# VLA MODELS INHERIT THE MEANING OF WORDS, NOT THE USE OF THEIR REASONING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Vision-language-action (VLA) models are built on language models so that robots can use their reasoning. Acting on a reasoned instruction requires deriving the goal, setting it as the goal of the action and executing the movement; chain-of-thought policies, hierarchies and language-preserving training seek to connect these steps, but task success alone cannot show which one fails. We identify the variable through which language sets the goal: a low-dimensional *goal code*, inherited from the parent language model, that the action reads. In OpenVLA and $\pi_{0.5}$, writing the code redirects the robot and deleting it removes the goal, while a matched random subspace does neither, so the code tells whether a computation reaches the action. By this test, VLA models can reason but cannot act on it. Text-restored OpenVLA policies answer a rule correctly when asked in text, yet told the rule as their instruction they leave the bowl in place (0 and 1 of 30 episodes): the robot's input does not engage the computation. That text answer, written into the goal code of the same policy while it hears a plain instruction, drives it to the rule's destination in 24 and 26 of 30. Across ten models, word meaning reaches the code but a destination selected by an in-context rule does not; a stated goal moves the action as far as training made the action depend on it; and conclusions delivered to the code drive both families (22 to 28 of 30 episodes). VLA models inherit the meaning of words, not the use of their reasoning.

## 1 INTRODUCTION

A vision-language-action (VLA) model is a language model with a body: OpenVLA (Kim et al., 2025) predicts discretized actions through the vocabulary head of Llama-2 (Touvron et al., 2023), and $\pi_{0.5}$ (Physical Intelligence et al., 2025) attaches a flow-matching action expert to a PaliGemma backbone (Beyer et al., 2024). The case for building robots this way, demonstrated by RT-2 (Zitkovich et al., 2023), is that such a policy can follow instructions beyond the commands it was trained on. Consider "red means plate, blue means stove, green means cabinet; the marker is blue; put the bowl where the marker says." Acting on it takes three steps: deriving the destination, setting it as the goal of the action, and executing the movement.

Three lines of work try to carry reasoning into the action: policies that write their reasoning before acting (Zawalski et al., 2025; Yin et al., 2025; Mazzaglia et al., 2026), hierarchical systems in which a high-level model reasons in text and hands the policy a command (Shi et al., 2025; Physical Intelligence et al., 2025), and training recipes that protect the backbone's language ability (Driess et al., 2025; Hancock et al., 2026). These approaches connect reasoning to control in different ways, but task success alone cannot show which step fails. Policies often ignore the instruction: their success changes little when the language is varied (Fei et al., 2025) and persists when it is corrupted (Zhou et al., 2025). A policy that fails the rule above may be unable to derive the destination, may derive it without setting it as the goal, or may be unable to execute the movement; telling these apart requires the variable through which language sets the action's goal, which has not been identified. Three observations in one simulated kitchen show what is at stake. A policy trained to state its goal before acting, made to write "target: plate" while instructed to use the cabinet, puts the bowl on the cabinet. OpenVLA, told to put the bowl on "the appliance used for cooking," succeeds in about half of the episodes. Told the rule above, it goes nowhere, although its parent language model, given the same rule with a few worked examples, answers correctly 33 times in 36.



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.

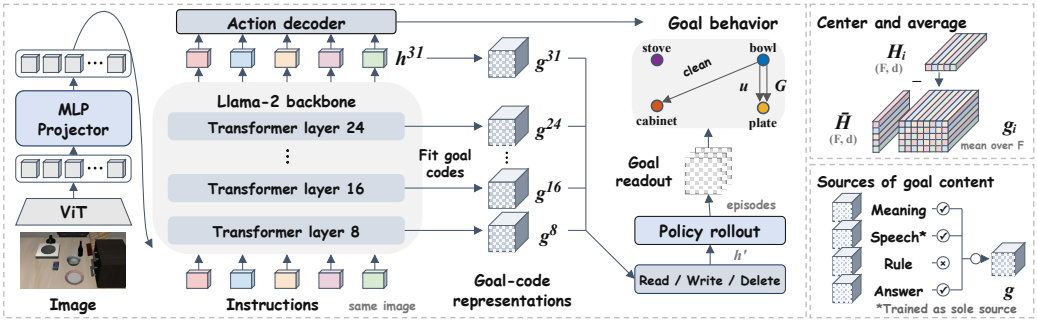



Figure 1: **Reading, writing and deleting the goal code.** OpenVLA is shown; pastel blocks are instruction variants on the same image. Centering residual states over instructions and averaging over frames yields $g_i$ (Eq. 1). We read the code, write along a destination direction $u$ or within the goal plane $G$, and delete $G$, then test behaviour in closed loop. Right: sources of goal content. Word meaning and delivered answers reach the code; in-context rule selection does not.

We identify this variable and use it to locate the failure. It is a *goal code*: a low-dimensional code in the backbone's residual stream, fitted as the mean difference between instructions on the same camera frames. Reading it tells where the robot will go, writing it redirects the robot, and deleting it removes the goal, while a matched random subspace does neither. A computation therefore reaches the action when it moves the code, and writing a conclusion into the code tests its execution on its own (Figure 1).

VLA models can reason but cannot act on it. Pairing robot training with text restores OpenVLA's ability to answer the rule when asked in text (0.92 to 1.00), yet told the rule as their instruction, the restored policies leave the bowl in place (0 and 1 of 30 episodes) and their goal code stays at chance. Their answer to the text question, written into the goal code while the policy hears a plain instruction naming another destination, drives them to the rule's destination in 24 and 26 of 30 episodes, as often as the correct code: the network derives the destination when asked and executes it when given, but the robot's input does not engage the derivation. The pattern holds across ten models. Word meaning reaches the code, even through descriptions that never name the destination, while a destination selected by an in-context rule stays at chance; a stated goal moves the action as far as training made the action depend on it; and conclusions placed on the code drive both families to within six episodes of an oracle.

These results tell each line of work where reasoning is lost. A chain of thought changes the action only when training makes the action depend on it; restoring a backbone's language ability restores what the network can derive, not what the action uses; hierarchies work because they deliver a conclusion to the goal; and a robot's goal code, not its statements, tells what it will do. Our contributions are:

- a goal code, readable, writable and selectively necessary, with additive slots inherited from the language model, that tests whether a computation reaches the action (§3);
- evidence that VLA models can reason but cannot act on it, for inferences from the instruction (§4), for the network's own conclusions (§5) and for stated chains (§6);
- evidence that delivered conclusions reconnect reasoning and action in both families, which explains why hierarchies work (§7).

## 2 Related work

**Reasoning for robot policies.** Chain-of-thought policies write a plan or a subtask before acting and report higher success (Zawalski et al., 2025; Yin et al., 2025; Mazzaglia et al., 2026), but where the gains come from is debated: generating reasoning in training improves the policy's representations (Chen et al., 2025a), HyT retains gains without generating a chain at inference (Mazzaglia et al., 2026), and in DeepThinkVLA's dynamics-shift tests, CoT gains require outcome-based training (Yin et al., 2025). Hierarchical systems pass a high-level model's conclusion to the policy (Shi et al.,

2025; Physical Intelligence et al., 2025), and training recipes protect the backbone's language ability (Driess et al., 2025; Hancock et al., 2026). Task success alone does not locate the failure; our test shows where reasoning stops (§5), when a chain is decorative (§6) and why hierarchies work (§7).

**Whether stated reasoning is used.** In language models, stated reasoning need not faithfully explain an answer (Turpin et al., 2023; Lanham et al., 2023; Chen et al., 2025b). Reasoning VLAs give mixed answers: a chain that contradicts the instruction can hijack the action (Huang et al., 2026), while in another policy substituting object names in the chain lowers success while rewriting its reasoning has little effect (Trinh et al., 2026). These studies compare policies that differ in architecture, data and training; ours differ only in how often the instruction was visible in training, which reads the mixed answers as training demand: a chain governs the action as far as training made the action depend on it (§6).

**Task variables inside networks.** Vectors extracted from hidden states encode tasks and steer behaviour in language models (Todd et al., 2024; Hendel et al., 2023; Rimsky et al., 2024); causal abstraction aligns such variables with distributed representations (Geiger et al., 2021; 2024), also across networks (Grant, 2025); causally effective task vectors need not be invariant across input formats (Opiełka et al., 2026). In VLAs, averaged instruction states can be recombined to execute unseen tasks (Li, 2025), directions found through the token embedding or by sparse autoencoders steer motion (Häon et al., 2025; Swann et al., 2026), and activation injection shows that, on the benchmarks tested, language matters when several goals share a scene (Grant et al., 2026). These studies show that such variables steer a policy or that language is used; we ask which computations over language reach the variable the action reads, and add selective necessity, additive structure and inheritance from the parent language model.

**Workspace and in-context computation.** Language models keep the results of some computations in a workspace of directions readable through the vocabulary, engaged when a result must be handed to a context-specified computation and bypassed by automatic processing; its authors conjecture that it is verbal because the output is verbal (Gurnee et al., 2026). Induction heads are a candidate mechanism for in-context retrieval (Olsson et al., 2022); its emergence also depends on the training distribution (Chan et al., 2022). Latent multi-hop reasoning occurs for some relation types (Yang et al., 2024b) but often starts its second hop too late (Biran et al., 2024). We carry these questions to models whose output is an action.

## 3 The action takes its goal from a code inherited from the language model

**Setting and models.** LIBERO-goal (Liu et al., 2023) places ten tasks in one kitchen, so the policy must read the instruction to know the goal. Our main arena is the bowl and its three destinations, the plate, the stove and the top of the cabinet; structure tests use a grid of two objects and four destinations. An episode is *completed* when the instructed goal's predicate fires within 300 steps and *redirected* when the object ends on a counterfactual destination; each closed-loop comparison runs in one job with its controls (Appendix I). The OpenVLA lineage is Llama-2-7B, Prismatic-7B (Karamcheti et al., 2024), OpenVLA-7B trained on the Open X-Embodiment mixture ("OXE") (Open X-Embodiment Collaboration, 2024) and its LIBERO-goal fine-tune; the $\pi$ lineage is Gemma-2B (Gemma Team, 2024), PaliGemma-3B, $\pi_{0.5}$-base and its LIBERO fine-tune. Matched policies are fine-tuned from the OpenVLA LIBERO-goal checkpoint on the same 52k frames for 6,000 steps (Appendix B).

**The goal code.** We compare the backbone's states under different instructions on the same frames. For instruction $i$, layer $\ell$ and position $p$,

$$g_i(\ell, p) \;=\; \mathbb{E}_x[h_{\ell,p}(x, i)] \;-\; \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}_x[h_{\ell,p}(x, j)]\,, \tag{1}$$

where $h_{\ell,p}(x, i)$ is the residual stream for frame $x$ and instruction $i$, the expectation runs over demonstration frames and $n = 10$; frame content cancels, so $g_i$ is what instruction $i$ adds. The codes

sum to zero and span a *goal plane* $G_\ell$ of dimension $n-1$. Positions are fixed in advance (OpenVLA: the destination noun and the last token before the first action token; $\pi_{0.5}$: the instruction suffix; language models: the end of the prompt); codes are fitted on training frames and every outcome is measured on held-out frames or episodes (Appendix A). *Reading* decodes a state as its nearest code, after centring a new sentence on the same-frame mean of its *family*, the sentences that share its wording and differ only in the goal. *Writing* sets the state's coordinate along $u = g_{\text{donor}} - g_{\text{main}}$ to the donor's value, or replaces its projection on $G_\ell$, at every control step over a band of layers; *deleting* projects $G_\ell$ out. Each operation is paired with a random direction of the same norm, a random subspace of the same size (ten directions, the basis used for the plane; Appendix A) and, where available, the plane of another variable.

The action reads its goal from this code. In OpenVLA the vocabulary cannot tell the instructed destination from the counterfactual one at any action position or depth, while the same states decoded by nearest code give the pursued goal at 0.97 to 0.99 (Table 1). Writing the code difference at layers 8–16 redirects 18 of 20 episodes and a random subspace of the same size none; the goal is taken by layer 16, since replacing the state at layers 16–24 no longer redirects. Deleting the goal plane leaves a robot that moves normally but never approaches the bowl, while a random subspace of the same size leaves 8 of 10 episodes completed. In $\pi_{0.5}$ the code difference redirects 10 of 10, and the goal is taken by layer 4: deleting the plane at layers 0–4 drops three tasks to 0, 1 and 5 of 10, while deleting it later, or deleting the plane of the object's identity, does not. The code decides behaviour where training made the instruction necessary (Appendix F), as in LIBERO-goal, which all tests below use. A computation therefore reaches the action when it moves the goal code where the action takes its goal (OpenVLA: last token, layers 8–16; $\pi_{0.5}$: instruction suffix, layers 2–10) and behaviour follows.

