026), affordance-prior subtraction (Zhan et al., 2026), counterfactual guidance (Fang et al., 2026), or attention recalibration (Zhang et al., 2026b). These works improve generalization or mitigate shortcut failures at the behavior level based on hypothesized failure mechanisms; we instead use mechanistic analysis to map task-critical information during successful task following and identify disruptions to information flow during compositional generalization failures.

3 VLA-PATCHLENS

In this section, we present VLA-PATCHLENS, a framework based on activation and path patching for analyzing task-critical representations inside a VLA. The pipeline is shown in Figure 2. Section 3.1 covers the VLA action-generation process and matched VLA run selection; Section 3.2 states the modeling assumption and uses activation patching to localize task-critical sites, followed by path patching for finer-grained analysis of information flow between specific source–receiver pairs, with shared rollout-based readouts.

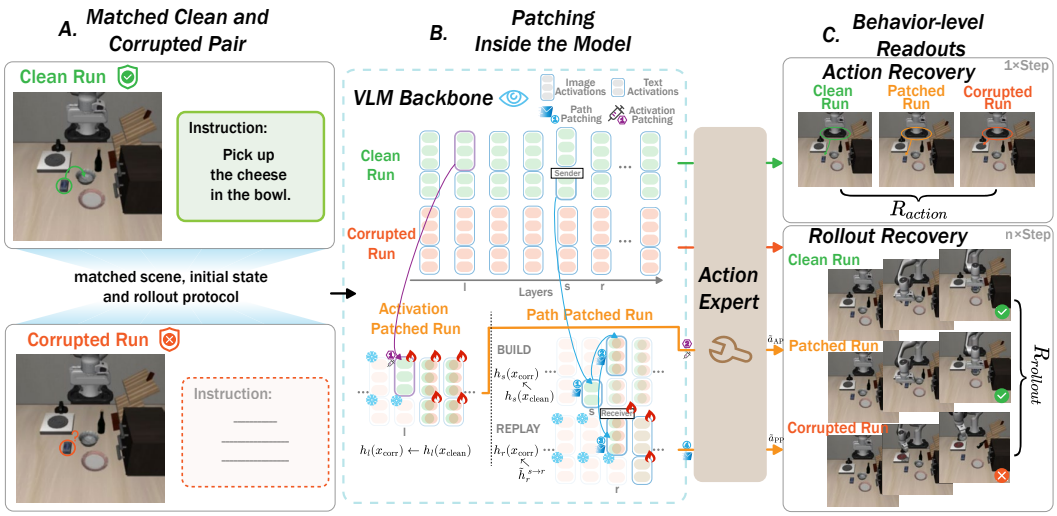


Figure 2: Overview of VLA-PatchLens. (A) Clean and corrupted pairs share the same task and layout but differ in instruction or task-following behavior. (B) Activation patching directly injects the clean-run activation of a layer-token group (e.g., the image tokens at layer 2) into the same location of the corrupted run and changes the action output. Path patching first injects the clean source activation and propagates it only along the selected path to update the receiver activation, then inserts the updated receiver activation into a second round of corrupted-run inference to change the action output. (C) Rollout behavior readouts measure R_{action} and R_{rollout}.

3.1 Matched VLA Runs

Policy Action Generation. We adopt $\pi_{0.5}$ (Intelligence et al., 2025b) as the main base policy. At inference time, it couples a VLM backbone with a transformer-based action expert. The VLM backbone takes encoded image and task-language inputs as prefix tokens; optional robot proprioceptive state is unused in our LIBERO environment (Liu et al., 2023). The action expert generates an action chunk through flow-matching denoising. The image-language prefix is computed once; at each denoising step, action tokens cross-attend to this cached prefix and are then updated. Thus, task information enters the VLA through task language and then passes into the backbone prefix. During denoising, action tokens repeatedly cross-attend to this prefix to finally deliver task information into action generation. We therefore use this prefix (i.e., inference-time backbone residual-stream activations) as the main object of analysis. We also analyze StarVLA-OFT (Community, 2026), which processes image, text, and action-query tokens in a shared VLM backbone with causal self-attention. An MLP head maps the action-query hidden states to a continuous action chunk.

Matched Clean-Corrupted Runs. Each matched clean-corrupted pair consists of two rollouts: `CleanTaskRun` for intended rollout behavior and `CorruptedTaskRun` for unintended rollout behavior. Runs within the same pair share the scene and rollout protocol, but differ in task language or policy behavior. We use three task suites, each targeting a different type of compositional generalization, and apply both analysis settings to each suite. In the successful task-following analysis, `CleanTaskRun` uses the full instruction, whereas `CorruptedTaskRun` uses blank filler; these runs share the initial state. In the generalization-failure analysis, `CleanTaskRun` is a successful rollout or a rollout that showcases task-language following, whereas `CorruptedTaskRun` is a rollout that fails to generalize compositionally under the same task instruction and layout.

3.2 Activation and Path Patching with Rollout-Based Readouts

Activation Patching. We use activation patching to localize layer-token sites that carry task-critical information. This causal intervention method (Heimersheim & Nanda, 2024; Goldowsky-Dill et al., 2023) replaces internal activations in a corrupted task run with those from a clean task run to observe behavioral changes. In our VLA context, we apply this technique to predefined token groups (e.g., image or text tokens).

Let $h_{\ell,g}(x)$ denote the residual-stream activation in the VLM backbone at layer ℓ for token group $g \in \{\text{img}, \text{text}\}$ during run x. For a matched clean-corrupted pair, we construct a patched corrupted run by replacing the corrupted activation $h_{\ell,g}(x_{\text{corr}})$ with the clean activation $h_{\ell,g}(x_{\text{clean}})$. We keep all other backbone groups unchanged, i.e., $\tilde{h}_{\ell,g} = h_{\ell,g}(x_{\text{clean}})$ and $\tilde{h}_{\ell,g'} = h_{\ell,g'}(x_{\text{corr}})$ for $g' \neq g$. The patched backbone prefix is then passed to the unchanged action expert.

Path Patching. This intervention (Goldowsky-Dill et al., 2023) tests whether task-critical information affects behavior through a single or multiple specified computational path. For a selected source and receiver, we propagate the clean source activation along this path while fixing off-path inputs to their corrupted-run values. We then insert the resulting receiver activation into an otherwise corrupted run. In our VLA context, sources and receivers are selected layer-token sites, and path effects are measured using the same R_{action} and R_{rollout} readouts. Formally, for a source site $s = (\ell_s, g_s)$ and a receiver site $r = (\ell_r, g_r)$ with $\ell_s < \ell_r$, the path-patched receiver activation $\tilde{h}_r^{s \to r}$ is computed by propagating the clean source activation $h_s(x_{\text{clean}})$ along the path $s \to r$ only, while every other input to r keeps its corrupted-run value. Replacing $h_r(x_{\text{corr}})$ with $\tilde{h}_r^{s \to r}$ gives the patched action $\tilde{a}_{s \to r}$, which enters the readouts below in place of $\tilde{a}_{\ell,g}$.

