Planting Hidden Failures: Towards a Mechanistic Understanding of VLAs

Hongyi Shi
Carnegie Mellon University
chrisshi@andrew.cmu.edu

Abstract: We study how a SmolVLA policy fine-tuned to point a single joint at a red block computes its output. We identify and causally test what the action expert reads from the vision-language model, and find an estimate of the block's angle in the expert's residual stream that selects which cross-attention heads read the block. We use this account to create a model organism of misalignment in a VLA, a policy that fails on one contiguous half of the target range and works on the other. The same edit applied to a simulated SO-101 pick-and-place policy makes the arm fail for cubes away from the bearing written into it.

Keywords: VLA models, mechanistic interpretability, flow matching

1 Introduction

Vision-language-action policies are beginning to run on real hardware, and a tool meant to find a hidden failure in one before deployment can be trusted only once it has been tried on failures whose mechanism and extent are known. These policies are built on the same transformers that mechanistic interpretability reads in language models, where circuit analysis [1, 2], activation steering [3, 4], probing [5, 6] and gradient attribution [7, 8] locate and change what a model computes. Recent work applies such tools to these policies [9, 10, 11, 12], but two questions remain open: how far the tools carry over to a policy whose output is a flow-matched action chunk rather than a token, and what the action expert, the transformer that generates the actions, computes once fine-tuned. We study both questions on a SmolVLA policy fine-tuned with a frozen backbone to point one joint at a red block (Figure 1), whose known correct angle scores every quantity in degrees.

We make three contributions. First, we identify what the expert reads from the vision-language model, a few directions of that model's residual stream, its hidden state between layers, at the image tokens, found where the gradient of the output angle with respect to the cross-attention keys and values concentrates. Earlier analyses locate what the expert uses by layer or by token [13, 14, 15, 16]. Locating it by direction lets us test it against random and principal subspaces of the same size: removing the directions raises the median error from 0.37° to 60.8°, while removing an equal random subspace leaves it unchanged (Section 3.1). Written into the backbone's residual stream, the directions mark a token for the vision-language model itself (Section 3.2), as edits that point a vision-language model at a region do in larger models [17]. A distractor acts through the same directions, and an edit built from them sends the policy to a chosen colour (Figure 3). Larger policies have been retargeted along learned or hand-chosen feature directions or by substituting the vision-language model's output for a donor scene [18, 10, 11, 9, 13], and cleared of a spurious cue by erasing a subspace of that output [19]; ours starts from the policy's own gradients, which with one direction separating the colours also flag the tokens to edit.

Second, we find in the expert's residual stream a two-dimensional estimate of the block angle. Replacing it, against a random plane of the same size, shows that half of the heads in layers 5 to 15 attending the block do so only when it puts the block on its true side of angle zero (Section 4.1).

Third, we plant a one-sided failure by editing those layers' query weights to use the estimate of one fixed angle in place of the stream's own, which keeps the error low on the half of the range

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

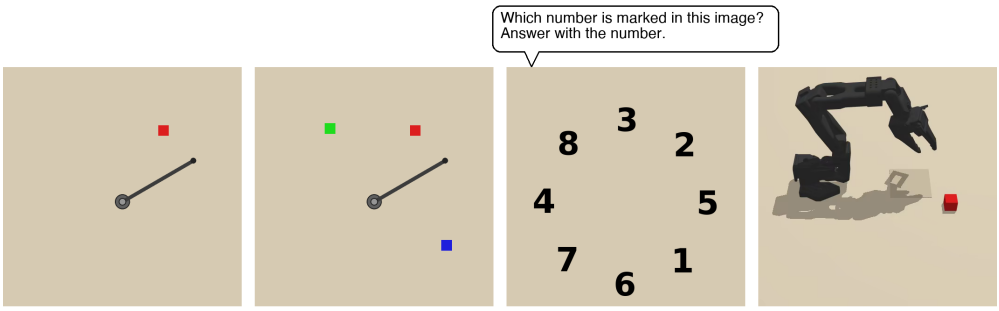


Figure 1: Tasks with known correct answers. (a) A one-link arm at a fixed pivot and a red block whose angle is the correct action. (b) The same with squares of other colours. (c) The pointer test, in which the frozen vision-language model is asked which digit was edited; a correct answer shows that the edit marks that token. (d) The simulated SO-101 arm and cube.

containing that angle and raises it on the other (Figure 2a). On a simulated SO-101 pick-and-place policy the refitted edit, writing the estimate of a cube at one bearing, gives closed-loop success 0.66 for cubes near that bearing and 0.19 elsewhere, against 0.84 unedited (Figure 2b). Failures have been planted in such policies as fine-tuned backdoors with text or image triggers [20, 21, 22] or as a state-conditioned trojan in a non-VLA controller [23], with detectors proposed for them [24, 25]. Ours is a weight edit derived from the mechanism, so its cause is known and its extent is measured. In the sense of Hubinger et al. [26] it is a model organism, limited to a planted competence failure, and we offer it as a test case for failure-finding tools.