Table 1: **Goal-code readout and interventions.** Readout reports accuracy (OpenVLA: first action position, layers 10–24; $\pi_{0.5}$: instruction suffix, layers 9–16); writes (OpenVLA: destination noun, layers 8–16; $\pi_{0.5}$: layers 0–4) and transfer to an untrained LIBERO-90 kitchen count redirected episodes; deletion counts completed episodes, for three $\pi_{0.5}$ tasks. Controls in gray.

| Test | Readout / intervention | OpenVLA | $\pi_{0.5}$ |
|---|---|---|---|
| Read | Goal code (chance 0.10) | 0.97–0.99 | 1.00 |
| | Vocabulary (chance 0.50) | 0.40–0.60 | – |
| Write | Code difference | **18/20** | **10/10** |
| | Span of the ten codes | 17/20 | 8/10 |
| | Random, ten directions | 0/20 | 0/10 |
| | Vocabulary exchange | 0/10 | – |
| Delete | Goal plane | **0/10** | **0/10, 1/10, 5/10** |
| | Random, ten directions | 8/10 | 9/10, 10/10, 10/10 |
| | Object-identity plane | – | 9/10, 8/10, 9/10 |
| Transfer | Code difference | – | 13/20 |
| | Literal instruction | – | 8/20 |

The code carries word meaning inherited from the language model, not a task index: fitted on five cells of the object–destination grid, $g(o,d) \approx \mu + a_o + b_d$ predicts the untrained cells (2.7 to 3 of 3) in all four models tested, and the slots are written separately. Writing the wine bottle's code into the object noun of "put the bowl on top of the cabinet" makes $\pi_{0.5}$ grasp the bottle in 20 of 20 episodes and OpenVLA in 17 of 20, while the same code in the destination slot leaves the bowl as the object; a written plate code follows the plate when the plate is moved (Appendix D). Llama-2 has the same code and structure, and the difference it computes between two instructions, added to OpenVLA's states, redirects 17 of 20 episodes (random: none). What training changes is the format. From the Jacobian lens $J_\ell = \mathbb{E}[\partial h_{\text{final}}/\partial h_\ell]$ of Gurnee et al. (2026) we take $V_r(\ell)$, the top-$r$ right-singular subspace of $W_U J_\ell$ over the word rows of the vocabulary, and call it a *privileged carrier* when an edit's component in $V_r$ changes the output and its projection on a random subspace of the same rank does not. $V_{256}$ is a privileged carrier of the destination in Llama-2 and Prismatic (38 and 35 of 48 answers flipped; random: none) and in OpenVLA at layers 0–8, but not at layers 8–16, where the goal is taken (Table 7). The policy inherits the content of the code, not its word format.

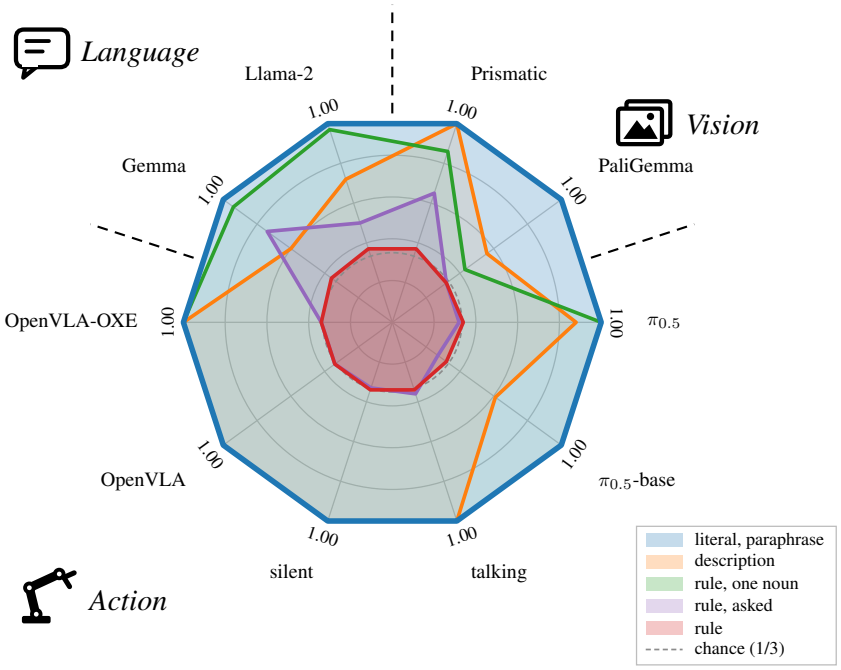



Figure 2: **What each instruction form puts on the goal code, across the lineage.** Each axis is a model, from the parent language models through the vision-language models to the policies; each polygon is a form, read on the goal code on held-out frames (accuracy, 0 at the centre and 1 at the rim; dashed ring: chance, $1/3$). Rule, one noun keeps the rule's wording with a single destination noun; rule, asked reads the next word after an explicit question. All values, negation included, are in Figure 7 and Tables 18 and 19.

## 4 Word meaning reaches the code; reasoning does not

We apply the test to five instruction forms, each written for all three destinations (Appendix C): a literal instruction, a paraphrase, a description that never names the destination, a rule that selects one of three named destinations by a marker, and a negation in both word orders; controls keep the rule's wording with two nouns or one. We read each form on the goal code in ten models of both lineages (Figure 2) and over depth in Llama-2 and OpenVLA (Figure 3), and run the forms in closed loop in four policies (Table 18). Without the destination noun, every OpenVLA policy puts the right destination on the code (1.00), and $\pi_{0.5}$ does so at 0.88; in OpenVLA the destination is there from layer 4, in Llama-2 only from layer 20. A rule reaches the code in none of the ten models, and the failure is the selection, not the sentence: with one noun the same wording reads at 0.86 to 1.00 in every model but PaliGemma, and with two it is at chance again. Under a rule, OpenVLA's state points to the first, second and third noun in 0.44, 0.39 and 0.17 of cases, tracking where the nouns sit rather than which one the rule selects, and it stays at chance at every layer, while in Llama-2 the rule rises above chance only after layer 18, to at most 0.50. Negation reaches the code through word order: OpenVLA's code follows the noun named last, pointing to the negated destination in 0.73 of cases when that noun comes last and to the target in 0.92 when the target does, and the $\pi_{0.5}$ code sits near chance for both orders. Only Prismatic, tuned on visual instructions, resolves negation on the code (0.92); its robot descendants do not.

Behaviour follows the code. Forms whose code is at chance fail in both families: under the rule, OpenVLA goes to no destination in 30 episodes and $\pi_{0.5}$ goes to the plate in 21, so its nine successes are all plate targets; under a negation with the negated noun last, OpenVLA puts the bowl on the negated destination in 8 of 18 episodes, as its code predicts. Reading the code thus separates failures that task success merges: no goal (rules), the wrong goal (negation), and the right goal with a shortfall downstream (descriptions, whose code is right in 0.87 to 1.00 of the reads taken during rollouts while half the episodes fail).

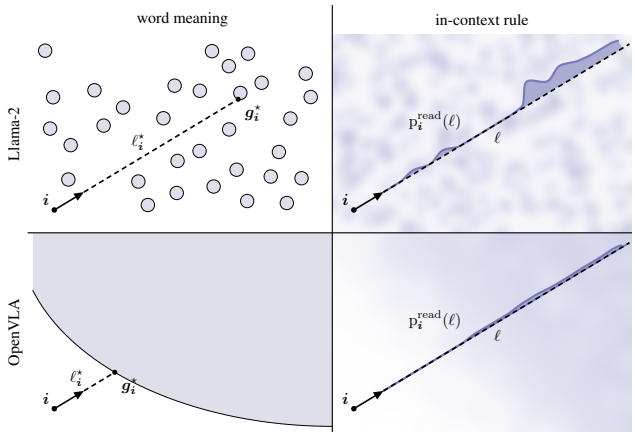



Figure 3: **Word meaning reaches the goal code early; in-context rule selection does not.** Rows: Llama-2, read at the prompt-ending colon, and OpenVLA, read at the last token before the first action token. Columns: descriptions and three-destination rules. Rays share a layer scale from 0 to 31. Left: the ray from instruction $i$ reaches the destination's code $g_i^\star$ at $\ell_i^\star$, the first measured layer with at least 90% correct reads (Llama-2: 20; OpenVLA: 4). Right: the profile is accuracy above chance, $p_i^{\text{read}}(\ell) = \max\{0, [a(\ell) - 1/3]/(2/3)\}$, interpolated between 17 measured layers. Particles, surfaces and textures are illustrative.

## 5 Reasoning the network computes does not reach the action

"The appliance used for cooking" names the stove through fixed word meaning; "the marker is blue" names it through a binding retrieved from the same sentence. Induction heads are a candidate mechanism for such retrieval (Olsson et al., 2022). Llama-2, Prismatic and Gemma perform this retrieval only when asked: over the read band their goal code at the end of the instruction is at chance (0.36 to 0.37), while their next word after "The robot should put the bowl on the" names the answer in 0.50 to 0.74 of cases, and no VLA answers even when asked (Table 19). Where the answer forms, it forms late: in a text question the destination code at the answer position is clearest at layer 24 in Llama-2 and at layers 20 to 22 in the restored policies below, while in the robot's format the same rule stays far below at every layer (Figure 4). The action takes its goal earlier, by layer 16 in OpenVLA and by layer 4 in $\pi_{0.5}$, before the layers at which the parents' verbal-workspace signature peaks (19–23 in Llama-2, 7 in PaliGemma; Appendix D).

Neither restoring the retrieval circuit nor training on rules brings the rule to the code. Llama-2 and Prismatic each have twelve induction heads, and in Llama-2 ablating the twenty strongest lowers the rule answer from 0.53 to 0.38 (twenty random heads: 0.48); OpenVLA and every policy we trained from it have none (Appendix G). Pairing every robot batch with generic text, copying and rule questions over unrelated nouns brings 20 to 22 heads back (Appendix B), and the restored policies answer rule questions in text on held-out markers and nouns at 0.90 to 0.96 while keeping their action accuracy, yet the rule on the goal code stays at chance, also when rule sentences are added to their robot data. Moved into the robot's instruction template, the rule is answered right only when the target is the first noun named (1.00, against 0.17 to 0.46), a preference Llama-2 does not show, and with the camera frame and the robot's question it falls to chance (Table 24): the ability is bound to the text format, as causally effective task vectors can be bound to their input format (Opiełka et al., 2026). Training on the sentences themselves installs operators, not in-context binding. With half of the bowl frames' instructions rewritten as negations, the code reads held-out negations at 1.00 and closed-loop success rises to 28 and 26 of 30 (two seeds); rewritten as rules, the policies complete 8 and 10 of 30 rule episodes, at chance. They exploit scene and arm-pose cues, a form of shortcut learning (Geirhos et al., 2020); removing these cues leads them to memorise training episodes (Appendix G).

Written into the code, however, the policy's own answer is carried out. The restored policies derive the rule's destination in a text question: read at layer 21 of the answer position, their state gives it for 31 to 36 of the 36 sentences used in closed loop. Told the rule as their instruction, the two policies trained on literal robot data leave the bowl in place in 30 and 29 of 30 episodes, and the two

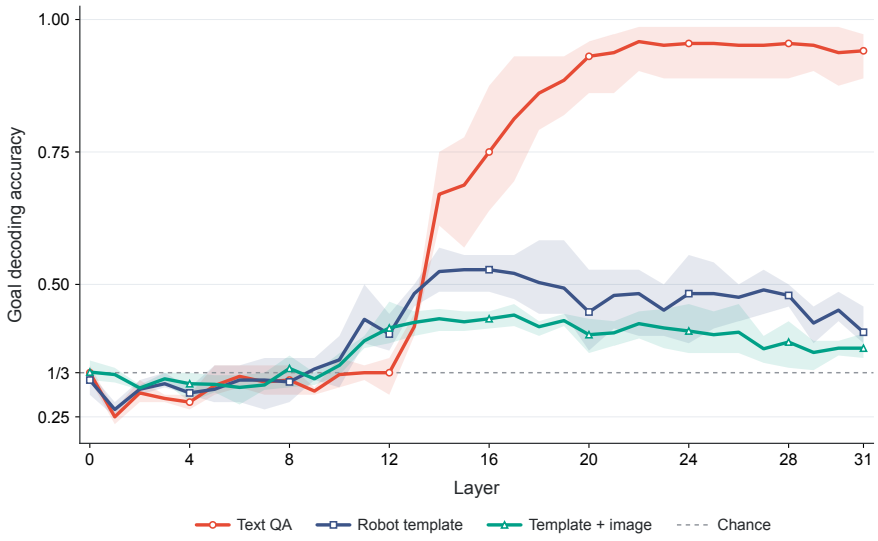



Figure 4: **Late goal decoding depends on the input format.** Mean destination-decoding accuracy at the answer position in four restored OpenVLA runs, from two recipes with two seeds each. Curves compare a text question (S1), the robot template (S2), and that template with an image (S3); shading spans the runs, and the dashed line marks chance ($1/3$). Table 24 gives the full comparison.

trained with rule sentences move it at chance. Decoded and written into the policy's own destination slot under a literal instruction naming another destination, the answer drives the robot to the rule's destination in 24 to 26 of 30 episodes, against 25 to 29 for the correct code and none for a random direction (Table 2); the policy's spoken answer, given as its instruction, does as well. OpenVLA shows the same for negation without restoration: its answer when asked, decoded in 11 of 12 held-out sentences and written into its own destination slot, completes 25 of 30 episodes against 16 when it hears the negation (random: 0; Appendix H). The network derives the conclusion when asked and executes it when given; the robot's input does not engage the derivation.