Rollout-Based Readouts. To evaluate these interventions in VLA control, we define two behavioral readouts.

The first readout is R_{action}. It measures the single-step prediction change, $R_{\text{action}}(\ell, g) = 1 - \|\tilde{a}_{\ell,g} - a_{\text{clean}}\|_2 / \|a_{\text{corr}} - a_{\text{clean}}\|_2$. Here, a_{clean}, a_{corr}, and $\tilde{a}_{\ell,g}$ denote the clean, corrupted, and patched action predictions. The term $\|\cdot\|_2$ denotes the L_2 distance. A value of 1.0 means the patched action perfectly matches the clean action, while 0 means no improvement over the corrupted baseline.

The second readout is R_{rollout}. It measures the absolute success rate improvement during closed-loop execution, $R_{\text{rollout}}(\ell, g) = \text{SR}(\tilde{\pi}_{\ell,g}) - \text{SR}(\pi_{\text{corr}})$. Here, π_{corr} and $\tilde{\pi}_{\ell,g}$ are the corrupted and patched rollout policies, and $\text{SR}(\cdot)$ represents the success rate. This readout verifies whether a local intervention improves multi-step task behavior.

4 Experiments

After the experimental setup (Section 4.1), we apply VLA-PatchLens in two settings: successful task following (Section 4.2) and compositional generalization failure (Section 4.3), and then extend it to another architecture and a real robot (Section 4.4).

4.1 Experimental Setup

Environment Setup. We conduct experiments with matched run pairs defined in Section 3.1. We mainly run with $\pi_{0.5}$ (Intelligence et al., 2025b) using four A6000 Blackwell GPUs in the LIBERO environment (Liu et al., 2023). We create three compositional generalization settings from LIBERO: paraphrase, seen-target recombination, and seen-goal recombination. Appendix C defines the suite construction for these settings, and we use the same task suites in Sections 4.2 and 4.3. For each task, we assign it a unique layout and run at least 15 rollout pairs, resulting in at least 450 rollout pairs in total. We additionally evaluate StarVLA-OFT (Community, 2026) on the same three generalization suites and $\pi_{0.5}$-DROID zero-shot on three tasks with a Franka robot (Section 4.4) to show the general applicability of our method.

Metrics. For activation and path patching, we use R_{action} and R_{rollout} from Section 3.2 to measure the amount of task-critical information (also referred to as information hereafter) from patched activations. For certain experiments, we also report success rate, activation similarity score, and attention scores between action, image, and text token groups. Appendix B defines the notation, token groups, and metric implementations used for these measurements.

4.2 Task-Critical Information Distribution During Successful Task Following

Research Questions. Our experiments focus on addressing the following research questions:
Q1. How is task-critical information distributed during successful task following? Does this distribution show a regular main pattern during inference-time forward passes?

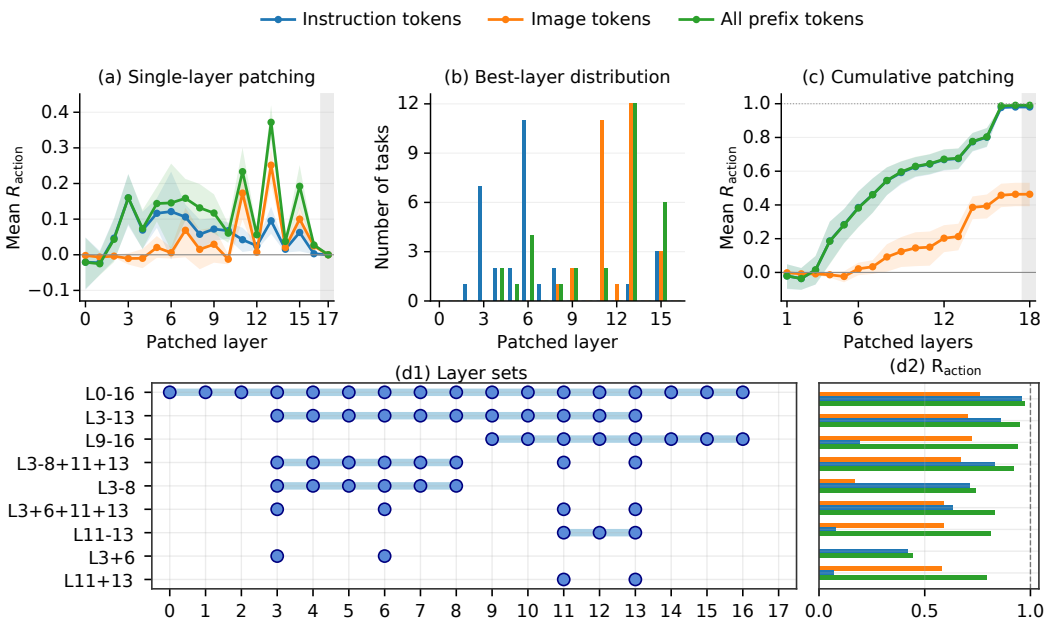


Figure 3: (a) Single-layer R_{action}. (b) Best-R_{action} layer distribution across 30 tasks. (c) Cumulative R_{action}, where k patched layers means patching layers 0 to $k-1$ together. (d1,d2) R_{action} for selected layer-patching combinations: (d1) marks the layers in each patched set, and (d2) reports the resulting R_{action} for each token group.

Q2. Besides the main pattern, does the VLA show more feature-specific patterns? Such as when the policy performs planning and how object-related critical information is distributed

Q1: Task-Critical Information Distribution. Q1 asks how task-critical information is distributed during successful task following. Figure 3 summarizes R_{action} for each layer when patching image, text, and their combination tokens. We primarily report the effects of patching at the beginning of the rollout. Appendix A reports patching effects at different simulator time steps during task execution for Figure 3(a,c). Appendix D provides per-suite results for Figure 3(a). Appendix F evaluates patching effectiveness, showing substantially higher R_{action} for matched clean activation patching than for random activation injection or activation injection from other tasks.

Findings: For text tokens, task-critical information is relatively evenly distributed across layers, and different layers provide complementary task information.

Figure 3(b) shows that the best-R_{action} layer varies across tasks rather than concentrating at a single layer. Figure 3(a) shows the mean R_{action} by layer, where text-token R_{action} remains evenly distributed. In Figure 3(c), cumulative R_{action} for text tokens closely overlaps with all-token R_{action} and shows a clear additive effect across layers, indicating that task information in different layers works together and is complementary; this complementarity is also reflected in Figure 3(d1,d2), where patching a few task-information-rich layers together yields high R_{action}. Appendix A and Figure 8 reinforce the effects in Figure 3(a,c) by showing the same pattern when patching at different simulator time steps.

Findings: Image tokens contain substantially more task information in later layers than in early layers, and this information comes from early text-token processing.

