Dissecting Egocentric Action Pretraining in Vision-Language-Action Models

Tommaso Felice Banfi* **Francesco Dorati†**

Jacob Forrestor Valdez† **Lorenzo Ortolani** **Giacomo Galbiati** **Gabriel Voss**

Talos Robotics AI
Milan, Italy
tommasofelice.banfi@talosrobotics.ai

Abstract

Vision-language-action models (VLAs) are built on vision-language models (VLMs) on the assumption that two properties carry over: grounding in the scene and a language interface for steering the robot. We compare one backbone before and after egocentric action pretraining and follow its states into the action expert, a flow-matching policy. Action pretraining reallocates the roles of the layers: the VLM reads the image in its first half, whereas the action-pretrained backbone fuses the scene only in its last quarter, attends to objects there, and forms action tokens at the final layer. Language loses its causal role inside the network: internal states can no longer be read as words, and the instruction direction no longer steers the decision. The new scene code is spatial and object-level: visual tokens form coherent regions and object positions are gained, whereas object identity and parts are not encoded. The action expert hardly uses this code and it acts mostly from the robot state, and a safety correction added to its output moves the action, whereas a change of its attention does not. Language is therefore a second-class input, and steering and safety signals have to reach what the action expert reads.

1 Introduction

Vision-language-action models (VLAs) reuse a vision-language model (VLM) as their backbone [Black et al., 2026, Physical Intelligence et al., 2025, Wei et al., 2026a], betting that two abilities carry over: *grounding*, the VLM's knowledge of what is in a scene and where, and *language as an interface*, through which a robot can be instructed, corrected [Belkhale et al., 2024, Shi et al., 2024] or adapted in context [Sridhar et al., 2025]. Behavioural studies suggest the bet only partly pays off: VLAs often ignore their instructions and follow visual priors [Fei et al., 2025, Grant et al., 2026], and action fine-tuning lowers the VLM's benchmark scores [Driess et al., 2025, Hancock et al., 2026, Kachaev et al., 2025]. They show that something is lost, but not where, what replaces it or what the action reads, and thus whether to fix the backbone, the action head or their interface.

We answer these questions mechanistically, comparing a VLA's backbone with its source VLM before and after egocentric action pretraining, and following its representations into the action expert. We find that (i) **action pretraining reallocates the roles of the layers**: the VLM integrates the image early, the action-pretrained backbone only in its last layers, where it builds a new scene code; (ii) **language becomes a second-class input**, no longer word-like inside the network and without steering effect on the decision; (iii) **the scene code is spatial and object-level**, without identity or

*Corresponding author. † Work done in collaboration with Talos Robotics AI.

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

parts; and (iv) **the action does not use this code**: the expert relies on the robot state and early layers, and a correction added to its output moves the action whereas an attention bias does not. A second VLA shows the same loss of language and reliance on the state. Steering and safety signals should therefore be routed to what the action reads.

2 Related work

What VLAs keep from the VLM. Remedies for the erosion of the VLM block action gradients [Driess et al., 2025], express actions as text [Hancock et al., 2026] or align visual features [Kachaev et al., 2025], and measure it with benchmarks; we localise it by layer and show where capacity goes.

Language and steering. VLAs ignore instruction changes [Fei et al., 2025, Hou and Zhao, 2026] and follow vision [Grant et al., 2026], while steering via action tokens [Chan and Kao, 2026] or feed-forward directions [Häon et al., 2025] works. We show why, and which path reaches it.

Interpretability of VLMs and VLAs. VLMs move visual information into text positions in early and middle layers [Zhang et al., 2024, Kaduri et al., 2025]; VLA studies analyse a single trained model [Häon et al., 2025, Swann et al., 2026, Grant et al., 2026]. We diff the same backbone before and after action pretraining [Lindsey et al., 2024, Kassem et al., 2026] and follow it into the expert.

3 Experimental setup

Models. The base model is Qwen3-VL-2B [Bai et al., 2025]. The action-pretrained backbone is the stage-1 VLM of the humanoid VLA Ψ_0 [Wei et al., 2026a], trained on egocentric human video [Hoque et al., 2026] to predict FAST (frequency-space action sequence tokenization) action tokens [Pertsch et al., 2025] (Ψ_0's *pre-training*). Its flow-matching action expert [Lipman et al., 2023], trained afterwards on the frozen backbone (Ψ_0's *post-training*), reads the last-layer and robot states and outputs an action chunk of 30 joint targets for a Unitree G1 on the SIMPLE bend-and-pick task [Wei et al., 2026b]. The *decision token* is the last prompt token. Appendix C covers a second VLA, $\pi_{0.5}$ [Physical Intelligence et al., 2025].