Table 2: **The text-restored policies carry out their own rule conclusion.** Source: the rule asked as a text question, answer spoken and decoded from the layer-21 state, of the 36 sentences used in closed loop. Closed loop: episodes completed on the rule's destination, of 30, when the policy hears the literal instruction, hears the rule sentence, or hears a literal instruction naming another destination with its own decoded answer written into the destination slot (decoded destination, or barycentric weights), with a random direction of the same norm, or with the correct code; last column: its spoken answer given as the instruction.

| | Source, of 36 | | Closed loop, of 30 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Policy | spoken | decoded | literal | rule | own answer written | random written | own weights written | oracle | spoken answer given |
| restored, seed 0 | 32 | 33 | 27 | 0 | **24** | 0 | 21 | 27 | 24 |
| restored, seed 1 | 36 | 36 | 27 | 1 | **26** | 0 | 27 | 26 | 27 |
| + rules, seed 0 | 32 | 31 | 30 | 12 | **24** | 0 | 23 | 29 | 26 |
| + rules, seed 1 | 35 | 36 | 29 | 5 | **25** | 0 | 24 | 25 | 28 |

## 6 Spoken reasoning moves the action as far as training made the action depend on it

A word the policy writes can change the action only by changing the goal code. Three policies, trained from the same checkpoint on the same data, write "task: put; object: bowl; target: ⟨goal⟩;

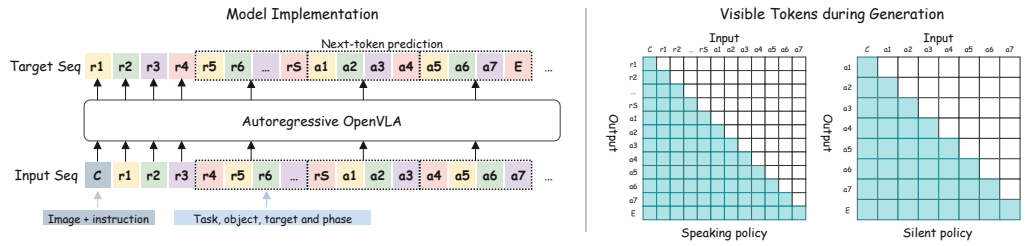



Figure 5: **A speaking policy writes its goal before it acts.** $C$ is the image and instruction context, $r_1, \ldots, r_S$ the written task, object, target and phase, and $a_1, \ldots, a_7$ the action tokens; $E$ ends the training target. Right: the context visible to each output (cyan) in a speaking policy and in the silent policy.

Table 3: **Withholding the instruction during training increases dependence on the spoken goal.** Instruction "put the bowl on top of the cabinet"; episodes redirected to the plate. Forced: the policy writes "target: plate" (two seeds; other columns seed 0). Spoken-word state: the written target's state replaced by its state under the donor target, whole or its component in $V_{64}$, $R_{64}$ or $V_{64}^{\perp}$; last row: the same operations on the silent policy's instruction noun.

| | | | Spoken-word state, layers 8–16 | | | | Whole state | |
|---|---|---|---|---|---|---|---|---|
| Policy | Instruction in training | Forced "plate" | whole | $V_{64}$ | $R_{64}$ | $V_{64}^{\perp}$ | layers 0–8 | layers 16–24 |
| talking | always | **0/20, 0/20** | 0/20 | 0/20 | 0/20 | 0/20 | 0/10 | 0/10 |
| half | half | 10/20, 11/20 | 0/10 | – | – | – | 6/20 | – |
| blind | never | **20/20, 19/20** | **19/20** | 0/20 | 0/20 | 19/20 | 19/20 | 0/20 |
| silent, instruction noun | | | 19/20 | 0/20 | 0/20 | 17/20 | 17/20 | 0/20 |

phase: ⟨reach | hold | release⟩. Action:" before every action (Figure 5) and differ only in how often the instruction was visible in training: always (*talking*), in half of the prompts (*half*) or never (*blind*), in which case the written target is the goal's only source. A *silent* policy trained without the line is the reference (Table 3). A spoken goal carries as much as training made it carry. Held at "cabinet" and made to write "target: plate," the talking policy redirects none of 20 episodes in either seed, the half policy 10 and 11, and the blind policy does what its line says. Where the written word carries the goal, it enters where the instruction noun does, at layers 0–16 and not later, and its word-aligned component carries no more than a random subspace; the phase word, which the camera also supplies, is inert even in the blind policy. The statement and the goal are computed separately: on descriptions such as "the appliance used for cooking," the talking policy's goal code reads the described destination at 1.00, but the target it writes names it only 32% of the time, since the line was trained to copy the instruction's noun.

A public chain-of-thought policy behaves like the talking one. The community ECoT checkpoint (Zawalski et al., 2025) writes a task, a plan, a subtask, a move and the visible objects before each action. On the two of 14 LIBERO-object scene pairs in which a changed instruction redirects it (14 of 20 episodes; screening fixed in advance, Appendix E), regenerating the whole chain under the other instruction redirects none of 20, rewriting the object's name in the chain changes nothing, and removing the chain leaves 18 of 20 completed. The action is sensitive to the chain a sixth as much as to the instruction; the chain is not the variable it reads its goal from.

## 7 Delivered conclusions reconnect reasoning and action

If reasoning is lost on the way to the goal, a conclusion delivered to the goal should be carried out, whoever computes it. We compute it with a language model, as a hierarchy would, and write it into the code of a robot that hears a literal instruction naming another destination; the rule sentence itself would stop OpenVLA even with the correct code written (Appendix H). Both families then do what

Table 4: **A language model's conclusion, delivered to the code, drives both families.** Episodes completed on the destination the sentence specifies, of 30. Written: Llama-2-7B's layer-15 answer, decoded against prototypes from sentences the robot never hears, is written into the destination slot (OpenVLA after layer 0; $\pi_{0.5}$ layers 0–9) of a robot hearing a literal instruction naming another destination. Given: the answer becomes the instruction. *All plate targets.

| | Rule | | Negation | |
|---|---|---|---|---|
| | OpenVLA | $\pi_{0.5}$ | OpenVLA | $\pi_{0.5}$ |
| robot hears the sentence | 0 | 9* | 16 | 18 |
| language model's conclusion, written into the code | **22** | **27** | **28** | **26** |
| language model's conclusion, given as the instruction | 24 | 25 | 27 | 28 |
| random direction of the same norm, written | 0 | 0 | 0 | 0 |
| correct code, written (oracle) | 28 | 28 | 28 | 25 |

the rule or negation specifies, within six episodes of the oracle, while a random direction of the same norm leaves the robot doing what it heard; the written code overrides the heard words in 23 and 26 of 30 rule episodes (Table 4). Giving the language model's answer as the instruction works as well (24 to 28 of 30), and this is the form in which a hierarchy delivers its conclusion: words whose meaning is what the goal code carries. A hierarchy never asks the action pathway to perform the inference, only to read word meaning, which the policy inherited from its language model.

## 8 Discussion

**Relation to the workspace account.** In the account of Gurnee et al. (2026), a verbal workspace serves context-specified computation and automatic processing bypasses it, and the authors conjecture that it is verbal because the output is verbal. Using their concept-coordinate exchange, we find no selective redirection of the tested VLA goals; the word-aligned subspace also loses its privilege where the action takes its goal. We interpret this pattern as consistent with their output-format conjecture. The context-specified computations the workspace supports in language models, such as retrieving a binding made in the sentence, are the ones we find do not reach the action.

**Conclusion.** VLA models inherit the meaning of words, not the use of their reasoning. The action takes its goal from a code inherited from the language model, which carries what words mean; reasoning over the instruction, even when the network performs it, does not reach that code; a stated chain reaches it as far as training made the action depend on it; and conclusions delivered to the code are carried out, whoever computes them. What robots that reason still lack is the connection from inference to goal. A backbone's score on text reasoning and a policy's task success do not measure it: restoring OpenVLA's text reasoning left its action untouched, and a policy that states the right goal need not pursue it. The goal code does: reading it where the action takes its goal, and writing a conclusion there, tests whether a chain-of-thought, hierarchical or language-preserving recipe connects its reasoning to the action.

### AI USE STATEMENT

In preparing the manuscript, we used generative AI tools mainly to polish the language and improve clarity and readability, with additional assistance in literature lookup, reference checking and formatting. These tools also assisted with implementing experiment and analysis code, providing feedback on experimental design, refining hypotheses and interpreting results. The authors reviewed the AI-assisted work and take responsibility for the methods, results and final manuscript.

### REPRODUCIBILITY STATEMENT

Appendix A gives the protocol for fitting, reading, writing and deleting the goal code, with the positions, layer bands, held-out splits and random controls; Appendix B the training recipe of every policy we trained; Appendix C the instruction forms and their templates; and Appendix I the software versions, the variation between repeated runs and the pairing of each comparison with its controls in one job. All models and the LIBERO benchmark are public.

## REFERENCES

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, et al. PaliGemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024. URL https://arxiv.org/abs/2407.07726.

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 14113–14130. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.781. URL https://aclanthology.org/2024.emnlp-main.781/.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 18878–18891. Curran Associates, Inc., 2022. doi: 10.52202/068431-1371. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/77c6ccacfd9962e2307fc64680fc5ace-Paper-Conference.pdf.

William Chen, Suneel Belkhale, Suvir Mirchandani, Karl Pertsch, Danny Driess, Oier Mees, and Sergey Levine. Training strategies for efficient embodied reasoning. In *Proceedings of The 9th Conference on Robot Learning*, volume 305, pp. 365–391. PMLR, 2025a. URL https://proceedings.mlr.press/v305/chen25a.html.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025b. URL https://arxiv.org/abs/2505.05410.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.02861.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025. URL https://arxiv.org/abs/2505.23705.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025. URL https://arxiv.org/abs/2510.13626.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 9574–9586. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/4f5c422f4d49a5a807eda27434231040-Paper.pdf.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. Finding alignments between interpretable causal variables and distributed neural representations. In *Proceedings of the Third Conference on Causal Learning and Reasoning*, volume 236, pp. 160–187. PMLR, 2024. URL https://proceedings.mlr.press/v236/geiger24a.html.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020. doi: 10.1038/s42256-020-00257-z. URL https://doi.org/10.1038/s42256-020-00257-z.

Gemma Team. Gemma: Open models based on Gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. URL https://arxiv.org/abs/2403.08295.

Bryce Grant, Xijia Zhao, and Peng Wang. Not all features are created equal: A mechanistic study of vision-language-action models. In *Multimodal Intelligence Workshop at ICLR*, 2026. URL https://arxiv.org/abs/2603.19233.

Satchel Grant. Model alignment search. *arXiv preprint arXiv:2501.06164*, 2025. URL https://arxiv.org/abs/2501.06164.

Wes Gurnee, Nicholas Sofroniew, Adam Pearce, Mateusz Piotrowski, Isaac Kauvar, Runjin Chen, Anna Soligo, Paul Bogdan, Euan Ong, Rowan Wang, T. Ben Thompson, David Abrahams, Subhash Kantamneni, Emmanuel Ameisen, Joshua Batson, and Jack Lindsey. Verbalizable representations form a global workspace in language models. *Transformer Circuits Thread*, 2026. URL https://transformer-circuits.pub/2026/workspace/index.html.