Figure 3(a) shows that image-token information is small in early layers but increases sharply around layers 11–15. In Figure 3(c), cumulative text-token R_{action} closely overlaps with cumulative all-token R_{action}, indicating that image tokens add little once sufficient text-token layers are patched and that image-token information comes from text tokens. And in Figure 3(a), at layers 11–17, patching image and text tokens still outperforms image-only patching, suggesting that image tokens mainly absorb earlier text information. Appendix A and Figure 8 reinforce this image-token effect across additional simulator time steps.

Figure 5(a) also supports this finding: in later layers, action-token attention shifts from text tokens to image tokens, and image-to-text attention also becomes stronger. More direct evidence comes from Figure 5(b), where path patching traces the text-to-image information flow and shows task-critical information moving from early text-token layers (around layers 1–6) to later image-token layers (around layers 11–13).

Findings: Action-loss and rollout recovery are highly consistent. Figure 4 shows a near-linear relationship between R_{action} and R_{rollout} across layer-set patching, supporting R_{action} as a lightweight metric for selecting patching sites before expensive closed-loop evaluation.

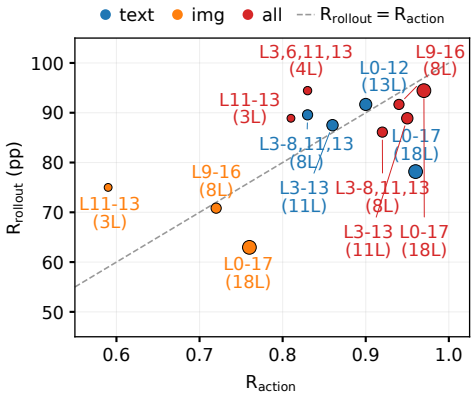


Figure 4: R_{action} vs. R_{rollout} for layer-patching combinations. Labels identify layer ranges; parentheses and marker size indicate patched-layer count.

Combining the findings above, we summarize the main task-information distribution pattern as follows: during a forward inference pass, the task-critical information first enters through the task language and is represented in text-token layers. Information from early text-token layers then moves into image tokens in later layers. Finally, action tokens in the action expert access task-critical information from all text-token layers and mainly from later image-token layers.

Q2: Feature-Specific Patterns. Q2 asks whether the VLA shows more feature-specific task-information patterns beyond the main information distribution pattern. In Appendix E, with Figure 10(a,b) and Table 1, we examine feature-specific cases: when the policy makes planning decisions, where object-specific information appears, and whether object text tokens alone explain task-language differences.

Findings: The policy makes planning decisions early in the rollout. Figure 10(a) shows patching results at different simulation time steps for matched runs that pick the same object but put it at different target objects. Earlier patching yields higher mean SR than later patching, suggesting that the model prefers to plan where to put the object near the beginning of the episode.

Findings: Object-specific information is distributed across different object-layer sites, and object text tokens alone are weak. Figure 10(b) shows object-dependent hotspots across several layers rather than one universal object layer. Table 1 shows that full text and scaffold text (full text token without obj-text token) reach 92% SR, whereas object text token-only intervention reaches 0% SR, showing that the object word matters in task language but is insufficient in backbone text tokens.

4.3 Task-Critical Information Distribution During Compositional Generalization Failure

Research Questions. Our experiments focus on addressing the following research questions:
Q3. Can VLA-PatchLens reveal and repair compositional generalization failures?
Q4. Does activation similarity directly correlate with SR or R_{action}?

Q3: Generalization Patching. Q3 asks whether activation patching and path patching can reveal and repair compositional generalization failures. In Figure 6, after applying source-to-receiver path patching across token-group combinations and layers, we summarize five major breaks in the task-critical information flow, identified by the consistently positive R_{action} after patching these paths.

Findings: In compositional generalization failures, we observe breaks in the task-critical information flow identified in Section 4.2. The broken paths align with the task-critical information distribution pattern found in Section 4.2: for example, information flows from text tokens at layer 6 into image tokens and then into the final actions.

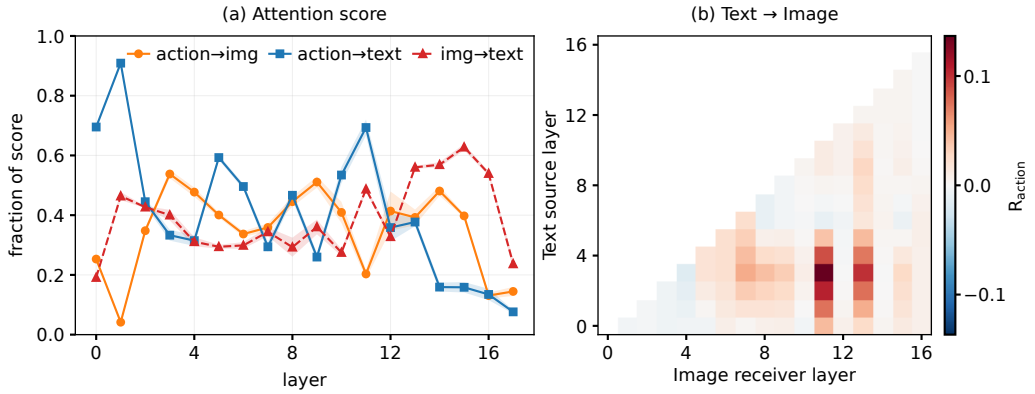


Figure 5: (a) Layer-wise attention over three task suites, showing each token group's attention fraction for action-to-image/text denoising attention and image-to-text backbone attention. (b) Source-to-receiver path-patching R_{action}, averaged over the three task suites; higher values indicate more direct task-critical information flow.

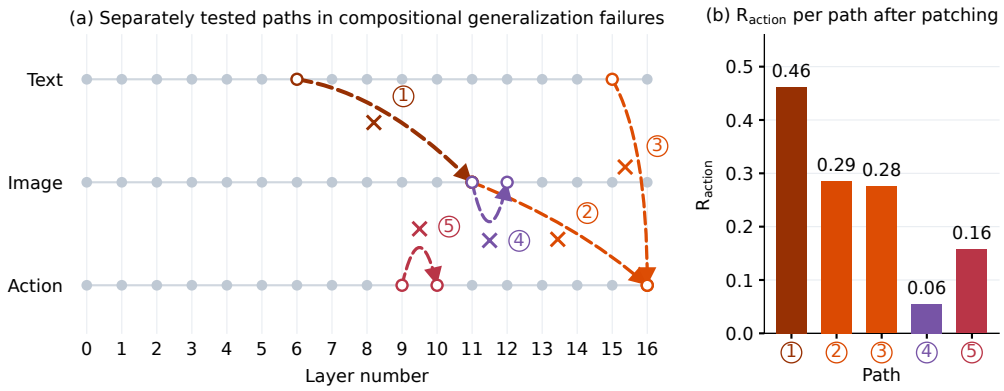


Figure 6: (a) Five source-to-receiver paths whose patching consistently has R_{action} across the three task suites in compositional generalization failures. Circled numbers denote the path index. Paths are tested separately; a path across layers combines all computational paths from its source to its receiver. (b) R_{action} after patching each path.