Data. The UMD part-affordance dataset [Myers et al., 2015] with 340 goal-level instructions that never name the affordance; simulated frames from training and unseen episodes; and 280 counterfactual queries in 30 scenes (obstacle moved, removed or replaced), each with a control-barrier-function (CBF) correction [Ames et al., 2019, Morton and Pavone, 2025].

Measurement. Action effects are in units of the *noise floor* (nf), the change from resampling flow noise. Intervals are 95 % cluster bootstraps over scenes, instances or instructions (Appendix A).

4 Results

4.1 Action pretraining reallocates the roles of the layers

We block the decision token's access to the image in layers $1..k$, or from layer k on, and measure the next-token Kullback–Leibler divergence (KL) from the unblocked run (Figure 1). The base model has read the image by layer 12 and no longer needs it after layer 15 (KL 0.04), as in other VLMs [Zhang et al., 2024, Kaduri et al., 2025]. The action-pretrained backbone reads it between layers 20 and 22 (KL 1.2 $\rightarrow$ 3.7) and still needs it at layer 27 (KL 1.9) (magnitudes are not comparable across the two models, whose predicted tokens differ). Attention, a descriptive measure [Jain and Wallace, 2019], agrees: from layer 4 the decision token spends 53–77 % of its attention on the image, with no attention sink [Xiao et al., 2024], and it concentrates on objects only in layers 23–28, peaking at layer 27 (3.7$\times$ above uniform on UMD, 6.3$\times$ on the targets of our scenes). Projected onto the action vocabulary, no layer matches the final action token before layer 28; in the weights every language-model projection changed (cosine 0.54–0.64) and the vision tower barely (0.97).

Action pretraining thus reassigns the layers' roles: the second half of a VLM turns visual evidence into language, whereas here the same layers compute where things are relative to the hands, late and next to the output that needs it. The scene code forms in layers 20–28, so measures that protect the VLM's knowledge (insulating gradients, freezing, regularisation) should target them.

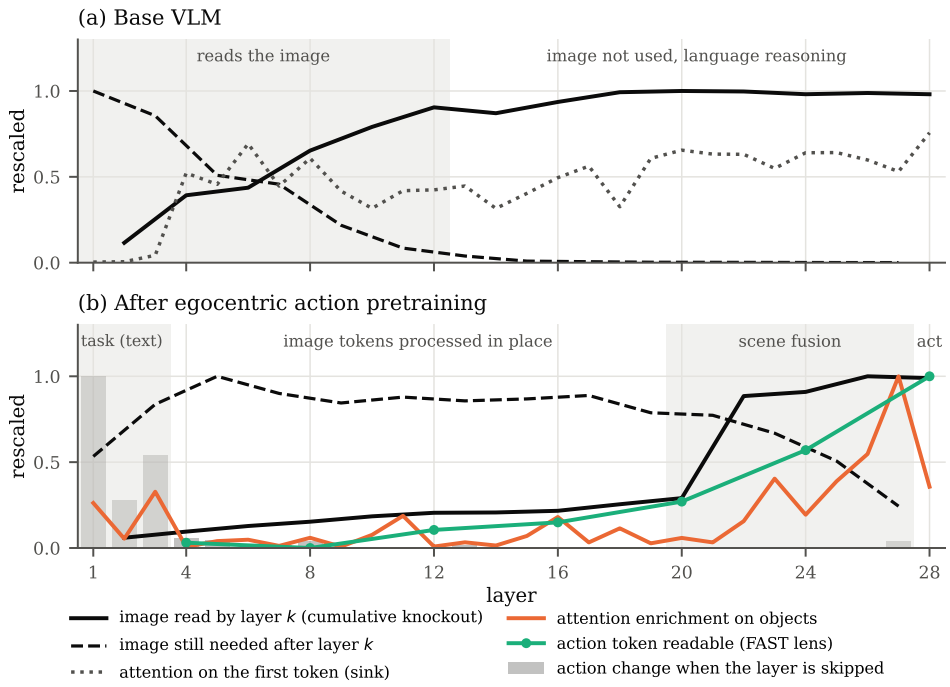

Figure 1: **Where each model uses the image across layers.** Next-token change when the image is blocked in layers $1..k$ (solid) or from layer k on (dashed), attention to the first token and to objects, readability of the action token, and the action change when a layer is skipped (bars); all rescaled to $[0, 1]$. (a) Base VLM. (b) Action-pretrained backbone.

4.2 Language becomes a second-class input

Language is lost inside the network. The action-pretrained states are not language-like at any layer: the likelihood of a fixed sentence drops from 2.9 to 30.9 nats per token, worse than a uniform guess (11.9). Patchscopes [Ghandeharioun et al., 2024], which make the base model read a hidden state as a word, recover a visual word from the base model's object tokens in up to 54 % of cases and from the action-pretrained ones in at most 1 %, and the base continues from the action-pretrained states (stitching, Bansal et al., 2021) only at a cost of 0.8–2.3 nats; the change is present at layer 4.