Asher Hancock, Xindi Wu, Lihan Zha, Olga Russakovsky, and Anirudha Majumdar. Actions as language: Fine-tuning VLMs into VLAs without catastrophic forgetting. In *International Conference on Learning Representations*, volume 2026, pp. 75333–75360, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/7a0f8055c838df8e62329a76c7c6403d-Paper-Conference.pdf.

Bear Häon, Kaylene Caswell Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. In *Proceedings of The 9th Conference on Robot Learning*, volume 305, pp. 2743–2762. PMLR, 2025. URL https://proceedings.mlr.press/v305/haon25a.html.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.624. URL https://aclanthology.org/2023.findings-emnlp.624/.

Zhengxian Huang, Wenjun Zhu, Haoxuan Qiu, Xiaoyu Ji, and Wenyuan Xu. TRAP: Hijacking VLA CoT-reasoning via adversarial patches. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2603.23117.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic VLMs: Investigating the design space of visually-conditioned language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 23123–23144. PMLR, 2024. URL https://proceedings.mlr.press/v235/karamcheti24a.html.

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. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270, pp. 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023. URL https://arxiv.org/abs/2307.13702.

Quanyi Li. VLAs are confined yet capable of generalizing to novel instructions. *arXiv preprint arXiv:2505.03500*, 2025. URL https://arxiv.org/abs/2505.03500.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Pietro Mazzaglia, Cansu Sancaktar, Markus Peschl, and Daniel Dijkman. Hybrid training for vision-language-action models. In *International Conference on Learning Representations*, volume 2026, pp. 135535–135556, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/db77389415a3e7aa66dd4f4fef384270-Paper-Conference.pdf.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022. URL https://arxiv.org/abs/2209.11895.

Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10611477. URL https://robotics-transformer-x.github.io/.

Gustaw Opiełka, Hannes Rosenbusch, and Claire E. Stevenson. Causality $\neq$ invariance: Function and concept vectors in LLMs. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.22424.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 54919–54933. PMLR, 2025. URL https://proceedings.mlr.press/v267/shi25d.html.

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. URL https://arxiv.org/abs/2603.19183.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=AwyxtyMwaG.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288.

Tuan Duong Trinh, Naveed Akhtar, and Basim Azam. Altered thoughts, altered actions: Probing chain-of-thought vulnerabilities in VLA robotic manipulation. *arXiv preprint arXiv:2603.12717*, 2026. URL https://arxiv.org/abs/2603.12717.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 74952–74965. Curran Associates, Inc., 2023. doi: 10.52202/075280-3275. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ed3fea9033a80fea1376299fa7863f4a-Paper-Conference.pdf.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a. URL https://arxiv.org/abs/2412.15115.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10210–10229. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.acl-long.550. URL https://aclanthology.org/2024.acl-long.550/.

Cheng Yin, Yankai Lin, Wang Xu, Sikyuen Tam, Xiangrui Zeng, Zhiyuan Liu, and Zhouping Yin. DeepThinkVLA: Enhancing reasoning capability of vision-language-action models. *arXiv preprint arXiv:2511.15669*, 2025. URL https://arxiv.org/abs/2511.15669v3. Version 3, revised August 2026.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270, pp. 3157–3181. PMLR, 2025. URL https://proceedings.mlr.press/v270/zawalski25a.html.

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. URL https://arxiv.org/abs/2510.03827.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

# A PROTOCOLS

**Fitting and reading the goal code.** Codes are fitted by Eq. 1 on demonstration frames and evaluated on held-out frames. For the scans of Table 1 we use 60 frames per instruction for fitting and 200 held-out frames for decoding over the ten LIBERO-goal instructions (chance 0.10). For the instruction forms of Figure 7 we use the first five steps of twenty demonstrations of each bowl task (60 frames, on which the image does not reveal the destination), 30 for fitting the three literal codes and 30 for testing (chance 1/3). A test sentence's state is centred on the same-frame mean of the three sentences of its family before it is compared with the codes. Read sites are fixed before any result: for the OpenVLA policies, the last token before the first action token, averaged over layers 8–16; for Llama-2 and Prismatic, the colon or the last token of the prompt, layers 8–24; for the $\pi$ lineage, the end of the prompt (Gemma-2B, PaliGemma) or the mean over the instruction suffix ($\pi_{0.5}$, $\pi_{0.5}$-base), layers 2–10. In closed loop the code is read every twenty control steps with the same centring (OpenVLA: last token, layer 16; $\pi_{0.5}$: instruction suffix, layer 8).

**Matched-state PCA.** Figure 6 uses Prismatic, OpenVLA and the silent and talking checkpoints on two objects crossed with four destinations. We use the prompt-ending colon for Prismatic and the last pre-action token for the policies, at layers 8, 10, 12, 14 and 16. The 60 shared frames are split into 30 even-indexed fitting frames and 30 odd-indexed display frames. Within each model and layer, we subtract the eight instructions' mean on the same frame, then divide by the root-mean-square norm of the eight fitting-condition means. A shared PCA is fitted to the resulting 160 condition means (four models, five layers and eight conditions). Its first two components retain 26.6% of their variance. Only held-out states are projected for display, giving 1,200 points per checkpoint and 4,800 overall. Frames recur across layers and instructions, so points do not count independent episodes. The full-dimensional tests of additivity are reported in Table 7.

**Writing and deleting.** All edits are applied at every control step. For a write with a donor instruction, the donor forward pass runs on the same frame and the two prompts are aligned from their end, so that the destination noun and the tail positions coincide. The one-dimensional write sets the coordinate along $u = g_{\text{donor}} - g_{\text{main}}$ to the donor's value and leaves the orthogonal part of the state unchanged; the span write replaces the projection on the goal plane; the whole-state replacement copies the donor state. Deletion projects the goal plane, or a subspace of it, out of the state. The plane's basis is the QR factorization of the ten codes at the layer. Because the codes are centred they sum to zero and span nine dimensions; their tenth singular value is at rounding level (below $10^{-3}$ of the ninth at every layer of both families), so the tenth basis vector is an arbitrary direction, and the span write and the deletion act on the nine-dimensional plane plus that direction. The random subspace of the same size has ten directions, drawn uniformly on the Grassmannian (a QR factorization of a Gaussian matrix, seeded per layer). The random direction has the norm of the real edit; in the multi-position writes of §7 it is scaled by one scalar per control step, the mean norm of the real edit over the edited positions, so that it does not concentrate where the real edit is large. In $\pi_{0.5}$, the codes contain 8 to 22% of their energy along one massive-activation dimension of PaliGemma, which is also the direction of the grand mean; deleting that dimension alone stops every task. Deletions in $\pi_{0.5}$ are therefore restricted to the instruction positions and use the goal plane with that dimension removed; the plane of a second variable (object identity) is prepared the same way.

**Word-aligned subspace.** The Jacobian lens (Gurnee et al., 2026) is fitted per model on a pretraining-like text corpus. $V_r(\ell)$ is the top-$r$ right-singular subspace of $W_U J_\ell$ restricted to the word rows of the model's vocabulary (31,744 for the Llama-2 lineage; the 256 action-token rows of OpenVLA are excluded). This fixed SVD subspace is our diagnostic; the original J-space is a union of cones formed by sparse, non-negative combinations of lens vectors. The vocabulary readout at a position ranks the candidate destination words by $W_U J_\ell h_\ell$. The destination test of Table 7 reads the language models on twelve instructions, the literal one and three paraphrases for each destination, each followed by "The robot should put the bowl on the"; the answer is the top-1 among the three destination words, and all twelve are answered correctly unedited. Of the pairs, 48 differ in destination (eight per ordered pair) and 12 share it; on the shared pairs none of the edits of Tables 7 and 11 moves the answer.

**Closed loop.** Episodes run in LIBERO with the official task predicates and a budget of 300 control steps. Each cell uses fixed initial-state indices and a fixed seed, and every comparison in a table

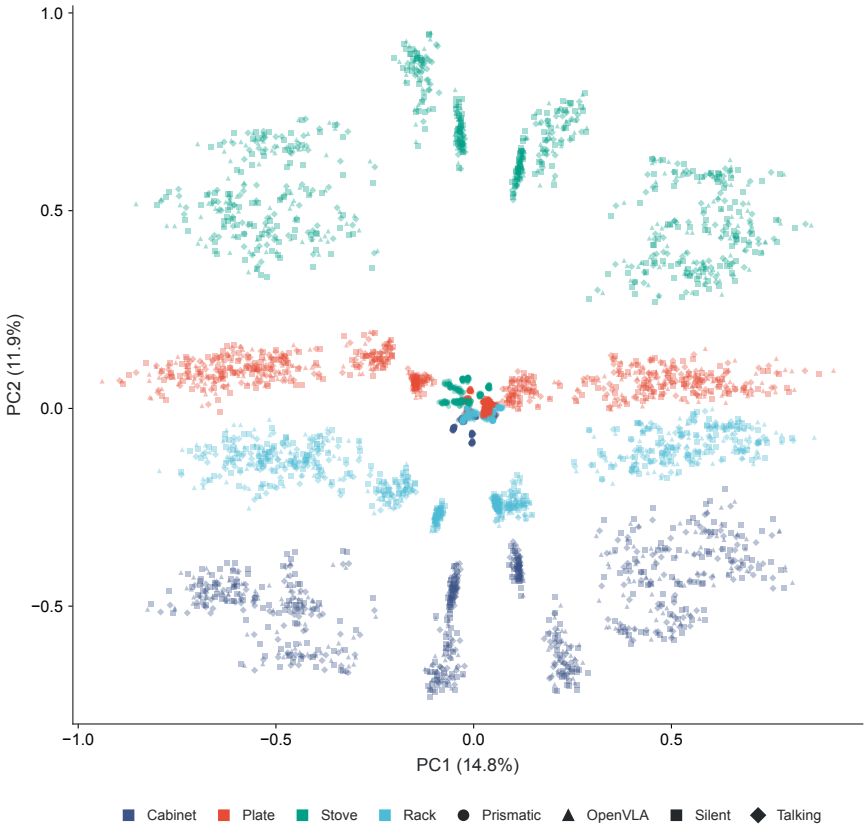



Figure 6: **A shared projection of matched-frame goal representations.** PCA of centred residual states from Prismatic, OpenVLA, silent and talking checkpoints. Colours identify four destinations; marker shapes identify checkpoints. All 4,800 displayed states are held out from PCA fitting: eight object–destination combinations, 30 frames and five layers (8, 10, 12, 14, 16) per checkpoint. The two components retain 26.6% of the variance among the normalized fitting means. Points reuse frames across conditions and layers. Appendix A specifies centring, normalization and the shared projection.

was run in the same job or with the same indices. An action slot that contains a non-action token is decoded as the nearest valid action and counted.

## B MODELS AND TRAINING

Table 5: Models. The matched policies are fine-tuned from the OpenVLA LIBERO-goal checkpoint on the 52k LIBERO-goal frames with AdamW (Loshchilov & Hutter, 2019) and 8-bit optimizer states (Dettmers et al., 2022), learning rate $10^{-5}$, 100 warm-up steps and cosine decay to 10%, batch 16, vision backbone and projector frozen, 6,000 steps (the shortcut-free rule policies: 12,000).

| Model | Backbone | Role |
|---|---|---|
| OPENVLA LINEAGE | | |
| Llama-2-7B | – | parent language model |
| Prismatic-7B | Llama-2-7B | vision-language parent of OpenVLA |
| OpenVLA OXE | Prismatic-7B | robot policy, many-robot mixture |
| OpenVLA | OpenVLA OXE | LIBERO-goal policy |
| silent | OpenVLA | original recipe, reference |
| talking, half, blind | OpenVLA | state the goal before acting (§6) |
| negation-, rule-trained | OpenVLA | sentences added to robot data (§5) |
| text-restored | OpenVLA | text batch added to every step (§5) |
| shortcut-free rule-trained | OpenVLA | rules on frames that hide the destination (§5) |
| $\pi$ LINEAGE | | |
| Gemma-2B | – | parent language model |
| PaliGemma-3B | Gemma-2B | vision-language parent |
| $\pi_{0.5}$-base | PaliGemma-3B | generalist robot policy |
| $\pi_{0.5}$ | $\pi_{0.5}$-base | LIBERO policy |
| EXTERNAL | | |
| ECoT | OpenVLA | public chain-of-thought policy, LIBERO-object |

**Speaking policies.** The talking, half and blind policies emit, before the seven action tokens, "task: ⟨verb⟩; object: ⟨object⟩; target: ⟨goal⟩; phase: ⟨phase⟩. Action:", where verb, object and goal are a deterministic parse of the instruction and the phase is *reach* until the demonstration's first gripper closure, *hold* while closed and *release* afterwards. In closed loop the phase is set from the policy's own last gripper action and the line is generated freely except where a condition forces it. The talking policy is trained with the instruction in every prompt; the half policy names the instruction with probability 0.5 and says "do the task" otherwise, drawn once with a fixed seed; the blind policy always says "do the task", so the goal enters only through its line, which is teacher-forced at test time. On 512 held-out frames the silent policy reaches 0.944 action top-1 (starting checkpoint 0.897), the talking policy 0.893 with 0.998 accuracy on the line, the half policy 0.874. A second seed was trained for each speaking policy.