Findings: While full-scale path patching is computationally costly, simple single-layer activation patching produces R_{rollout} without reducing trajectory diversity, and R_{action} is highly consistent with R_{rollout} across layers. Figure 7(c1) shows that patching improves SR over the corrupted baseline across the task suites, while Figure 7(c2) shows that center-based trajectory diversity remains comparable rather than collapsing to a narrower trajectory distribution. Figure 7(a) also shows layers with stronger R_{action} tend to be the same layers that produce stronger R_{rollout}, supporting R_{action} as a lightweight proxy before closed-loop evaluation.

Findings: Different task suites show different best patching layers, but Layer 13, a high-information layer in Figure 3(a), provides consistent R_{rollout}. Figure 7(a) shows that the strongest single-layer patching effect differs across paraphrase, seen-target and seen-goal recombination failures. Meanwhile, Layer 13 yields a positive patching effect across suites, suggesting a shared patching option.

Q4: Similarity and R_{rollout}. Q4 asks if activation similarity correlates with patching R_{rollout}.

Findings: Overall R_{rollout} is not explained by activation similarity. Figure 7(b) shows that cosine-distance and relative-L2 activation-similarity scores follow similar trends. However, these trends do not clearly overlap with R_{rollout} or indicate causal increases or decreases in R_{rollout} in Figure 7(a). Thus, activation similarity is a weak proxy for patching R_{rollout}.

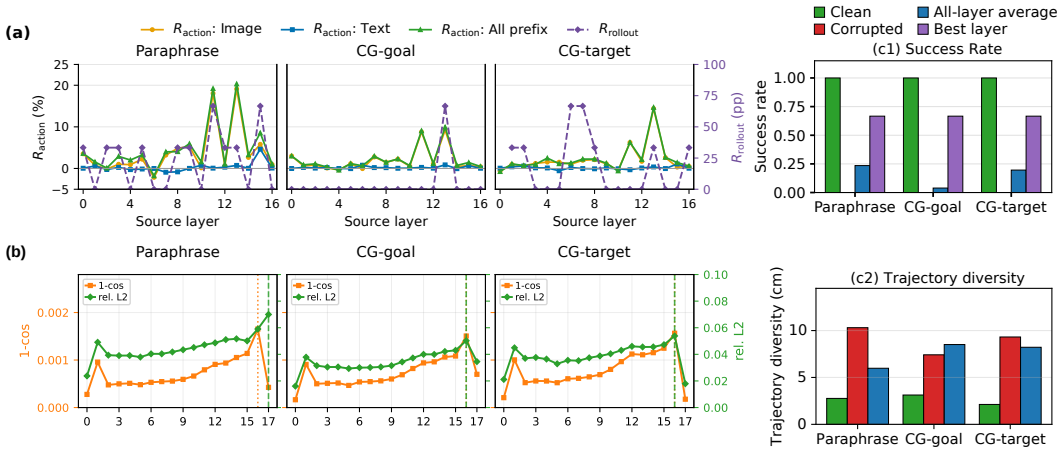


Figure 7: Compositional generalization (CG) failure patching summary. CG-goal and CG-target denote seen-goal and seen-target recombination. (a) Per-layer activation patching for the three task suites. (b) Per-suite cosine-distance and relative-L2 scores; lower values mean closer activations. (c) Per-suite rollout summary: (c1) mean and maximum SR after patching, where the best layer differs across task suites as shown in (a); (c2) center-based trajectory diversity. Appendix B defines the activation-similarity and trajectory-diversity metrics.

4.4 VLA-PatchLens for Other Architectures and Real-World Environments

We test whether VLA-PatchLens can analyze task-critical representations in a different architecture and in real-world manipulation. The full details are in Appendix G.

StarVLA-OFT. We apply the same failure-analysis protocol to StarVLA-OFT (Community, 2026) across paraphrase, seen-goal recombination, and seen-target recombination. Figure 12(a) shows stronger text- and all-token R_{action} in middle and later layers, with the broad pattern preserved from 0% to 80% of the simulation execution steps. Image-token patching yields near-zero R_{action}, consistent with the causal attention structure that prevents later task-language tokens from updating earlier image-token states. Single-layer patching also produces R_{rollout} across all three suites (Figure 12b). These results show that the intervention pattern depends on the policy architecture.

Real-Robot Evaluation. We evaluate VLA-PatchLens zero-shot with $\pi_{0.5}$-DROID on three real-world Franka tasks (Figure 13a), using the protocol of the successful task-following setting (Section 3.1). Full-instruction rollouts achieve task success, whereas blank-prompt rollouts fail. Figure 13(b) shows layer-wise R_{action}: text-token R_{action} is spread across the early and middle layers, whereas image- and all-token R_{action} stay near zero in early layers and peak around layer 12, retaining the broad layer-wise trend observed in simulation. All-token single-layer patching produces positive R_{rollout} (Figure 13c). This evaluation extends the diagnostic to physical sensing and actuation.

5 Conclusion

We presented VLA-PatchLens, an activation- and path-patching framework for analyzing task-critical information distribution and flow for compositional generalization. In $\pi_{0.5}$, complementary task information spans text-token layers; early text information concentrates in later image tokens and reaches actions through cross-attention. R_{action} provides a lightweight proxy for R_{rollout}. Across three suites, single-layer patching raises SR from 0% to 3.9–23.5% averaged over layers and 66.7% at the best layer; activation similarity remains uninformative. Experiments extend the analysis to StarVLA-OFT and real-robot manipulation. The framework requires matched clean runs for the same task and layout, and the located task-critical information patterns depend on the evaluated policies and settings. Future work includes obtaining clean runs via test-time training and extending the analysis to more general circuit-level tracing (Hanna et al., 2025).

AI USE STATEMENT

We used generative AI tools to assist with writing code and polishing the manuscript for grammar, clarity, and readability. The authors reviewed and tested the AI-assisted code and checked the edited text to ensure that it accurately reflected the intended technical content. The authors take full responsibility for the correctness of the code and the accuracy and integrity of the final manuscript.

REPRODUCIBILITY STATEMENT

We follow the reproducibility guidelines in the ICLR 2027 author guidelines. We will open-source the code, configuration files, and scripts needed to reproduce our results, including task-suite creation and model evaluation (this work does not train models), on platforms such as GitHub and Hugging Face as soon as possible.

REFERENCES

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2(5):6, 2023.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

Hugo Buurmeijer, Carmen Amo Alonso, Aiden Swann, and Marco Pavone. Observing and controlling features in vision-language-action models. *arXiv preprint arXiv:2603.05487*, 2026.

Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, et al. Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint arXiv:2510.13778*, 2025.

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

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.