2 Setting

In the toy task (Figure 1a), an image shows one link rotating about a fixed pivot and a red block at angle ϕ on the circle traced by its tip, and the policy must point the link at the block. The policy is SmolVLA [27], which pairs a vision-language model with an action expert [28], a transformer that generates a fixed-length action chunk, one row per action, by flow matching [29], from which we read one output angle. We fine-tune only the expert, with target ϕ held over the chunk, as in SmolVLA's own training and in other policies that freeze their vision-language model [27, 30, 31, 32]. At its odd layers the expert reads that model's output by cross-attention, with queries from the action rows and keys and values from the image tokens (Appendix A).

3 Understanding and Editing What the Expert Reads from the Vision-Language Model

The output angle depends mainly on the expert's odd layers. Skipping the attention and MLP of all eight even layers gives a median error of 4.31°, against 89.1° when the odd layers are skipped instead (Table 2). We therefore ask what the odd layers read from the vision-language model.

3.1 Finding and Testing the Directions the Expert Reads

The output angle depends on a few directions of the vision-language model's residual stream at the image tokens. We find them through the keys and values that the expert's cross-attention reads at each odd layer, which are linear functions of that stream. The gradient of the output angle with respect to these keys and values, taken over tokens, scenes and flow steps, is concentrated in a small subspace, and pulling that subspace back through the key and value projections gives directions of the stream itself. At each odd layer we keep the smallest set of gradient directions holding 90% of the gradient energy, defined as the sum of the squared gradient entries, which is an active subspace [33]. In all 67 directions remain over the eight odd layers, out of 320 per layer for each of keys and values (Appendix A), and from here on the directions the expert reads means these pull-backs.

Gradients measure only sensitivity, so we measure the causal effect of these directions in the open loop on held-out scenes, setting the components of the keys and values outside or inside the kept gradient directions to their means over scenes (Table 1, Appendix A). Removing the kept directions raises the median error from 0.37° to 60.8°, most of the way to the 82.5° reached with every image row at its mean, whereas removing an equal random subspace leaves it at 0.42°. Keeping the directions alone, with everything else at its mean, gives 1.30°, against 79.3° for an equal random subspace kept alone, and the same number of leading principal components does less on both tests. The output angle therefore passes almost entirely through these directions.

3.2 Writing the Directions into the Vision-Language Model

If these directions are what the expert reads, writing them into the frozen backbone's residual stream at one token should mark that token for the vision-language model, and at strengths well above a block token's own projection onto them they do. Because the backbone is frozen, the edited residual stream can be passed through the rest of the original vision-language model, with its language head, and the model run as normal. With unmarked digits in place of the block (Figure 1c), we edit one digit's token at one layer and ask the model which number is marked, choosing the layer on 200 scenes and measuring four strengths on 4,000 others (Tables 3 and 4). With the directions read through the values of layer 7, at 10 to 16 times that projection, the model names the edited digit with accuracy 0.808 to 0.891, against 0.122 with no edit (chance) and 0.975 with a black box drawn around the digit. A random direction of the same norm, tested at 14 times, does not raise it, and at a block's own strength, read through both keys and values, accuracy stays near chance (Table 3).

3.3 Retargeting the Policy Among Distractors

A distractor, a square of another colour, acts mainly through the same directions, and an edit on them sends the policy to a chosen colour. The square raises the median error to the red block from 0.4° to 27.9°. Removing the square's effect on the keys and values along those directions, at all image tokens, nearly undoes the rise, adding only that part of its effect to the scene without the square reproduces most of it, and the same part along a random subspace adds nothing (Table 5).

To point the joint at a chosen colour without retraining, we locate the block and square tokens by that component and pick the chosen colour's token with a colour-separating direction of that model. We then remove what the expert reads at the other tokens and enlarge it at the chosen one (Appendix D). With all three colours present, the median error to red falls from 42.7° to 10.2°. The edit gives lower errors for green and blue, on scenes that policies fine-tuned on those colours for this comparison miss (Figure 3). Without squares it raises the error from 0.4° to 1.1° (Table 5).