**Negation- and rule-trained policies.** Half of the bowl frames (6,965 of 13,991) have their instruction rewritten as a negation or as a rule, drawn from 180 distinct training sentences whose marker words are disjoint from the 30 test marker words. Two seeds were trained for 6,000 steps. A rule-only variant was trained for 24,000 steps with checkpoints at 6,000, 12,000 and 24,000.

**Text-restored policies.** Each optimizer step pairs the robot batch of 16 with a text batch of 8: four windows of generic text, two copying sequences (64 random word pieces repeated) and two rule questions of the form "$a$ means $x$, $b$ means $y$, $c$ means $z$. the marker is $m$. the answer is" over 20 marker triples and 40 nouns, all disjoint from the robot task and the test sets. The loss is the robot loss plus the mean of the three text losses. Two variants, without and with rule sentences in the robot data, were each trained with two seeds.

**Public chain-of-thought policy.** We evaluate the community checkpoint `leepanic/ecot-libero-object-r32`,[1] an OpenVLA-7B policy for LIBERO-object.

[1] https://huggingface.co/leepanic/ecot-libero-object-r32

It writes TASK, PLAN, SUBTASK REASONING, SUBTASK, MOVE REASONING, MOVE, GRIPPER POSITION and VISIBLE OBJECTS, regenerated every eight control steps, and generates the seven action tokens after "ACTION:" at every step.

## C INSTRUCTION FORMS

Table 6: Instruction forms for the three bowl destinations. Every family is written for all three destinations; in rules, every destination is the answer at every position equally often.

| Form | Template or example | Sentences |
|---|---|---|
| literal | put the bowl on the plate / on the stove / on top of the cabinet | 3 |
| paraphrase | place the bowl onto _; the bowl should go on _; move the bowl to _ | 9 |
| description | a synonym (dish, cooktop, cupboard); "the flat round thing people eat from"; "the thing that heats pots and pans" | 9 |
| rule | red means plate, blue means stove, green means cabinet. the marker is blue. put the bowl where the marker says (four test marker sets: colours, shades, shapes, animals; rule-first and marker-first orders) | 72 |
| two nouns | the same rule with two destinations (chance 0.5) | 48 |
| one noun | the same rule with one destination (no selection) | 24 |
| negation | put the bowl on _, not on _; do not put the bowl on _, put it on _ | 12 |
| new templates | _ rather than _; _, never _; never _; _; instead of _, _ | 24 |

## D THE GOAL CODE: FURTHER RESULTS

Table 7: Components carrying the goal and additive slots. Components of a counterfactual edit are injected separately; energy is the edit's share of squared norm in $V_{256}$ (random: 6.3%). †Answers flipped of 48, read after the instruction and edited at the destination noun. ‡22.7% at the token embedding (layer 0). Slot entries are step cosines with a shuffled reference; untrained cells give mean nearest-code matches out of three, after fitting on five grid cells.

**Components carrying the goal**

| Model | Layers | $V_{256}$ | $R_{256}$ | $V_{256}^{\perp}$ | Energy in $V_{256}$ |
|---|---|---|---|---|---|
| Llama-2† | 8–16 | **38/48** | 0/48 | 45/48 | 11–15% |
| Prismatic† | 8–16 | **35/48** | 0/48 | 41/48 | 10–13% |
| OpenVLA | 0–8 | **17/20** | 0/20 | 18/20 | 3.0–3.7%‡ |
| OpenVLA | 8–16 | 0/20 | 0/20 | 18/20 | 2.0–3.2% |

**Additive object and destination slots**

| Model | Destination step | Object step | Shuffled 95th pct. | Untrained cells |
|---|---|---|---|---|
| Llama-2 | 0.87 | 0.85 | 0.71 | 3/3 |
| Prismatic | 0.85 | 0.73 | 0.66 | 2.9/3 |
| OpenVLA | 0.82 | 0.79 | 0.63 | 3/3 |
| $\pi_{0.5}$ | 0.84 | 0.84 | 0.77 | 2.7/3 |

**Additive structure.** The codes of Table 7 are read at the end of the prompt in Llama-2 and Prismatic (layers 8–24), at the last token in OpenVLA (layers 8–16) and over the instruction suffix in $\pi_{0.5}$ (layers 2–10). An untrained cell counts as predicted when $\mu + a_o + b_d$ is nearest to its own code among the eight cells, averaged over frames. A destination decoder fitted on the bowl row reads the bottle row at 1.00 in all four models, and the reverse.

**Readability by position.** In OpenVLA, with the true instruction and the demonstration's action tokens teacher-forced (120 frames per pair), the vocabulary readout ranks the true destination above

the counterfactual one in 0.4 to 0.6 of frames at every action position and depth, with the true word's mean rank between $6\times10^3$ and $2\times10^4$. A lens fitted on Llama-2 itself and applied to OpenVLA's states reads the word perfectly at the position where it is typed (layers 4–8, mean rank 70–222) and nowhere downstream. The goal code reads 1.00 at the destination noun, 0.95–1.00 at the last token, 0.97–0.99 at the first action position (layers 10–24), 0.90–0.94 at the fourth and about 0.5 at the seventh. In the speaking policies the vocabulary reads the target only at the position about to emit it and only from layer 20 on, while the code reads 0.94–1.00 at every position that carries the line.

Table 8: Writing and deleting the goal code in detail. Episodes redirected (write) or completed (delete); reached: episodes in which the gripper comes within 0.1 m of the bowl.

| Condition | Redirected | Completed | Reached |
|---|---|---|---|
| OpenVLA, write at the destination noun, layers 8–16 ($n = 20$) | | | |
| literal counterfactual instruction | 17/20 | – | – |
| one-dimensional code difference | 18/20 | – | – |
| span of the ten codes | 17/20 | – | – |
| top three directions of the span | 14/20 | – | – |
| random subspace, rank 10 | 0/20 | 19/20 | – |
| OpenVLA, delete ($n = 10$) | | | |
| clean | – | 10/10 | 10/10 |
| goal plane, all positions, layers 8–16 | – | **0/10** | 0/10 |
| random rank 10, same place | – | 8/10 | 10/10 |
| goal plane, all positions, layers 4–16 | – | 0/10 | 0/10 |
| goal plane, last seven positions, layers 8–16 | – | 0/10 | 8/10 |
| random rank 10, same place | – | 7/10 | 10/10 |
| three leading directions, all positions | – | 0/10 | 10/10 |
| random rank 3, same place | – | 9/10 | 10/10 |
| goal plane, all positions, layers 24–32 | – | 9/10 | – |
| top ten vocabulary directions, all positions | – | 0/10 | 8/10 |
| top ten vocabulary directions, last seven positions | – | 7/10 | – |
| $\pi_{0.5}$, write at layers 0–4 ($n = 10$) | | | |
| whole state of the donor | 8/10 | – | – |
| one-dimensional code difference | 10/10 | – | – |
| span of the ten codes | 8/10 | – | – |
| random subspace, rank 10 | 0/10 | 8/10 | – |
| $\pi_{0.5}$, delete at the instruction positions, three tasks ($n = 10$ each) | | | |
| clean | – | 9, 9, 10 | – |
| goal plane, layers 0–4 | – | **0, 1, 5** | – |
| object-identity plane, layers 0–4 | – | 9, 8, 9 | – |
| random, rank 10 | – | 9, 10, 10 | – |
| goal plane, layers 4–9 | – | 9, 9, 10 | – |
| $\pi_{0.5}$, unseen LIBERO-90 kitchen, write at layers 0–4 ($n = 20$) | | | |
| literal counterfactual instruction | 8/20 | – | – |
| one-dimensional code difference | 13/20 | – | – |
| span of the ten codes | 11/20 | – | – |
| random subspace, rank 10 | 0/20 | 11/20 | – |

**Deletion depth and failure mode.** Deleting the goal plane at every position stops the policy up to layer 24 and is harmless after it (Table 8); deleting it only at the last seven positions after layer 16 is also harmless (7/10 and 8/10 completed), which places the goal at the tail positions until layer 16 and downstream afterwards. With the whole plane deleted the arm never reaches the bowl; with a smaller deletion it reaches the bowl and does not lift it. Deleting the top vocabulary directions at every position also stops the policy, but the arm reaches the bowl: those directions bear load in execution, not in the choice of goal.

**Lineage transplant and geometry.** The difference between Llama-2's states for two instructions on the text prompt, added to OpenVLA's states at the last seven positions at every control step, redirects

Table 9: Writing the object slot. Instruction "put the bowl on top of the cabinet"; the wine bottle's code is written at the object noun. Counts of 20 episodes: object grasped (bowl / bottle / none) and object placed on the cabinet (bowl / bottle).

| | OpenVLA, layers 0–9 | | $\pi_{0.5}$, layers 0–9 | |
|---|---|---|---|---|
| Condition | grasped | placed | grasped | placed |
| clean | 20 / 0 / 0 | 17 / 0 | 20 / 0 / 0 | 17 / 0 |
| literal bottle instruction | 0 / 20 / 0 | 0 / 18 | 0 / 19 / 1 | 0 / 18 |
| one-dimensional code difference | 0 / **17** / 3 | 0 / 15 | 0 / **20** / 0 | 0 / 19 |
| whole donor state | 0 / 17 / 3 | 0 / 15 | 0 / 20 / 0 | 0 / 20 |
| random rank 1 | 20 / 0 / 0 | 18 / 0 | 20 / 0 / 0 | 18 / 0 |
| random direction, same norm | 18 / 0 / 2 | 15 / 0 | 18 / 1 / 1 | 16 / 1 |
| bottle code in the destination slot | 20 / 0 / 0 | 17 / 0 | 20 / 0 / 0 | 16 / 0 |

Table 10: Destination codes act on the object and follow the plate. (a) $\pi_{0.5}$, instruction "put the wine bottle on top of the cabinet", a destination code learned only from bowl tasks, $g(\text{bowl}, \text{plate}) - g(\text{bowl}, \text{cabinet})$, written into the destination slot; 20 episodes. (b) Instruction "put the bowl on top of the cabinet" with the plate code written, the plate moved before the episode; bowl placed on the plate at its new position, 16 episodes.

**(a)** Synthetic destination code on the bottle

| Condition | Bottle grasped | Bottle within 10 cm of plate | Bottle on plate | Original task completed |
|---|---|---|---|---|
| clean | 17 | 0 | 0 | 16 |
| literal "wine bottle on the plate" (untrained) | 18 | 18 | 6 | 0 |
| synthetic code | 19 | **10** | 1 | 8 |
| bowl-to-plate code | 18 | 8 | 0 | 7 |
| random direction, same norm | 17 | 1 | 0 | 16 |

**(b)** Plate moved before the episode

| | $\pi_{0.5}$ | | | OpenVLA | | |
|---|---|---|---|---|---|---|
| Plate shift | literal | code | random | literal | code | random |
| none | 13 | 14 | 0 | 11 | 10 | 0 |
| $+8$ cm in $x$ | 9 | 9 | 0 | 7 | 7 | 0 |
| $+8$ cm in $y$ | 7 | 10 | 0 | 0 | 0 | 0 |
| $-8$ cm in $y$ | 11 | 9 | 0 | 2 | 5 | 0 |
| $+15$ cm in $y$ | 4 | 6 | 0 | 0 | 0 | 0 |

17/20 episodes at layers 0–8 (OpenVLA's own literal instruction swap: 16/20), 8/20 at layers 8–16 and 0/20 at layers 16–24; a random direction of the same norm redirects 0/20. The cosine between Llama-2's difference and OpenVLA's own falls from 0.88 at layer 0 to 0.29 at layer 16. The pairwise cosine matrices of the ten codes in OpenVLA and $\pi_{0.5}$ correlate at $r = 0.63$–$0.81$ at matched relative depth and $0.81$–$0.90$ at the best-matched layers (permutation $p < 0.001$), against $0.86$–$0.91$ between layers within one model and $0.13$–$0.34$ for the overlap of the instructions' words.