Yu Fang, Yuchun Feng, Dong Jing, Jiaqi Liu, Yue Yang, Zhenyu Wei, Daniel Szafir, and Mingyu Ding. When vision overrides language: Evaluating and mitigating counterfactual failures in vlas. *arXiv preprint arXiv:2602.17659*, 2026.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

Bryce Grant, Xijia Zhao, and Peng Wang. Not all features are created equal: A mechanistic study of vision-language-action models. *arXiv preprint arXiv:2603.19233*, 2026.

Michael Hanna, Mateusz Piotrowski, Jack Lindsey, and Emmanuel Ameisen. Circuit-tracer: A new library for finding feature circuits. In *Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 239–249, 2025.

Bear Häon, Kaylene Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. *arXiv preprint arXiv:2509.00328*, 2025.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*, 2024.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Xinchen Jin, Aditya Chatterjee, Pranav Kumar, and Rohan Paleja. Event-grounded sparse autoencoders for vision-language-action policies. *arXiv preprint arXiv:2605.17204*, 2026.

Momin Ahmad Khan, Novak Boskov, Fatima M Anwar, and Manzoor A Khan. Controlling vision–language–action policies through sparse latent directions. In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025.

Chanyoung Kim, Minwoo Kim, Minseok Kang, Hyunwoo Kim, and Dahuin Jung. Libero-para: A diagnostic benchmark and metrics for paraphrase robustness in vla models. *arXiv preprint arXiv:2603.28301*, 2026.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Quanyi Li. Vlas are confined yet capable of generalizing to novel instructions. *arXiv preprint arXiv:2505.03500*, 2025.

Shijie Lian, Bin Yu, Xiaopeng Lin, Laurence T Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Cong Huang, and Kai Chen. Langforce: Bayesian decomposition of vision language action models via latent action queries. *arXiv preprint arXiv:2601.15197*, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Shi Liu, Kecheng Zheng, and Wei Chen. Paying more attention to image: A training-free method for alleviating hallucination in lvlms. In *European Conference on Computer Vision*, pp. 125–140. Springer, 2024.

Hong Lu, Hengxu Li, Prithviraj Singh Shahani, Stephanie Herbers, and Matthias Scheutz. Probing a vision-language-action model for symbolic states and integration into a cognitive architecture. In *2025 IEEE International Conference on AI and Data Analytics (ICAD)*, pp. 1–8. IEEE, 2025.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Miranda Muqing Miao, Subin Kim, Brandon Yang, and Lyle Ungar. Contrastive conceptor activation steering (coast): Unlocking vision-language-action models through hidden states. *arXiv preprint arXiv:2605.17144*, 2026.

Marco Molinari, Leonardo Nevali, Saharsha Navani, and Omar G Younis. Emergent world representations in openvla. *arXiv preprint arXiv:2509.24559*, 2025.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Haoyuan Shi, Xiancong Ren, Yingji Zhang, Qinfan Zhang, Jiayu Hu, Haozhe Shan, Han Dong, Jinpeng Lu, Yinda Chen, Yi Zhang, et al. Vla-trace: Diagnosing vision-language-action models through representation and behavior tracing. *arXiv preprint arXiv:2605.30117*, 2026.

Aiden Swann, Lachlain McGranahan, Hugo Buurmeijer, Monroe Kennedy III, and Mac Schwager. Sparse autoencoders reveal interpretable and steerable features in vla models. *arXiv preprint arXiv:2603.19183*, 2026.

Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 976–985, 2026.

Jinghan Yang, Yunchao Zhang, Wang Yuan, Haolun Wan, Jiaming Zhang, Zhengyang Hu, and Yanchao Yang. Tri-info: Generalizable, interpretable failure prediction for vla models via information theory. *arXiv preprint arXiv:2606.19998*, 2026.

Lihan Zha, Asher J Hancock, Mingtong Zhang, Tenny Yin, Yixuan Huang, Dhruv Shah, Allen Z Ren, and Anirudha Majumdar. Lap: Language-action pre-training enables zero-shot cross-embodiment transfer. *arXiv preprint arXiv:2602.10556*, 2026.

Zhihao Zhan, Yuhao Chen, Jiaying Zhou, Qinhan Lyu, Hao Liu, Keze Wang, Liang Lin, and Guangrun Wang. Stable language guidance for vision–language–action models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4137–4159, 2026.

Hanxin Zhang, Mingshuo Xu, Abdulqader Dhafer, Shigang Yue, Hongbiao Dong, and Zhou Daniel Hao. Embodied interpretability: Linking causal understanding to generalization in vision-language-action models. *arXiv preprint arXiv:2605.00321*, 2026a.

Ji Zhang, Shihan Wu, Xu Luo, Hao Wu, Lianli Gao, Heng Tao Shen, and Jingkuan Song. Inspire: Vision-language-action models with intrinsic spatial reasoning. *arXiv preprint arXiv:2505.13888*, 2025.

Ninghao Zhang, Bin Zhu, Shijie Zhou, and Jingjing Chen. Restoring linguistic grounding in vla models via train-free attention recalibration. *arXiv preprint arXiv:2603.06001*, 2026b.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

APPENDIX

A ADDITIONAL K-STEP RESULTS

K-step patching protocol. To test whether the main information-distribution pattern depends on the first simulator frame, we repeat the patching sweep at three simulator time steps: $K = 25$, $K = 50$, and $K = 100$. For each step, we patch the same token regions used in the main successful-task-following analysis: image tokens, instruction-text tokens, and all prefix tokens. We report the K-step sweep across three task suites.

Figure 8 separates two views of the same intervention sweep. The top row asks whether R_{action} accumulates as we patch progressively larger layer prefixes, $L_0, \ldots, L_k$.

The bottom row asks whether any single layer alone explains the R_{action} pattern.

Showing both views is useful because cumulative R_{action} tests whether information is distributed across layers, whereas single-layer R_{action} tests whether the effect is concentrated at a narrow intervention site.

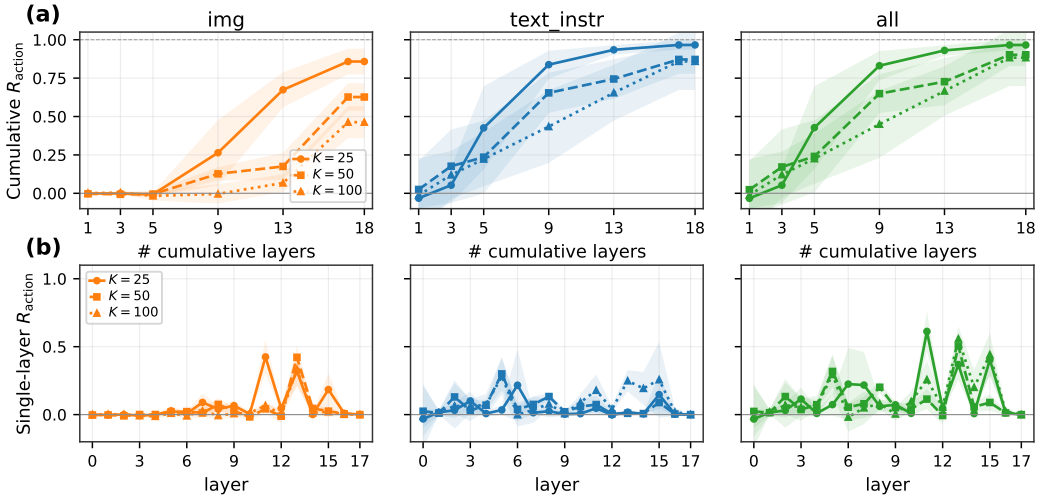


Figure 8: K-step activation-patching results for successful task-following rollouts. The top row reports cumulative $L_0, \ldots, L_k$ R_{action} for image, instruction-text, and all prefix tokens. The bottom row reports single-layer R_{action} for the same token regions. Solid, dashed, and dotted curves correspond to $K = 25$, $K = 50$, and $K = 100$, respectively.