4 Understanding and Editing the Cross-Attention Queries

4.1 Finding and Testing an Angle Estimate in the Expert's Residual Stream

At the last flow step, the action-row mean of the residual stream entering each of layers 5 to 15 varies across scenes mostly within a plane, which holds 0.83 to 0.92 of its variance, and its position in that plane follows a line in the block angle with r^2 from 0.97 to 0.98 (Table 6, Appendix E). We call the position on that line the stream's estimate of the block angle. Because the queries are linear in the stream and the action rows start from noise, any scene-dependent selection of image content must be written in the stream, and we test whether this estimate is what the queries use. We replace the query-projection input with its mean plus the point on the line for a fixed angle A, and read the error on each side of angle zero, the sign of the block angle, for blocks at least 90° from it (Table 7, Appendix E). The mean alone raises the error on both sides. A point on the block's side, with $|A|$ from 90° to 165°, brings it near the unedited level, one on the opposite side raises it further, and an equal addition in a random subspace stays at the mean's level. Of the 20 heads that attend the block, 10 do so only with A on the block's side, 7 with A on either side, none only with A opposite, and

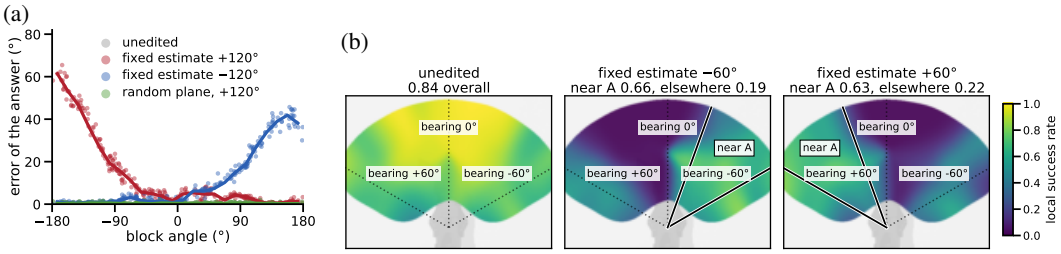


Figure 2: With the estimate of a fixed angle A written into the query weights, the policy keeps working on the side of angle zero containing A and fails on the other. (a) Toy policy, open loop, 288 scenes; dots are scenes, lines medians in 15° bins, and the random plane of equal size is the control. (b) Arm policy, closed loop, success by cube position, 3,000 layouts per map, of which 1,277 ($A = -60°$) and 1,339 ($A = +60°$) lie in the marked band within 40° of A on its side.

3 fit neither pattern (Table 8). The estimate's side therefore gates half of the heads that attend the block, and the wrong-side error is consistent with the policy fetching the wrong image half.

4.2 Writing a Fixed Estimate into the Query Weights

If, as Section 4.1 suggests, the queries of layers 5 to 15 take the block's side from the estimate, a constant estimate should keep the policy accurate on the half containing its angle and break the other half. Each action row's query is its stream row times the query weight W_q, so the mean query over action rows is nW_q with n the mean stream row, and over the scenes used to fit the edit (Appendix E) $n \approx \bar{n} + cB$, with $\bar{n}$ the mean of n over those scenes, c the scene's two-coordinate estimate and B the two-row matrix of stream directions carrying it. We replace W_q by MW_q, with M (Equation 1, Appendix E) chosen so that $nM \approx n - cB + c_A B$ for the estimate c_A at angle A. This writes the estimate of A permanently in these six weight matrices, with no intervention at inference.

On one open-loop action chunk (Figure 2a), $A = +120°$ gives a median error of 2.5° on the half with positive angle, growing with distance from A, against 16.0° on the other, and $A = -120°$ mirrors this. The same construction in a random plane leaves the error unchanged, and removing the estimate without writing an angle raises it on both (Table 9).