Table 11: Components of a counterfactual edit injected alone, per layer band. Destination: the destination read after the instruction, edit at the destination noun, answers flipped of 48 pairs; OpenVLA: goal difference at the last seven positions, episodes redirected of 20. Two-hop: edit at the city token of a two-hop question, answers flipped of 40 pairs. $V_r$: word-aligned subspace; rand.: random subspace of the same rank; comp.: complement of $V_r$. Gemma-2B has 18 layers.

| | Llama-2-7B | | | Prismatic-7B | | | Gemma-2B | | OpenVLA | |
|---|---|---|---|---|---|---|---|---|---|---|
| Component | 0–8 | 8–16 | 16–24 | 0–8 | 8–16 | 16–24 | 0–8 | 8–16 | 0–8 | 8–16 |
| DESTINATION | | | | | | | | | | |
| whole edit | 48 | 47 | 34 | 44 | 42 | 36 | 48 | 48 | – | 18 |
| $V_{64}$ | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | – | 0 |
| rand. 64 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | – | 0 |
| comp. 64 | 48 | 46 | 34 | 43 | 42 | 36 | 48 | 48 | – | 17 |
| $V_{256}$ | 44 | **38** | 0 | 39 | **35** | 0 | 48 | 28 | **17** | 0 |
| rand. 256 | 0 | 0 | 0 | 0 | 0 | 0 | 39 | 22 | 0 | 0 |
| comp. 256 | 48 | 45 | 34 | 42 | 41 | 34 | 44 | 48 | 18 | 18 |
| TWO-HOP QUESTION | | | | | | | | | | |
| whole edit | 39 | 40 | 33 | 40 | 40 | 32 | 39 | 39 | – | – |
| $V_{64}$ | 6 | 6 | 0 | 2 | 3 | 0 | 23 | 13 | – | – |
| rand. 64 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | – | – |
| comp. 64 | 39 | 39 | 23 | 40 | 40 | 21 | 39 | 38 | – | – |
| $V_{256}$ | 39 | 37 | 0 | 36 | 27 | 0 | 39 | 39 | – | – |
| rand. 256 | 0 | 0 | 0 | 2 | 1 | 0 | 37 | 16 | – | – |
| comp. 256 | 39 | 35 | 12 | 40 | 39 | 14 | 11 | 11 | – | – |

**Word alignment across the lineage.** Table 11 extends Table 7 to every band, to Gemma-2B and to a two-hop question. In Llama-2 and Prismatic the word-aligned subspace of rank 256 carries the destination at layers 0–16 while a random subspace of the same rank carries none of it, and carries nothing at layers 16–24; the edit's energy in $V_{256}$ is 11–15% and 10–13% at layers 8–16 against 6.25% for a random subspace. The two-hop question gives the same picture. In Gemma-2B a random subspace of rank 256 is one eighth of the residual dimension and carries much of the edit at the destination noun; at the position where the answer is read, $V_{256}$ flips 31 of 48 answers at layers 8–16 against 2 for the random subspace. In OpenVLA the word-aligned subspace carries the goal at layers 0–8, near the input tokens, and carries nothing at layers 8–16, where a random subspace of the same rank also carries nothing.

**The same exchange in OpenVLA.** Exchanging the coordinates of "cabinet" and "plate" at layers 10–24 with $\alpha = 1$ leaves the policy completing the cabinet task 10/10. With $\alpha = 2$ it completes 0/10, every one of the $300 \times 7$ action slots holds a word token (*cabinet cabinet plate cabinet cabinet cabinet plate*), and the arm never approaches the bowl (closest approach 0.30 m against 0.04 m); a random direction at the same gain leaves 9/10 completed. At layers 8–16, 0–32 and at $\alpha = 3$ the result is the same (1,791 to 2,100 word tokens per episode).

**The verbal-workspace signature.** The top-1 autocorrelation of lens readouts across positions, a structural diagnostic of the verbal workspace (Gurnee et al., 2026), peaks at 5.3 in layers 19–23 of Llama-2 and at 5.8 in layer 7 of PaliGemma. Action training flattens it (OpenVLA 1.6; $\pi_{0.5}$ 0.9), for OpenVLA trained on the many-robot mixture as much as for its LIBERO fine-tune. Training the policy to state its goal lowers its text negative log-likelihood from 17.7 to 12.8 and leaves the signature flat (1.3).

Table 12: Exchanging the vocabulary coordinates of two concepts at every position of a layer band, with gain $\alpha$. Qwen2.5-7B (Yang et al., 2024a) is an additional reference model. Language models: the unspoken country in a two-hop question; answers flipped of 40 pairs. The reported-country columns ask for the country itself. Random: a direction of the same norm at $\alpha = 2$.

| | | Two-hop answer flipped | | | Reported country flipped | | |
|---|---|---|---|---|---|---|---|
| Model | Band | $\alpha$=1 | $\alpha$=2 | random | $\alpha$=1 | $\alpha$=2 | random |
| Llama-2-7B | 0–32 | 5 | 23 | – | 24 | 24 | – |
| | 8–24 | 1 | 16 | 0 | 14 | 14 | 0 |
| Qwen2.5-7B | 0–28 | 3 | 23 | – | 20 | 24 | – |
| | 8–20 | 0 | 15 | 0 | 0 | 11 | 0 |
| Gemma-2B | 0–18 | 27 | 20 | – | 28 | 19 | – |
| | 4–12 | 4 | 18 | 11 | 1 | 16 | 6 |

## E What the policy says: further results

Table 13: The speaking policies and the silent reference, seed 0, instruction cabinet, donor plate. Cells with $n = 20$ pool the 3,000- and 6,000-step checkpoints, which agree cell by cell; cells with $n = 10$ are the 6,000-step checkpoint. Entries: redirected / completed.

| Condition | silent | talking | blind |
|---|---|---|---|
| clean (own line for the speaking policies) | 0/20 · 20/20 | 0/20 · 17/20 | 0/20 · 16/20 |
| instruction (or line) says plate | 20/20 · 0/20 | 18/20 · 0/20 | 20/20 · 0/20 |
| instruction cabinet, made to say plate | – | 0/20 · 17/20 | – |
| made to say piano | – | 0/20 · 19/20 | 9/20 · 1/20 |
| phase word forced to release | – | 0/20 · 17/20 | 0/20 · 16/20 |
| vocabulary exchange, layers 8–16, $\alpha$=1 | 0/20 · 16/20 | 0/20 · 18/20 | 0/20 · 18/20 |
| vocabulary exchange, layers 8–16, $\alpha$=2 | 0/20 · 0/20 | 0/20 · 0/20 | 0/10 · 0/10 |
| INSTRUCTION NOUN, WHOLE STATE | | | |
| layers 0–8 | 17/20 · 0/20 | 9/10 · 0/10 | – |
| layers 8–16 | 19/20 · 0/20 | 6/10 · 1/10 | – |
| layers 16–24 | 0/20 · 17/20 | 0/10 · 10/10 | – |
| WRITTEN TARGET WORD, WHOLE STATE | | | |
| layers 0–8 | – | 0/10 · 10/10 | 18/20 · 0/20 |
| layers 8–16 | – | 0/10 · 9/10 | 18/20 · 0/20 |
| layers 16–24 | – | 0/10 · 10/10 | 0/20 · 18/20 |
| layers 24–32 | – | 0/10 · 9/10 | 0/10 · 10/10 |
| from the written target to “Action:”, layers 8–16 | – | 2/10 · 0/10 | 18/20 · 0/20 |
| COMPONENTS AT LAYERS 8–16, RE-RUN WITH RANK-MATCHED CONTROLS ($n = 20$) | | | |
| instruction noun: whole / $V_{64}$ / rand. 64 / comp. 64 | 19 / 0 / 0 / 17 | – | – |
| written word: whole / $V_{64}$ / rand. 64 / comp. 64 | – | 0 / 0 / 0 / 0 | 19 / 0 / 0 / 19 |

**Public chain-of-thought policy: screening.** In LIBERO-object each object is the target only in its own scene, so a changed instruction is usually out of distribution. Before testing the chain we screened 14 scene pairs with a rule fixed in advance: a pair enters if the changed instruction puts the other object in the basket in at least 5 of 10 episodes and the clean policy completes at least 7 of 10. Two pairs pass (Table 17); in eleven, the changed instruction redirects none. Repeated runs of the same cell with identical seeds differ by up to three episodes in ten, because bf16 reductions on the GPU are not deterministic and 300 closed-loop steps amplify the difference.

## F What the policy infers: further results

Figure 7 gives the values behind Figure 2 for all five forms, negation included. Table 18 gives the noun-count controls and the closed-loop outcomes; Table 19 separates goal-code and vocabulary readouts at the question position.

Table 14: The half policy, 10 episodes per cell, seed 0, in both prompt modes. Entries: redirected / completed.

| | Instruction visible | | Instruction hidden | |
|---|---|---|---|---|
| Condition | 3k | 6k | 3k | 6k |
| clean | 0/10 · 8/10 | 0/10 · 8/10 | 0/10 · 9/10 | 0/10 · 8/10 |
| instruction or line says plate | 10/10 · 0/10 | 10/10 · 0/10 | 10/10 · 0/10 | 10/10 · 0/10 |
| instruction cabinet, made to say plate | 4/10 · 3/10 | 6/10 · 3/10 | – | – |
| phase word forced | 0/10 · 10/10 | 0/10 · 9/10 | 0/10 · 8/10 | 0/10 · 9/10 |
| written word, layers 0–8 | 2/10 · 6/10 | 4/10 · 3/10 | 9/10 · 0/10 | 10/10 · 0/10 |
| written word, layers 8–16 | 1/10 · 8/10 | 0/10 · 10/10 | 9/10 · 0/10 | 10/10 · 0/10 |
| written word to "Action:", layers 8–16 | 9/10 · 0/10 | 9/10 · 0/10 | – | – |
| instruction noun, layers 8–16 | 0/10 · 8/10 | 0/10 · 8/10 | – | – |

Table 15: Second training seed, 20 episodes per cell, next to seed 0. Entries: redirected (completed).

| Policy | Condition | Seed 1 | Seed 0 |
|---|---|---|---|
| talking | made to say plate | 0/20 (18) | 0/20 (17) |
| | written word, whole, layers 8–16 | 0/20 (18) | 0/10 (9) |
| | instruction noun, whole, layers 8–16 | 15/20 (2) | 6/10 (1) |
| half | made to say plate | 11/20 (6) | 10/20 |
| | written word, whole, layers 0–8 | 8/20 (10) | 6/20 |
| | written word to "Action:", layers 8–16 | 18/20 (0) | 18/20 |
| | hidden instruction, line says plate | 17/20 (0) | 20/20 |
| blind | line says plate | 19/20 (0) | 20/20 (0) |
| | written word, whole, layers 8–16 | 19/20 (0) | 18/20 (0) |
| | written word, $V_{64}$ / comp. 64 | 0/20 / 18/20 | 0/20 / 19/20 |
| | phase word forced to release | 0/20 (18) | 0/20 (16) |

Table 16: Replications, 10 episodes per cell. Reverse arena: instruction plate, donor cabinet, entries are episodes that stay on the plate. Stove pair: instruction stove, donor plate ("stove" is two tokens), entries are episodes redirected.

| | Reverse (on plate) | | Stove pair (redirected) | |
|---|---|---|---|---|
| Condition | silent | talking | silent | talking |
| clean | 10/10 | 10/10 | 0/10 | 0/10 |
| instruction names the donor | 0/10 | 0/10 | 10/10 | 10/10 |
| made to say the donor | – | 9/10 | – | 0/10 |
| written word, layers 8–16, whole | – | 9/10 | – | 0/10 |
| instruction noun, layers 8–16, whole | 0/10 | 0/10 | 10/10 | 9/10 |
| instruction noun, $V_{64}$ component | 9/10 | 7/10 | 0/10 | 0/10 |
| vocabulary exchange, $\alpha$=1 | 9/10 | 7/10 | 0/10 | 0/10 |

Table 17: The public chain-of-thought policy on the two screened pairs (tomato sauce → milk, chocolate pudding → orange juice), 20 episodes per condition. Chain regenerated: the instruction is kept and the whole chain is replaced by the one generated under the changed instruction. Open loop: on the clean trajectory's frames, the component of the change in predicted motion along the direction that separates the two objects as seen from the gripper, before the grasp, mean over 20 episodes with 95% bootstrap intervals.

| Condition | Instructed completed | Other object in basket | Gripper to instructed (m) | Gripper to other (m) | Open-loop pull |
|---|---|---|---|---|---|
| clean | 16/20 | 0/20 | 0.019 | 0.202 | – |
| instruction changed | 0/20 | **14/20** | 0.216 | 0.037 | 0.233 [0.200, 0.266] |
| chain regenerated | 8/20 | **0/20** | 0.037 | 0.162 | 0.036 [0.015, 0.058] |
| object renamed in chain | 19/20 | 0/20 | 0.017 | 0.201 | −0.007 [−0.014, −0.001] |
| no chain | 18/20 | 0/20 | 0.018 | 0.192 | 0.003 [−0.014, 0.019] |