Observation. Across all three simulator time steps, cumulative R_{action} from instruction-text tokens and all prefix tokens rises steadily as more layers are patched, while image-token R_{action} stays small in early layers and increases mainly after the middle layers.

The single-layer plots show that the exact peak layer can move with the simulator step, but the broader pattern remains stable: text-token information is distributed across layers, whereas image-token R_{action} is concentrated in later layers.

These results show that the main information-distribution pattern is not an artifact of patching only the first simulator frame.

B METRIC DEFINITIONS

Notation and setup. The following notation is for $\pi_{0.5}$. Unless stated otherwise, activations are the post-FFN residual stream of the backbone (PaliGemma) at layer $\ell \in \{0, \ldots, 17\}$. The backbone prefix spans 968 tokens: image tokens $[0 : 768]$ (three camera views, 256 patches each) and task-language tokens $[768 : 968]$. A patched token group g is the image group, the instruction span (the

instruction-only subrange of the language tokens, located by the common-prefix difference between the native and the empty prompt), or the full prefix $[0:968]$. The action chunk $a \in \mathbb{R}^{10\times 7}$ is the flow-matching denoising endpoint over 10 Euler steps, with 7 degrees of freedom per step (6 delta end-effector and 1 gripper) in unnormalized LIBERO units. The end-effector trajectory τ keeps only the xyz position and is resampled to $T = 100$ steps by uniform index subsampling, which is sufficient to cover task completion for most evaluated tasks.

R_{action}.

We patch one layer-token group: the clean-run activation at (ℓ, g) replaces the corrupted-run activation, and the patched prefix is decoded by the unchanged action expert into $\tilde{a}_{\ell,g}$. We then measure

$$R_{\text{action}}(\ell, g) = 1 - \frac{\|\tilde{a}_{\ell,g} - a_{\text{clean}}\|_2}{\max(\|a_{\text{corr}} - a_{\text{clean}}\|_2,\ 10^{-9})},$$

where a_{clean}, a_{corr}, and $\tilde{a}_{\ell,g}$ are the clean, corrupted, and patched action chunks, and $\|\cdot\|_2$ is the Euclidean norm over the flattened 10×7 chunk. A value of 1 means the patched action matches the clean action and 0 means no gain over the corrupted action; the score is not clipped and turns negative when the patch moves the action further from clean. Runs are deterministic under a fixed seed, so each pair yields one 18×3 layer-by-group matrix that we average across pairs.

R_{rollout} **and success rate.**

For N closed-loop rollouts with binary simulator success flags s_i, the success rate is $\text{SR} = N^{-1} \sum_{i=1}^{N} \mathbf{1}[s_i = 1]$, with $N = 15$ per condition.

We compute $R_{\text{rollout}}(i) = \text{SR}(\widetilde{\pi}_i) - \text{SR}(\pi_{\text{corr}})$ as a fraction. Best-layer summaries select $\arg\max_\ell \text{SR}(\widetilde{\pi}_{\ell,g})$ on the evaluation data and are descriptive.

For the reported R_{rollout} experiments, pairs are selected before intervention: the clean run succeeds and the corrupted run fails. We report every layer without filtering pairs by R_{rollout}, including ineffective and negative interventions. We aggregate per pair and then per suite.

Trajectory diversity. For a set of M end-effector trajectories $\{\tau_i\}_{i=1}^{M}$, each resampled to $T = 100$ steps, the center trajectory is $\bar{\tau}(t) = M^{-1} \sum_{i=1}^{M} \tau_i(t)$. We report the mean trajectory spread

$$100 \cdot \frac{1}{M} \sum_{i=1}^{M} \frac{1}{T} \sum_{t=1}^{T} \|\tau_i(t) - \bar{\tau}(t)\|_2$$

in centimeters, using only xyz positions.

Activation similarity score. To compare internal representations between successful and failed runs, we form one mean activation per outcome at each layer ℓ. Within a trial we average the post-FFN residual stream over denoising steps for all task-language tokens; we then average over successful trials to obtain u and over failed trials to obtain v, each flattened from a token-by-hidden-dimension matrix to a single vector. We report cosine distance $D_{\cos}(u, v) = 1 - u^\top v/(\|u\|_2\|v\|_2)$ and relative L_2 distance $D_{\text{rel}}(u, v) = \|u - v\|_2/((\|u\|_2 + \|v\|_2)/2)$. Both scores are computed in double precision with a 10^{-12} stabilizer in the denominator, then averaged per pair and per suite.

Attention score. For layer ℓ, head h, query token q, and key token k, the attention weight is $w_{\ell,h,q,k} = \text{softmax}_k(Q_{\ell,h,q} K_{\ell,h,k}^\top / \sqrt{d_{\text{head}}})$. The score from token group a to token group b sums the weight over keys in b and averages over heads and over queries in a:

$$A_\ell^{a\to b} = \frac{1}{|\mathcal{H}|\,|Q_a|} \sum_{h\in\mathcal{H}} \sum_{q\in Q_a} \sum_{k\in K_b} w_{\ell,h,q,k},$$

the mean fraction of attention mass that group-a queries place on group-b keys. Action-to-image and action-to-text scores use the action expert denoising attention, with the 10 action-token queries averaged over denoising steps, while image-to-text and text-to-image scores use the backbone prefix self-attention. We average each layer score over the three task suites and report the pooled mean and standard deviation.

C Compositional-Generalization Task Suites

Construction principle. Each compositional-generalization suite follows the matched-run protocol described in Section 3.1. The clean and corrupted runs share the scene, initial state, rollout budget, and patching protocol. They differ in whether the rollout follows the task-critical compositional cue or instead follows a shortcut behavior, such as preserving an old object-target association or using only part of the instruction.

Pair selection. For each suite, we retain matched pairs where the clean rollout expresses the intended compositional behavior and the corrupted rollout exhibits the corresponding compositional generalization failure. Activation patching then asks whether replacing one backbone layer-token group in the corrupted run with the clean activation can restore the intended behavior without changing the environment or the action expert.