4.3 Applying the Query-Weight Edit to a Simulated Arm Policy

The same edit plants a similar failure on a simulated arm in closed loop, at a cost even near the written bearing. We apply it in MuJoCo on the SO-101 arm to a SmolVLA policy fine-tuned to pick a cube from a random position and place it in a tray, with the cube's bearing from the base in place of the block angle. We refit the estimate's plane as in Section 4.1, take $A = \pm 60°$ of bearing, and scale both terms of the edit by a factor λ (Appendix F). With $A = -60°$ success is 0.66 for cubes within 40° of A on its side, where the unedited policy reaches 0.72 to 0.92, and 0.19 elsewhere (Figure 2b); λ was chosen by a rule fixed before the runs on one set of layouts and confirmed on a separate set (Table 10). The mirror edit $A = +60°$ keeps the mirror band and mostly fails where $A = -60°$ works.

5 Limitations & Future Work

The toy task is greatly simplified: one joint, a fixed prompt, one open-loop chunk and one answer angle. Even there we lack a mechanistic account of exactly how the answer is produced. We know what the expert reads and that its estimate gates the later heads, not what the queries compute from it or how the angle is assembled, and the arm result covers one task and one edit without the random-plane control. Each gap is a next step: tasks where the text matters; what the queries do and how the answer is constructed; tasks needing different actions over time, resampled chunks, several joints or state dependence; what the even layers do; and which parts of the model depend on the estimate. The SO-101 transfer suggests these parts may be shared across VLA models built alike.

References

[1] K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations (ICLR)*, 2023.

[2] A. Conmy, A. N. Mavor-Parker, A. Lynch, S. Heimersheim, and A. Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2304.14997.

[3] A. Arditi, O. Obeso, A. Syed, D. Paleka, N. Panickssery, W. Gurnee, and N. Nanda. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.11717.

[4] B. W. Lee, I. Padhi, K. N. Ramamurthy, E. Miehling, P. Dognin, M. Nagireddy, and A. Dhurandhar. Programming refusal with conditional activation steering. In *International Conference on Learning Representations (ICLR)*, 2025.

[5] G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. In *ICLR Workshop*, 2017.

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

[7] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2014.

[8] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *ICML*, 2017.

[9] B. Häon, K. Stocking, I. Chuang, and C. Tomlin. Mechanistic interpretability for steering vision-language-action models. In *Conference on Robot Learning (CoRL)*, 2025. arXiv:2509.00328.

[10] H. Buurmeijer, C. Amo Alonso, A. Swann, and M. Pavone. Observing and controlling features in vision-language-action models. *arXiv preprint arXiv:2603.05487*, 2026.

[11] A. Swann, L. McGranahan, H. Buurmeijer, M. Kennedy, III, and M. Schwager. Sparse autoencoders reveal interpretable and steerable features in VLA models. *arXiv preprint arXiv:2603.19183*, 2026.

[12] M. M. Miao, S. Kim, B. Yang, and L. Ungar. Contrastive conceptor activation steering (COAST): Unlocking vision-language-action models through hidden states. *arXiv preprint arXiv:2605.17144*, 2026.

[13] A. Swann, M. Papillon, M. Jacobs, S. Boppana, T. Haklay, E. Bigelow, A. Lee, C. Watts, T. Icard, M. Schwager, O. Lewis, J. Merullo, E. S. Lubana, A. Geiger, T. Fel, and M. Kowal. Causal analysis of robot foundation models enables efficiency gains. *arXiv preprint arXiv:2610.20750*, 2026.

[14] B. Grant, X. Zhao, and P. Wang. Not all features are created equal: A mechanistic study of vision-language-action models. *arXiv preprint arXiv:2603.19233*, 2026.

[15] S. Jiang, J. Cao, Q. He, F. Deng, K. Wu, J. Sun, J. Wang, Q. Zhang, Q. Zheng, C. Song, P. Luo, and A. F. Luo. Encoded but not in control: Revealing the grounding gap in vision-language robot policies. *arXiv preprint arXiv:2610.06235*, 2026.

[16] S. Hegde and J. Y. B. Katta. GUARD: Grounding uncertainty and ablation-based risk detection for diffusion-based VLAs. *arXiv preprint arXiv:2608.04510*, 2026.

[17] R. Gandikota and D. Bau. Gaze heads: How VLMs look at what they describe. *arXiv preprint arXiv:2606.14703*, 2026.

[18] A. Swann, M. Jacobs, M. Papillon, S. Boppana, D. Chanin, D. Hazra, U. Bhalla, C. Tigges, M. Schwager, O. Lewis, J. Merullo, E. S. Lubana, A. Geiger, T. Fel, and M. Kowal. Multi-dimensional featurizers yield fine-grained steering and data shaping in robot foundation models. *arXiv preprint arXiv:2610.20767*, 2026.