Table 18: Controls and closed loop for the instruction forms of Figure 7. Top: the rule's wording with two destination nouns (chance 0.50) or one, decoded on the goal code as in Figure 7. Bottom: episodes completed of 30. OXE, LIBERO: OpenVLA trained on the robot mixture and its LIBERO fine-tune; silent, talking: our fine-tunes of the latter; base, LIBERO: $\pi_{0.5}$-base and its LIBERO fine-tune. *All plate targets.

| | OpenVLA lineage | | | | | | $\pi$ lineage | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Llama-2 | Prismatic | OXE | LIBERO | silent | talking | Gemma | PaliGemma | base | LIBERO |
| GOAL CODE | *rule wording* | | | | | | | | | |
| two nouns | 0.53 | 0.49 | 0.50 | 0.53 | 0.54 | 0.54 | 0.49 | 0.38 | 0.50 | 0.50 |
| one noun | 0.97 | 0.86 | 1.00 | 1.00 | 1.00 | 1.00 | 0.94 | 0.43 | 1.00 | 1.00 |
| CLOSED LOOP | *episodes completed of 30* | | | | | | | | | |
| literal | | | | 27 | 24 | 26 | | | | 30 |
| paraphrase | | | | 24 | 27 | 28 | | | | 26 |
| description | | | | 16 | 16 | 11 | | | | 15 |
| rule | | | | 0 | 0 | 0 | | | | 9* |
| negation | | | | 16 | 17 | – | | | | 18 |

Table 19: The instruction forms read at a question position: the sentence followed by "The robot should put the bowl on the," read at its last token. (a) Goal code at that position (layers as in Figure 7). (b) Destination named in the model's own next word, family-debiased. Chance 0.33 except two nouns (0.50).

| | Llama-2 | Prismatic | OpenVLA OXE | OpenVLA | silent | talking | Gemma | PaliGemma | $\pi_{0.5}$ base | $\pi_{0.5}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| (A) GOAL CODE AT THE QUESTION POSITION | | | | | | | | | | |
| literal | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.99 | 1.00 | 1.00 |
| description | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.77 | 0.61 | 0.90 | 0.77 |
| rule | 0.43 | 0.55 | 0.34 | 0.33 | 0.33 | 0.32 | 0.33 | 0.34 | 0.34 | 0.33 |
| two nouns | 0.53 | 0.58 | 0.49 | 0.49 | 0.50 | 0.49 | 0.49 | 0.46 | 0.51 | 0.50 |
| negation | 0.71 | 0.86 | 0.66 | 0.65 | 0.65 | 0.66 | 0.49 | 0.50 | 0.63 | 0.52 |
| negation, new templates | 0.80 | 0.92 | 0.74 | 0.75 | 0.75 | 0.75 | 0.72 | 0.50 | 0.56 | 0.57 |
| (B) DESTINATION NAMED IN THE NEXT WORD | | | | | | | | | | |
| literal | 1.00 | 1.00 | 0.92 | 0.90 | 0.84 | 0.96 | 1.00 | 0.27 | 1.00 | 0.29 |
| description | 1.00 | 1.00 | 0.39 | 0.33 | 0.37 | 0.42 | 1.00 | 0.47 | 1.00 | 0.31 |
| rule | 0.50 | 0.65 | 0.34 | 0.34 | 0.33 | 0.36 | 0.74 | 0.32 | 0.27 | 0.32 |
| negation | 1.00 | 1.00 | 0.52 | 0.50 | 0.49 | 0.47 | 1.00 | 0.23 | 0.79 | 0.32 |

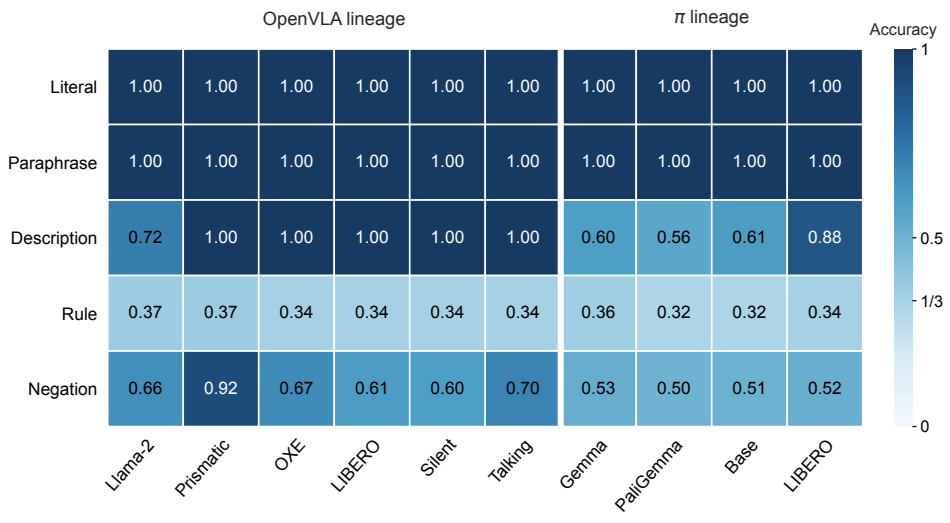



Figure 7: **Instruction forms differ in what reaches the goal code.** Held-out destination-decoding accuracy for five forms across ten models. Each cell gives the raw accuracy, each sentence centred on its family; colour uses one 0–1 scale, with three-destination chance $(1/3)$ marked on the bar. Read positions and layer bands follow §3 (Llama-2 and Prismatic: end of the prompt, layers 8–24). OXE and the first LIBERO column are OpenVLA before and after LIBERO-goal fine-tuning; base and the second LIBERO column are the corresponding $\pi_{0.5}$ checkpoints. Silent and talking are our matched OpenVLA fine-tunes.

**Reading the table.** The language models answer descriptions and negations in words and the rule partly; the robot policies lose even the description in words (0.33 to 0.42) while their goal code keeps it (1.00). PaliGemma and $\pi_{0.5}$ do not answer in this format at all. Asking lifts the rule on the code in the language models (Llama-2 0.43, Prismatic 0.55) and in no robot policy.

**Depth profiles for descriptions and rules.** Over the preselected read bands, the depth scans of Figure 3 give description and rule accuracies of 0.72 and 0.37 in Llama-2 and 1.00 and 0.34 in OpenVLA, as in Figure 7.

Table 20: Negation by word order. Code: share of held-out sentences whose centred state points to the target (the rest point to the negated noun). Closed loop: episodes by where the bowl ends.

| | Code on target | | Closed loop | | | |
|---|---|---|---|---|---|---|
| Model | negated noun last | target last | order | target | negated | none |
| Llama-2-7B | 0.52 | 0.69 | | | | |
| Prismatic-7B | 0.84 | 0.95 | | | | |
| OpenVLA | 0.27 | 0.92 | negated last | 8/18 | **8/18** | 2/18 |
| | | | target last | 8/12 | 1/12 | 3/12 |
| $\pi_{0.5}$ | 0.58 | 0.51 | negated last | 15/18 | 1/18 | 2/18 |
| | | | target last | 3/12 | **5/12** | 4/12 |

**New negation templates.** On four negation templates not used anywhere else, OpenVLA completes 21/48 episodes, $\pi_{0.5}$ 31/48 and the silent policy 23/48 (plain negation: 17/30).

**Where the code decides behaviour.** In LIBERO-object each object is the target only in its own scene. There OpenVLA completes 28 of 30 episodes when each object is asked for in its own scene and none of 30 when the same sentences are spoken in another object's scene, although its code

decodes the named object at 1.00 (Table 21): the code decides behaviour only where training made the instruction necessary.

Table 21: A second variable: which object to pick in LIBERO-object (butter, milk, tomato sauce), read at the same sites as the destination. Closed loop: literal instruction, in scenes where the named object is not at the usual target position (neutral) and in each object's own scene.

| | OpenVLA | $\pi_{0.5}$ |
|---|---|---|
| GOAL CODE (CHANCE 0.33) | | |
| literal | 1.00 | 1.00 |
| paraphrase | 1.00 | 1.00 |
| description | 0.52 | 0.46 |
| rule | 0.34 | 0.33 |
| one noun | 1.00 | 1.00 |
| negation | 0.65 | 0.57 |
| CLOSED LOOP, LITERAL INSTRUCTION, COMPLETED OF 30 | | |
| neutral scenes | 0 | 0 |
| own scenes | 28 | 13 |

**No stage variable.** In a two-stage LIBERO-10 task whose fixed instruction asks for two objects in the basket, the $\pi_{0.5}$ state on the axis between the two single-step instructions does not move when the first stage completes, in the text positions, the image tokens or the action expert; writing the second step's code during the first stage does not change the order. When the first object is returned to its place after being put in the basket, the policy goes on to the second object in every one of eleven such episodes: progress is carried by the scene and the arm, not by the goal code.

## G WHY: FURTHER RESULTS

Figure 8 samples the depth curves in Figure 4 at five layers, retaining the variation across the four restored runs. Tables 22–24 provide the circuit, checkpoint and input-format measurements.

Table 22: Retrieval circuitry along both lineages. Induction heads: repeated-sequence score (Olsson et al., 2022) above 0.3 (best of four input variants). Marker and answer: held-out accuracy of the best single head at attending, from the marker and from the answer position, to the correct destination noun in rule sentences (chance 0.33). Answer at ask: destination named in the next word.

| Model | Induction heads | Best induction score | Marker head | Answer head | Answer at ask |
|---|---|---|---|---|---|
| OPENVLA LINEAGE | | | | | |
| Llama-2-7B | 12 | 0.97 | 0.97 | 0.47 | 0.53 |
| Prismatic-7B | 12 | 0.97 | 0.82 | 0.56 | 0.64 |
| OpenVLA OXE | 0 | 0.20 | 0.81 | 0.31 | 0.32 |
| OpenVLA | 0 | 0.10 | 0.84 | 0.23 | 0.32 |
| silent | 0 | 0.09 | 0.83 | 0.27 | 0.36 |
| talking / half / blind | 0 | 0.13–0.15 | 0.86–0.90 | 0.23–0.28 | 0.32–0.38 |
| rule-trained | 0 | 0.04 | 0.76 | 0.23 | 0.36 |
| text-restored (four runs) | 20–22 | 0.95 | 0.88–1.00 | 0.23–0.42 | 0.31–0.38 |
| $\pi$ LINEAGE | | | | | |
| Gemma-2B | 7 | 0.98 | 0.89 | 0.73 | 0.65 |
| PaliGemma-3B | 0 | 0.09 | 0.23 | 0.25 | 0.33 |
| $\pi_{0.5}$-base | 2 | 0.54 | 0.47 | 0.34 | 0.33 |
| $\pi_{0.5}$ | 0 | 0.01 | 0.22 | 0.25 | 0.33 |
| REFERENCE | | | | | |
| Qwen2.5-7B | 28 | 0.99 | 0.84 | 0.96 | 0.85 |

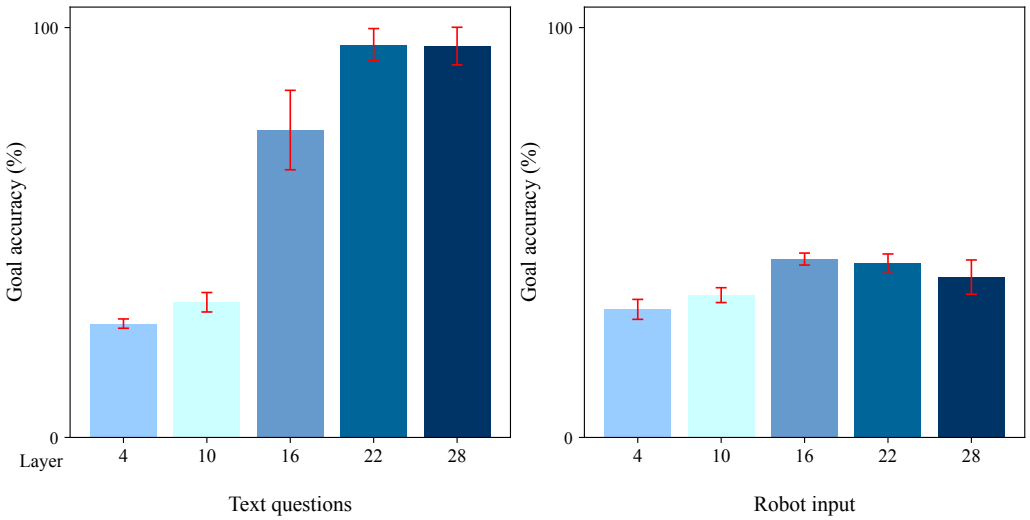



Figure 8: **Selected layers of the text and robot-format readouts.** Mean destination-decoding accuracy at layers 4, 10, 16, 22 and 28 in four restored OpenVLA runs: two training recipes with two seeds each. Red error bars show one sample standard deviation across runs. Text questions contain 72 rule problems; robot input places the same problems in the robot template with three camera frames, yielding 216 readouts per run. Both plots use the same percentage scale; chance is $1/3$. Figure 4 shows all measured layers and the intermediate template-only condition.