Paraphrase. The paraphrase suite keeps the underlying task semantics fixed while changing the instruction wording. The corrupted rollout fails when the policy follows a shortcut tied to the training phrasing or ignores the task-critical wording change.

Seen-goal recombination. The seen-goal recombination suite reuses a familiar manipulation goal, such as laying an object down on a target surface, but pairs that goal with a different target object. This suite is shown as CG-goal in the figures.

Seen-target recombination. The seen-target recombination suite reuses familiar target objects but changes their goal relation, such as the target location or receptacle implied by the instruction. This suite is shown as CG-target in the figures.

D Per-Suite Patching Results

Figure 9 reports the single-layer and cumulative patching results separately for paraphrase, seen-goal recombination, and seen-target recombination. Each suite contains 10 tasks. We use the first-query observations and the same norm-input intervention as in Figure 3(a,c). Curves average R_{action} within each task and then weight tasks equally; shaded bands denote variation across tasks. The main figure averages the three suite means equally and computes its intervals by resampling tasks within each suite.

Image- and all-prefix single-layer R_{action} peak at source layer 13 in all three suites, whereas the strongest instruction-token layer varies across suites. Cumulative instruction-token R_{action} closely tracks all-prefix R_{action} and exceeds image-only R_{action} at later endpoints. These per-suite curves show that the aggregate pattern is shared across the three suites, despite differences in R_{action} magnitude and individual layer effects.

E Feature-Specific Patching Results

Patching timing. Figure 10(a) uses matched pick-and-put runs that pick the same object but place it on different targets, and patches all prefix tokens only during a window of action chunks. The left panels mark each window as a fraction of the pick phase and the put phase; the right panel reports the mean SR of each strategy against the clean and blank baselines. Windows in the pick phase (before pick and chunks 0–6) recover most of the SR, whereas windows in the put phase (after pick and chunks 6–16) recover almost none, so the target is decided before the object is lifted.

Object-specific information. Figure 10(b) patches one layer at a time from a successful full-instruction rollout into failed blank runs of the same scene, for six target objects, and colors each cell by the resulting SR. The effective layers differ across objects, and several objects have more than one effective layer, so object information is not confined to a single object layer. Table 1 isolates the text tokens at the object position: patching them alone gives an R_{action} of 0.04 and no successful rollouts, whereas patching the remaining scaffold tokens matches the full-text patch. The object word therefore matters in the instruction, but its effect is not carried by the object token positions in the backbone.

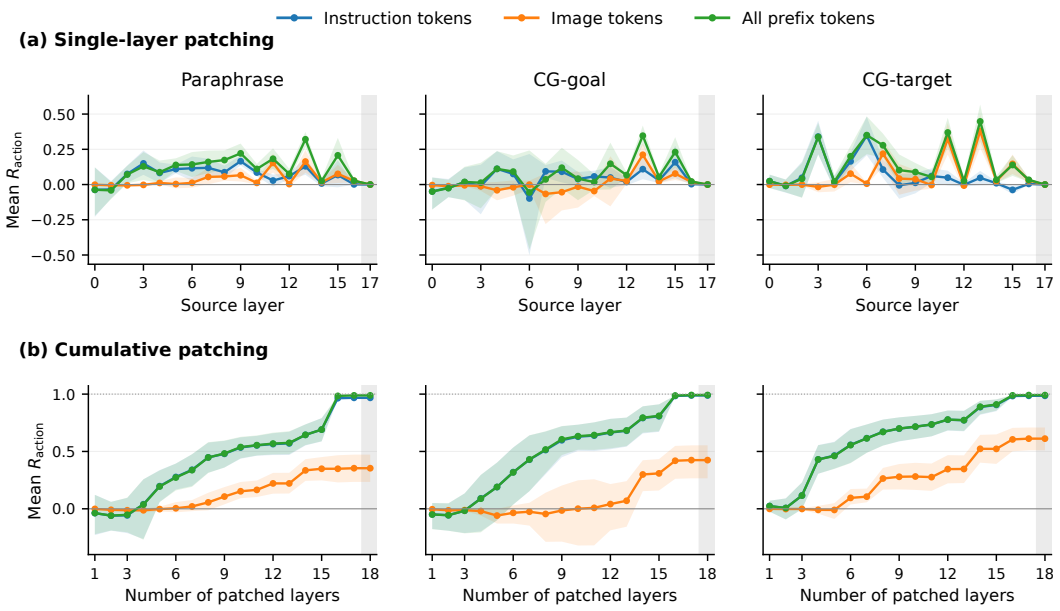


Figure 9: Per-suite R_{action} on successful rollouts. Columns show paraphrase, seen-goal recombination (CG-goal), and seen-target recombination (CG-target). (a) Single-layer norm-input patching. (b) Cumulative patching from source layer 0 onward. Colors identify instruction, image, and all-prefix token groups. Shading denotes variation.

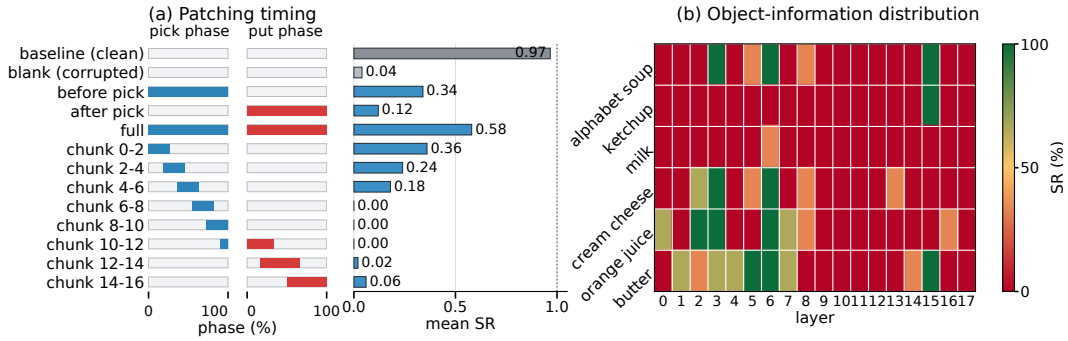


Figure 10: (a) Patching-timing result for pick-and-put examples. The left panel marks simulator-time-step patching windows in the pick and put phases, including chunk-index ranges; the right panel reports mean SR for each strategy. Earlier patching achieves higher SR. (b) Object-specific information distribution heatmap, showing where object-specific information is internally localized across tasks.