[19] Y. Zhang, Y. Wu, H. Duan, and J. Han. CofactVLA: Deconfounding vision-language-action models via counterfactual intervention. *arXiv preprint arXiv:2608.04396*, 2026.

[20] X. Zhou, G. Tie, G. Zhang, H. Wang, P. Zhou, and L. Sun. BadVLA: Towards backdoor attacks on vision-language-action models via objective-decoupled optimization. *arXiv preprint arXiv:2505.16640*, 2025.

[21] J.-H. Liu, K.-Y. Lin, Y.-L. Wei, X.-H. Chen, Y. Li, Z. Li, Y.-M. Li, Q. Zhang, X. Fan, D. Jiang, Y. Li, and W.-S. Zheng. TrapVLA: Trapping vision-language-action models in configured failure modes. *arXiv preprint arXiv:2608.26578*, 2026.

[22] X. An, T. Luo, G. Peng, Y. Wang, K. Ren, and D. Wang. FlowHijack: A dynamics-aware backdoor attack on flow-matching vision-language-action models. *arXiv preprint arXiv:2604.09651*, 2026.

[23] F. Younesi, W. Lucia, and A. Youssef. Trojan attacks on neural network controllers for robotic systems. *arXiv preprint arXiv:2602.05121*, 2026.

[24] Z. Zhou, J. Zhang, H. Xu, X. Zhang, E. Wen, J. Sun, and H. Jia. TMT: Runtime backdoor detection for vision-language-action policies on unseen tasks. *arXiv preprint arXiv:2610.09462*, 2026.

[25] P. Fu, X. Guo, X. Li, W. Huang, R. Wang, W. Zhao, W. Sui, and M. Ye. TrustVLA: Mechanism-guided inference-time defense against vision-language-action backdoors. *arXiv preprint arXiv:2607.12571*, 2026.

[26] E. Hubinger, C. Denison, J. Mu, M. Lambert, M. Tong, M. MacDiarmid, T. Lanham, D. M. Ziegler, T. Maxwell, N. Cheng, et al. Sleeper agents: Training deceptive LLMs that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.

[27] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, S. Alibert, M. Cord, T. Wolf, and R. Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[28] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems (RSS)*, 2025. arXiv:2410.24164.

[29] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[30] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, H. Li, and T. Kong. Vision-language foundation models as effective robot imitators. In *ICLR*, 2024.

[31] Y. Wang, P. Ding, L. Li, C. Cui, Z. Ge, X. Tong, W. Song, H. Zhao, W. Zhao, P. Hou, S. Huang, Y. Tang, W. Wang, R. Zhang, J. Liu, and D. Wang. VLA-Adapter: An effective paradigm for tiny-scale vision-language-action model. *arXiv preprint arXiv:2509.09372*, 2025.

[32] NVIDIA. Isaac GR00T N1.7. `https://github.com/NVIDIA/Isaac-GR00T`, 2026. Language and visual backbones frozen; projector and diffusion model trained.

[33] P. G. Constantine. *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*. SIAM, 2015.

The appendix holds the tables and figures the body cites, each with the procedure behind it. Every other setting, including the training configuration, scene generation, thresholds and the simulation, is that of the code at https://github.com/czhs/toy-vla.

A Direction Details

The toy task is as in Section 2, with the policy's training configuration given in the code. Table 1 tests the directions of Section 3.1.

Directions. We take the gradient of the output angle with respect to the keys and, separately, the values at the 64 image tokens at every layer of the vision-language model, on 72 fitting scenes with block angles on a grid, at all ten flow steps. Stacking these gradients as rows gives one matrix per layer and kind, and the directions of that layer and kind are its leading right singular vectors, cut at the smallest count whose squared singular values hold 90% of the squared Frobenius norm, an active subspace of the output angle [33].

Stream directions. At layer ℓ the value row of an image token is $v = W_v\,(\gamma_\ell \odot x/\mathrm{rms}(x))$, where x is the token's residual stream entering the layer, γ_ℓ the gain of the layer's input normalisation and W_v the value projection, and the key row is the same with W_k followed by the token's positional rotation R_j. A value direction d in row space therefore corresponds to the stream direction $\gamma_\ell \odot W_v^\top d$, and a key direction to $\gamma_\ell \odot W_k^\top R_j^\top d$ at token j, dropping the scale factor $1/\mathrm{rms}(x)$.