**Ablating induction heads in Llama-2.** Replacing the outputs of Llama-2's $k$ strongest induction heads by their mean lowers the rule answer at the question position from 0.53 to 0.46, 0.49 and 0.38 for $k = 5$, 10 and 20; the same number of random heads from the same layers gives 0.53, 0.51 and 0.48 ($\pm 0.03$).

Table 23: Text-restored policies after 6,000 steps. Action: teacher-forced top-1 on held-out frames (the rule variant is evaluated on frames that include rule sentences). Text NLL and autocorrelation: generic text, lens screen. Copy: held-out copying accuracy. Rule QA: held-out rule questions over unseen marker sets and nouns. Code: rule on the goal code at the last token and at the question position, layers 8–16. *The OpenVLA checkpoint; on the rule variant's evaluation frames its action accuracy is 0.80.

| Run | Action | Text NLL | Autocorr. peak | Copy | Rule QA | Induction heads | Rule on code last / ask |
|---|---|---|---|---|---|---|---|
| before training* | 0.90 | 17.72 | 1.28 | 0.00 | 0.39 | 0 | 0.34 / 0.33 |
| text-restored, seed 0 | 0.94 | 4.13 | 2.38 | 0.81 | 0.94 | 20 | 0.33 / 0.33 |
| text-restored, seed 1 | 0.94 | 4.14 | 2.33 | 0.80 | 0.96 | 21 | 0.34 / 0.34 |
| + rule sentences, seed 0 | 0.88 | 4.21 | 2.37 | 0.80 | 0.90 | 22 | 0.37 / 0.34 |
| + rule sentences, seed 1 | 0.88 | 4.21 | 2.32 | 0.80 | 0.94 | 22 | 0.37 / 0.34 |

**Negation- and rule-trained policies in closed loop.** The two seeds complete literal instructions 28 and 29 of 30 times, descriptions 18 and 18, rules 8 and 10, and negations 28 and 26; on the four new negation templates, seed 0 completes 32 of 48.

**Rule training without the pose shortcut.** Rule sentences are drawn from 512 (or 64) marker sets of single-token words, disjoint from the held-out marker words and from every template word, with all six marker-to-destination assignments. In the window arms every rule example takes a frame from $t = 2$ to 9 of a training episode of its destination. There a linear probe on the policy's frozen visual features decodes the destination at 0.35 to 0.41 (chance 0.33; five folds split by episode) while the demonstrated action decodes it at 0.60 to 0.64; the proprio state, which the policy does not see,

Table 24: The same rule as the prompt moves from text to the robot's format, family-debiased answer in the next word. S0: text question, unrelated nouns. S1: the same with plate, stove and cabinet. S2: S1 inside the robot's instruction template. S3: S2 with a camera frame. S4: the robot's rule instruction followed by a question. Last row: best layer at which the S1 answer reaches the destination code.

| Format | Llama-2 | silent | restored seed 0 | restored seed 1 | + rules seed 0 | + rules seed 1 |
|---|---|---|---|---|---|---|
| S0 | 0.82 | 0.33 | 0.97 | 0.99 | 0.93 | 0.99 |
| S1 | 0.62 | 0.29 | 0.93 | 1.00 | 0.92 | 1.00 |
| S2 | 0.60 | 0.43 | 0.51 | 0.61 | 0.46 | 0.50 |
| answer named first | 0.42 | 0.58 | 1.00 | 1.00 | 1.00 | 1.00 |
| answer named second / third | 0.58 / 0.79 | 0.29 / 0.42 | 0.33 / 0.21 | 0.38 / 0.46 | 0.17 / 0.21 | 0.21 / 0.29 |
| S3 | – | 0.37 | 0.36 | 0.40 | 0.34 | 0.38 |
| S4 | 0.50 | 0.33 | 0.36 | 0.35 | 0.34 | 0.34 |
| S1 on the code, best layer | 0.68 (24) | 0.40 (3) | 0.96 (20) | 0.99 (22) | 0.90 (22) | 0.99 (22) |

Table 25: Training on negation and rule sentences. (a) Rule-only training for 24,000 steps, two seeds: teacher-forced action accuracy on training rule frames, and the rule on the goal code at the last token for the training sentences and for held-out sentences. (b) Change of the predicted action tokens (mean absolute bin difference, early frames, held-out marker sets) when only the marker changes, when the rule is reworded with the same answer, and when a literal instruction changes.

**(a)** Rule-only training

| Steps | Action fit | Code, training sentences | Code, held-out sentences |
|---|---|---|---|
| 6,000 | 0.98 / 0.96 | 0.39 / 0.38 | 0.36 / 0.37 |
| 12,000 | 1.00 / 1.00 | 0.38 / 0.39 | 0.35 / 0.36 |
| 24,000 | 1.00 / 1.00 | 0.38 / 0.39 | 0.36 / 0.36 |

**(b)** What moves the action

| Policy | Marker changed (answer changes) | Rule reworded (answer kept) | Literal instruction changed |
|---|---|---|---|
| negation- and rule-trained, seed 0 | 2.85 | 4.73 | 13.50 |
| negation- and rule-trained, seed 1 | 3.21 | 4.47 | 11.85 |
| silent | 8.13 | 14.17 | 12.92 |

decodes it at 0.44 to 0.55 from the first frames on. A fifth of the bowl episodes (29, balanced by destination) is held out with its 232 window frames removed from training in every arm. The rule dose per step equals that of the negation- and rule-trained policies. Table 26 gives the result after 12,000 steps; the code and the held-out accuracies at 3,000 and 6,000 steps are within 0.02 of it, and the contrast index is negative at every checkpoint. On the training episodes' window frames the window arms ignore the sentence: changing the marker or the wording moves their predicted action by at most 0.14 bins, against 11 for a changed literal instruction. The window arms never see a rule sentence after the window and in closed loop mostly leave the bowl in place; the arm with rule sentences on every frame moves it to destinations unrelated to the rule, as the negation- and rule-trained policies do.

Table 26: Rule training without the pose shortcut, after 12,000 steps. Code: the rule on the goal code at the last token, layers 8–16, with held-out and with trained marker words (chance 0.33). Held-out episodes: teacher-forced action top-1 on the 232 window frames of the held-out episodes under a rule with held-out marker words that names the frame's destination (right) or, with only the marker changed, another one (wrong), and under the right and the wrong literal instruction. Index: marker-swap contrast on the same frames. Closed loop: episodes completed of 30, literal and rule instructions.

| | | | | Held-out episodes | | | |
|---|---|---|---|---|---|---|---|
| Arm | Rule frames | Marker sets | Code held-out / trained | Rule right / wrong | Literal right / wrong | Index | Closed loop literal / rule |
| S512, seed 0 | $t = 2$–$9$ | 512 | 0.35 / 0.35 | 0.47 / 0.47 | 0.94 / 0.41 | $-0.55$ | 27 / 0 |
| S512, seed 1 | $t = 2$–$9$ | 512 | 0.35 / 0.28 | 0.48 / 0.48 | 0.93 / 0.40 | $-0.46$ | 28 / 1 |
| S64 | $t = 2$–$9$ | 64 | 0.35 / 0.35 | 0.47 / 0.47 | 0.94 / 0.41 | $-0.51$ | 29 / 3 |
| A512 | all | 512 | 0.35 / 0.35 | 0.58 / 0.57 | 0.94 / 0.41 | $-0.25$ | 28 / 9 |
| RS512* | $t = 2$–$9$ | 512 | 0.35 / 0.33 | 0.48 / 0.48 | 0.97 / 0.41 | $-0.45$ | 28 / 2 |
| silent | – | – | 0.34 / – | – | – | – | 24 / 0 |

*Initialised from a text-restored policy and trained with its text batch; 20 induction heads at 12,000 steps.

## H DELIVERY: FURTHER RESULTS

**Writing into the sentence the robot hears.** Writing the correct code into the noun positions of the sentence the robot hears helps where the policy can act under that sentence: $\pi_{0.5}$ rises from 15 to 23 of 30 on descriptions and from 0 to 14 of 20 on stove and cabinet rules, and OpenVLA from 16 to 22 of 30 on negations (random directions of the same norm: 16, 0 and 15). Under the rule sentence OpenVLA still goes nowhere in 30 episodes, although the code reads the target at 0.93 to 0.95 and every action token is valid: that sentence itself stops this policy. The deliveries of Table 4 therefore write into a literal instruction, the carrier.

**The bridge.** The source is Llama-2-7B. For rules the prompt gives six worked examples in the same format and has the model restate the marker before the answer position; for negations two examples suffice. The layer-15 state at the answer position is projected onto the plane of three destination prototypes and expressed in barycentric coordinates. The prototypes are fitted only on sentences the robot never hears (other marker sets for rules; four other negation templates over six pairs for negations), and the example scaffold is chosen on the fitting set. No robot outcome enters any fit. Llama-2 answers the rule test set correctly in 33 of 36 cases; the decoded conclusion is right for 29 of the 30 rule sentences and 30 of 30 negations used in closed loop, and its spoken answer, used for the word route, is right for 27 of 30 rules and 30 of 30 negations.

**The text-restored policies' own rule conclusion.** Each of the four text-restored policies reads the 72 rule sentences rewritten as text questions ("$a$ means $x$, $b$ means $y$, $c$ means $z$. the marker is $m$. the answer is"), with the markers, assignment, stated marker and order of the robot's sentences and without an image. Its state at the last position is centred on the mean of its balanced sentence group and decoded against three destination prototypes fitted on marker sets 2 and 3, which the closed loop does not use; layer 21 is chosen by leave-one-group-out accuracy on those sets in all

four policies, from the band where the answer reaches the destination code (Table 24). The decoded destination's code is written into the destination slot of the literal carrier instruction after layer 0, as for the language model's conclusion, either as the decoded destination alone or as the barycentric weights; each policy's own codes serve as the goal monitor. No robot outcome enters the source. Table 2 gives every cell.

**OpenVLA's own answer to a negation.** The source is OpenVLA itself, reading each negation followed by "The robot should put the bowl on the". Its state at the last position is centred on the mean of its balanced sentence group and decoded against three destination prototypes fitted on the four new negation templates of Table 6, which the closed loop does not use; the layer (6) is chosen by leave-one-group-out accuracy on those templates. The decoded destination is right for 11 of the 12 negations used in closed loop, while the policy's next word names it for 6. The decoded destination's code is written into the destination slot of the literal carrier instruction after layer 0, as for the language model's conclusion; a random direction of the same norm completes none of 30 episodes.

# I REPRODUCIBILITY

**Environments.** Every OpenVLA closed-loop cell ran with NumPy 1.26.4 and every $\pi_{0.5}$ closed-loop cell with NumPy 2.2.6. Forward readouts are identical under the two versions to the last bit; the difference is confined to simulation. A matched comparison of four $\pi_{0.5}$ cells gives 74 of 120 completed episodes under NumPy 2.2.6 against 80 of 120 under 1.26.4 (McNemar $p = 0.24$), and the $\pi_{0.5}$ rule bridge gives 29 of 30 against 27 of 30. One cell of the half policy with the instruction hidden is sensitive: 9 of 20 under NumPy 2.2.6 against 19 of 20 under 1.26.4; the value in Table 15 is from the latter. The six OpenVLA jobs behind the writes and deletions of Table 8 and the lineage transplant were re-run with the NumPy version printed in the log: all 47 cells reproduce (two-sided Fisher $p \geq 0.34$; the code difference redirects 17 of 20 against 18 published, the span 18 against 17, the goal-plane deletion completes 0 of 10 again).

**Variation and controls.** Rollouts use bf16 on H100 GPUs, whose reductions are not deterministic; over 300 closed-loop steps this moves single episodes, and repeated runs of one cell differ by up to three episodes in ten. For this reason every comparison we draw is between conditions run in the same job, with the same initial states, and every write or deletion is paired with a random control of the same norm or rank run alongside it.