Table 1: Object text-token patching. Obj-only patches only the text token(s) at the target-object position, whereas scaffold patches all text tokens outside that position.

	blank	**full**	**obj-only**	**scaffold**
R_{action}	0.000	0.940	0.038	0.932
SR	12%	92%	0%	92%

F Patching Effectiveness and Control Experiments

Negative Controls. We evaluate a subset of clean–blank rollout pairs, selecting initial states where the clean rollout succeeds and the blank rollout fails. Random patches replace instruction-token activations with Gaussian directions rescaled to each clean token's activation norm, averaging over three random seeds. Cross-task patches use clean instruction-token activations from another task with a different prompt and observation, truncated to the shorter instruction span. This control changes both task and scene, rather than layout alone. Reverse patching replaces clean instruction-token activations with blank-run activations. All conditions use the same clean-reference score, so reverse-patching damage appears as a decrease from 1.

Figure 11(a) shows positive matched R_{action}, near-zero or negative cross-task R_{action}, and negative random R_{action}. The plotted matched curve is the all-prefix-token single-layer R_{action} from Figure 3(a), whereas the three controls patch instruction tokens on this subset, so the comparison holds neither the token group nor the pair set fixed. Random R_{action} reaches -2.86, below the displayed vertical range.

Fixed-Layer Reporting. Figure 11(b) reports R_{rollout} at every fixed layer for all pairs across the three suites, without excluding pairs by outcome (Appendix B).

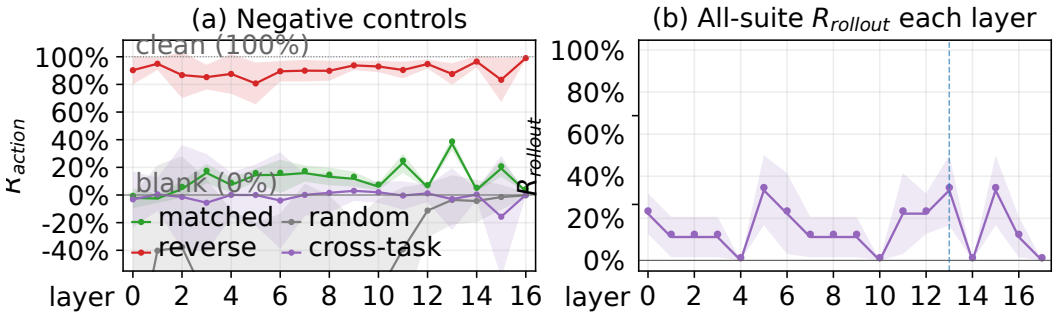


Figure 11: Control interventions and fixed-layer evaluation. (a) Matched all-token patching and instruction-token reverse, norm-matched random, and cross-task interventions. Shading denotes variation; parts of the random curve fall below the displayed range. Reverse patching is measured relative to the clean baseline of 1; the other interventions start from the blank baseline of 0. (b) R_{rollout} in percentage points at each fixed layer, averaged over the three suites of Figure 7(a); shading denotes variation.

G Cross-Architecture and Real-Robot Figures

This appendix collects the figures for the StarVLA-OFT and real-robot evaluations discussed in Section 4.4.

Real-robot tasks. Figure 13(a) shows the three tasks, each run zero-shot with $\pi_{0.5}$-DROID on a Franka arm with the instruction printed above the scene. In *pick the long bread and put on plate*, the robot grasps the elongated bread among distractor objects and places it on the plate. In *push the yellow button*, the robot presses the yellow button while nearby cups and bottles serve as distractors. In *put the white cup on the empty plate*, two plates are present and the robot must place the white cup on the plate that holds nothing. Following the successful task-following protocol (Section 3.1), the clean run uses the full instruction and the corrupted run uses a blank prompt with the same scene; only pairs whose clean run succeeds and whose blank-prompt run fails are kept, and Figure 13(b) reports layer-wise R_{action} over these pairs.

Real-robot R_{rollout}. For Figure 13(c), we run the policy closed-loop on the robot with the blank prompt while replacing the all-token activations at a single layer with the clean-run activations, and record whether the task is completed. $R_{\text{rollout}}(\ell) = \text{SR}(\tilde{\pi}_\ell) - \text{SR}(\pi_{\text{blank}})$ as in Section 3.2; the dashed lines mark the success rates of the full-instruction (clean) and blank-prompt rollouts, and since blank-prompt rollouts never succeed, each bar equals the success rate of the patched rollouts at that layer. We report the five patched layers evaluated on the robot.

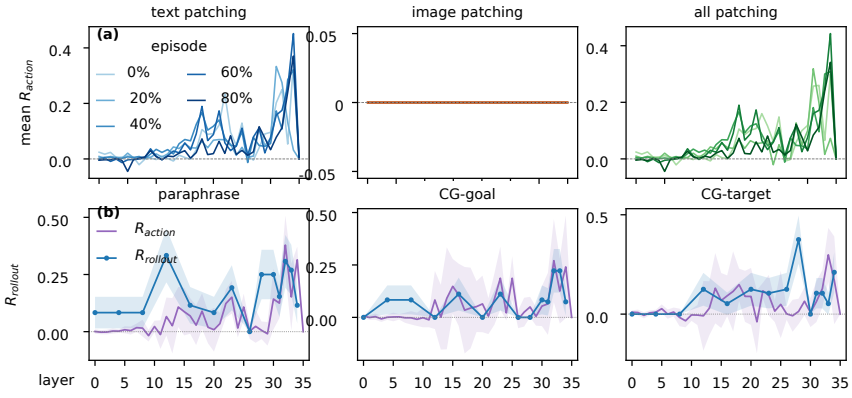


Figure 12: StarVLA-OFT extension. (a) Layer-wise R_{action} at 0%, 20%, 40%, 60%, and 80% of the simulation execution steps. (b) R_{action} and R_{rollout} across the three generalization suites.

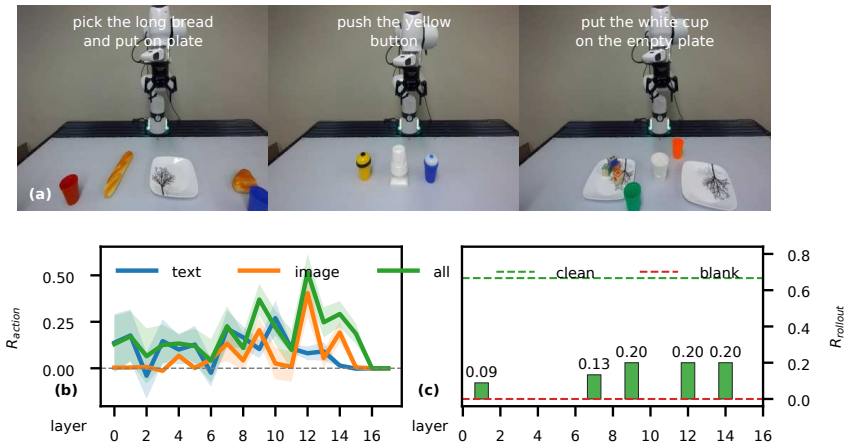


Figure 13: Real-robot evaluation with $\pi_{0.5}$-DROID on a Franka robot. (a) Three tasks. (b) Layer-wise R_{action}. (c) R_{rollout} for selected all-token single-layer patches with clean and blank baselines.