Edits of Table 1. Let X be the key or value rows of the 64 image tokens at one odd layer, M their mean per token position over separate fitting scenes, and P the orthogonal projector onto the directions of that layer and kind. Keeping a subspace alone replaces the rows by $M + (X - M)P$, removing it replaces them by $X - (X - M)P$, and replacing the rows by M is the reference for removing the image entirely. Each edit is applied to both kinds at all eight odd layers and at every flow step. The two controls use a subspace of the same dimension as the directions at each layer and kind, a random orthonormal subspace and the leading principal components of $X - M$ on the fitting scenes.

Table 1: Median error (degrees) on 256 held-out scenes with the components of the image keys and values along a subspace kept alone or removed at the eight odd layers; the subspaces are of equal size.

subspace, equal size	kept alone	removed
intact	0.37	0.37
gradient directions, 67	1.30	60.8
random	79.3	0.42
principal components	4.94	35.5
all rows at their means	82.5	–

B Even-Layer Removal

Skipping the eight even layers leaves a small error, and skipping the eight odd layers loses the output angle, which is why Section 3 studies the odd layers.

Table 2: Median error of the output angle (degrees) with groups of expert layers skipped, the 72 fitting scenes of Appendix A. Skipping a layer removes its attention and MLP contributions to the residual stream.

configuration	median error (°)
intact	0.31
all eight even layers removed, attention and MLP	4.31
all eight odd layers removed	89.1

C Vision-Language Model Edit

Scenes and readout. Each scene draws the digits 1 to 8 at eight positions on the circle traced by the link's tip, on the plain table with no link and no block, with the marked digit and the arrangement drawn at random. The vision-language model is the policy's own frozen SmolVLM2 with its language head, given the image as the policy gives it and the question "Which number is marked in this image? Answer with the number." Its reply is the first digit in the generated text, and accuracy is the fraction of scenes in which that is the marked digit, with chance at 0.125. The layer and strength were chosen on 200 scenes with a longer phrasing of the question (Table 3), and the measurements of Section 3.2 use 4,000 scenes drawn independently of those (Table 4).

The edit. The edit of strength m at layer ℓ adds to the token's residual stream entering that layer the minimum-norm vector in the span of the layer's stream directions that moves the token's key and value projections on the directions by m times a block token's own mean projection, taken relative to the mean projection of that token position. The edit with values only uses the value directions alone, and the edit at all eight odd layers adds each layer's own vector at that layer. The random control adds a random direction with the same norm at the same token and layer.

Table 3: Accuracy at naming the marked digit under an edit at one layer that adds the layer's key and value directions, by layer and strength, with the random control of the same norm in parentheses, on the 200 scenes and the phrasing used for selection. Layer 7 at strength 8 gives the largest margin over its control.

layer	1	2	4	8	12	16
1	0.16 (0.11)	0.28 (0.16)	0.33 (0.16)	0.10 (0.08)	–	–
3	0.20 (0.14)	0.27 (0.16)	0.22 (0.15)	0.08 (0.05)	–	–
5	0.13 (0.12)	0.20 (0.13)	0.35 (0.17)	0.30 (0.08)	0.15 (0.02)	0.07 (0.02)
7	0.13 (0.12)	0.15 (0.12)	0.23 (0.17)	0.60 (0.11)	0.39 (0.05)	0.15 (0.03)
9	0.12 (0.11)	0.13 (0.12)	0.20 (0.16)	0.24 (0.15)	0.10 (0.06)	0.04 (0.03)
11	0.12 (0.11)	0.12 (0.13)	0.17 (0.12)	0.25 (0.17)	–	–
13	0.13 (0.10)	0.17 (0.10)	0.24 (0.13)	0.29 (0.09)	–	–
15	0.12 (0.11)	0.12 (0.12)	0.14 (0.11)	0.19 (0.14)	–	–

Table 4: Accuracy at naming the marked digit, with edits at layer 7 and the question of Section 3.2 on 4,000 scenes. The lower rows give the baselines of Table 3 on the 200 selection scenes, and the edit at all eight odd layers there.

condition	accuracy	random control, same norm
no edit	0.122	–
drawn box	0.975	–
keys and values, strength 8	0.734	0.123
keys only, strength 8	0.811	–
values only, strength 10	0.808	–
values only, strength 12	0.881	–
values only, strength 14	0.891	0.050
values only, strength 16	0.848	–
selection scenes, no edit	0.105	–
selection scenes, drawn box	0.735	–
selection scenes, all eight odd layers, strength 2	0.585	0.180

D Token Detector and Retargeting Edit

Scenes. A retargeting scene is the toy scene with the red block at a random angle and one green and one blue square at other angles on the same circle; there are 2,048 such scenes, and 512 scenes with the red block alone for the cost of the edit. The one-square measurements of Table 5 use 256 scenes with one green or blue square, each rendered once with and once without the square; the edits there replace, at the image tokens of all eight odd layers, the keys and values of the scene with

the square by those of the scene without it, plus or minus the projection of their difference on the directions of Appendix A, and the random control uses a random subspace of the same dimension per layer and kind. The policies trained on green and on blue were fine-tuned as the red policy, on scenes with a single block of their colour.

Detector and edit. The detector flags an image token when its layer-3 key row, relative to the mean key row of that token position over calibration scenes with the link alone, has a large component along the layer-3 key directions of Appendix A, and among the flagged tokens one direction in the layer-3 key space, estimated from calibration squares of each colour, picks out the chosen colour. At each odd layer the edit then removes, at the flagged tokens of the other colours, the component of the key and value rows along that layer's directions, relative to the link-only reference row, and enlarges it at the chosen colour's token. Nothing else in the keys and values is changed; the thresholds and the gain are given in the code.

Table 5: Median error to the red block (degrees) with one distractor square of another colour, 256 scenes, under edits of the keys and values at all 64 image tokens and the eight odd layers built from the difference between the scene with and without the square, and the cost of the retargeting edit on scenes with the red block alone, 512 scenes. Removing the component along the directions nearly undoes the distractor's effect.

no distractor	0.4
one distractor, no edit	27.9
square present, its component along the directions of Section 3.1 removed, all image tokens	1.0
square absent, only that component of it added, all image tokens	22.3
square absent, its component along a random subspace of the same size added	0.4
red block alone, no edit	0.4
red block alone, retargeting edit active	1.1

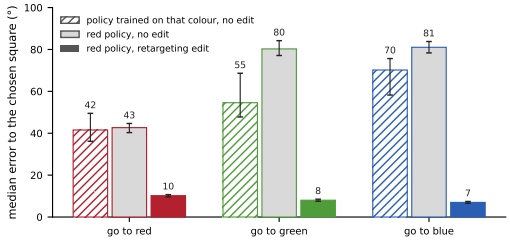


Figure 3: With the retargeting edit, the policy trained on red has the lowest median error to the chosen square for every colour, with a red block and green and blue squares. Solid bars show the edit and grey bars no edit, on 2,048 scenes. Hatched bars show a policy trained on that colour, on the first 128 of them, where the red policy's error is 41.55° against 42.68° on all. Error bars are bootstrap 95% intervals.

E Angle Estimate and Query-Weight Edit

The estimate. At each layer $\ell \in \{5, 7, 9, 11, 13, 15\}$ the two-dimensional subspace is spanned by the two leading principal directions of the row mean of the residual stream entering the layer at the last flow step, over fitting scenes with block angles on a grid, and the coordinate in it is fitted as a line in ϕ (Table 6).

Table 6: Fit of the estimate at each layer over the fitting scenes.

layer	5	7	9	11	13	15
fraction of the scene variance in the subspace	0.86	0.83	0.86	0.90	0.90	0.92
r^2 of the line in ϕ	0.98	0.98	0.97	0.97	0.97	0.98

The replacement test. The test writes, into the input of the query projection of these layers at the last flow step and every action row, the mean of that input over the fitting scenes plus the mean coordinate of the fitting scenes near A mapped through the least-squares B of Equation 1. The random control uses an orthonormal matrix of the same shape drawn at random in place of B, with the coordinate of the same fitting scenes in that subspace.

Table 7: Median error (degrees) with the query input of layers 5 to 15 replaced at the last flow step by the mean over the fitting scenes plus the estimate of a fixed angle A, on the 128 test scenes with $|\phi| \geq 90°$, by the side of the block. The lower rows give the controls.

A	-150	-105	-60	-30	0	30	60	90	105	120	135	150	165	180
$\phi > 0$	51.3	48.1	39.6	30.8	20.8	14.1	7.5	0.9	1.1	0.9	0.9	1.1	1.7	19.9
$\phi < 0$	1.0	0.7	1.3	6.5	14.7	19.6	27.5	44.1	51.1	59.6	62.2	65.9	67.3	14.5

Control	$\phi > 0$	$\phi < 0$
Unedited policy	0.3	0.4
Mean over the fitting scenes alone, no estimate written	24.2	11.0
Same estimate in a random subspace, range over $A \in \{-150, 0, 90, 150\}$	24.1 to 24.2	10.8 to 11.1

The heads. The head set holds the 20 heads at these layers that attend the block tile on the test scenes, chosen by a rule given in the code. A head's own side is the side of the circular mean of the block angles on which it attends, and the written estimate counts as on the own side or the far side when its angle lies well within that side.

Table 8: Heads of the head set by how the written estimate gates their attention to the block, using median block mass with the estimate on each side. Half of the 20 attend only on their own side, and the heads not listed fit neither rule.

behaviour	rule	heads
attends only with the estimate on its own side	far side at most 0.1, own side at least 0.5	10
attends on both sides	both at least 0.5	7
attends only with the estimate on the far side	far side at least 0.5, own side at most 0.1	0

The weight edit. Here n is the row mean over the action rows of the input to the query projection, the output of the layer's input normalisation. B and $\bar{n}$ are fitted at each layer by least squares of n on $[1, c]$ over the fitting scenes at the last flow step, where c is the scene's coordinate in the subspace above, and c_A averages c over the fitting scenes near A. With P the projector onto the rows of B and $g = \bar{n}/\|\bar{n}\|^2$, so that $ng^\top \approx 1$ for every action row, the edit replaces W_q by MW_q with

$$M = (I - P) + g^\top (\bar{n}P + c_A B) . \tag{1}$$

The random control replaces B by an orthonormal matrix of the same shape drawn at random, with c_A the mean coordinate of the same fitting scenes in that subspace, and the edit that removes the estimate without writing an angle uses $M = (I - P) + g^\top \bar{n} P$.

Table 9: Median error (degrees) of the toy policy by the side of the block on 288 test scenes, 128 with $\phi > 0$ and 160 with $\phi < 0$, for the weight edit of Equation 1. Each edit keeps the error lower on the side of its angle, and the random subspace changes nothing. Figure 2a plots the same data as medians in 15° bins.

	$\phi > 0$	$\phi < 0$
unedited	0.3	0.4
$A = +120°$	2.5	16.0
$A = -120°$	18.1	0.9
random subspace, $A = +120°$	0.3	0.4
estimate removed, no angle written	7.9	1.7

F Arm Policy Details

Task and policy. The arm is a SO-101 in MuJoCo with one fixed camera, a cube placed at a random position with bearing within ±60° of the arm's forward axis, and a fixed tray. The policy is a SmolVLA fine-tuned on demonstrations of the task, with the vision-language model untouched, run in closed loop, and an episode succeeds if the cube is placed in the tray within the time limit; the simulation and its settings are given in the code.

The edit. The construction of Equation 1 is applied to the query weights of layers 5 to 15 with the removal and the write scaled by λ, so that $M = I - \lambda P + \lambda g^\top(\bar{n}P + c_A B)$. The subspace, B and $\bar{n}$ are fitted at the last flow step on the first frames of the fitting episodes, g is fitted by least squares to $ng^\top \approx 1$, and c_A averages the coordinate of fitting frames whose cube bearing is near A. The band of bearings near A is $[-60°, -20°]$ for $A = -60°$ and $[20°, 60°]$ for $A = +60°$. λ was chosen among candidates on 200 layouts by a rule fixed before the runs, taking the candidate with success at least 0.70 in that band and at most 0.20 elsewhere, or otherwise the largest difference between the two; no candidate met the first condition, so the second branch chose λ, and that value was then run once on 200 separate layouts. The 3,000-layout runs of Figure 2b use the same edited weights.

Table 10: Closed-loop success on the arm by cube bearing (degrees), 3,000 layouts per condition with shared layouts. The lower rows give success in the band near A and elsewhere, on the 200 layouts used to choose λ and the 200 separate layouts used to confirm it.

	$[-60, -40)$	$[-40, -20)$	$[-20, 0)$	$[0, 20)$	$[20, 40)$	$[40, 60]$	all
unedited	0.72	0.92	0.97	0.91	0.91	0.74	0.84
$A = -60°$	0.69	0.62	0.17	0.01	0.04	0.40	0.39
$A = +60°$	0.42	0.12	0.01	0.11	0.60	0.66	0.40

	layouts choosing λ, 200	layouts confirming it, 200
unedited	0.79 / 0.79	0.78 / 0.82
$A = -60°$	0.67 / 0.21	0.73 / 0.16