The instruction is still encoded, since its verb and the requested hand can be decoded at every layer, but its direction no longer steers the decision. The decision token predicts the affordance an instruction asks for with 80–84 % accuracy, against 94–97 % in the base (instructions and probe follow Dorati and Banfi, 2026). In the causal test we add to the decision token the difference between its states under two instructions [Turner et al., 2024]. In the base model this reproduces the effect of actually changing the instruction (KL 0.072 vs. 0.073) and is 4× stronger than a random direction at layer 24; in the action-pretrained backbone it equals a random direction at every layer. The instruction's content survives without the image, but the words no longer point at their objects (Appendix E.4).

This is consistent with VLAs ignoring their instructions [Fei et al., 2025, Hou and Zhao, 2026], and it has a practical consequence: any method that steers the policy through the prompt, from language corrections [Shi et al., 2024] to in-context examples [Sridhar et al., 2025], depends on a pathway that action pretraining has weakened. Methods that protect language during action training [Driess et al., 2025, Hancock et al., 2026] should be evaluated with this causal test as well as with benchmarks.

4.3 The scene code is spatial and object-level

The new code encodes where objects are, and less well what they are. Probing ten scene properties at seven depths, 43 of 70 cells are kept, 19 gained and 7 lost (Figure 2). The losses are object identity (-0.35 [-0.39, -0.30] at layer 28, growing across layers); the gains are spatial and appear in the last

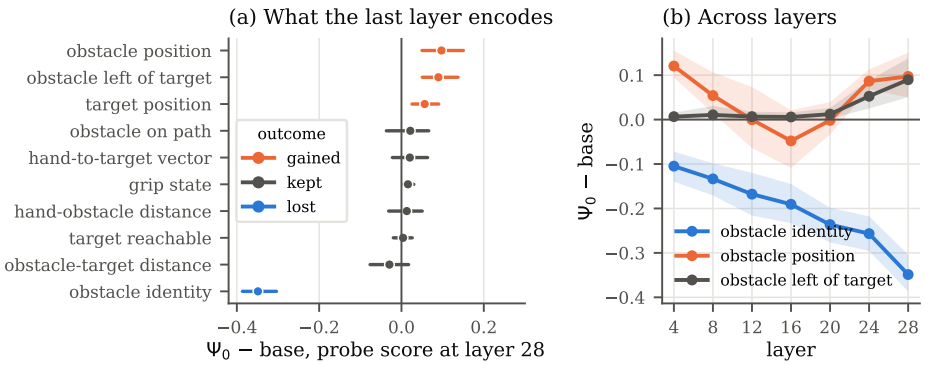

Figure 2: **Probe scores, action-pretrained minus base.** (a) Ten scene properties at layer 28. (b) Three properties across layers. A property is kept, gained or lost when the 95 % scene-level interval of the difference lies within, above or below ± 0.05.

layers: obstacle position ($+0.10$), whether the obstacle is left of the target ($+0.09$) and target position ($+0.06$). The decision token also carries object layout (a 4×4 object mask is decoded with R^2 0.69 vs. 0.40 on UMD, and 0.32 vs. 0.12 for the obstacle in our scenes), and at the last layer moving an object shifts its states along one direction shared across scenes (Appendix E.6); it behaves like a coordinate. The code does not reach parts. On UMD, where an instruction asks for one part of an object, blocking or graying the instructed part changes the decision token as much as another part of the same object (KL 1.141 vs. 1.144), while blocking the whole object has $5\times$ the effect of an equal background area (Appendix E.2). A part map can be decoded, but a prior that knows only the object category and the requested affordance predicts it better. The backbone has likely learned what egocentric hand motion needs, the position of objects relative to the body, and does not encode what they are or how each part is used. Human video can thus supply layout, a good basis for reaching and avoidance; tasks that hinge on identity or parts (which tool, which handle) need that content from labels, an explicit affordance model or auxiliary objectives, and gains on position-dependent tasks should not be read as semantic grounding [Yu et al., 2026].

4.4 The action expert bypasses the scene code

The action expert hardly uses what the backbone builds (Appendix D). Removing the robot state changes the action by $51\times$ nf, whereas removing the image tokens or changing the instruction changes it by at most $2.5\times$ nf, and only the first three backbone layers matter when skipped. On training episodes the image does matter ($19.5\times$ nf), but only to recognise the episode: on unseen episodes the effect drops to $2.7\times$ nf. Decodable information need not be used [Elazar et al., 2021]: the obstacle's position is linearly decodable from every layer, yet moving it barely changes the action ($1.3\times$ nf). Adding a CBF safety correction [Morton and Pavone, 2025] to the expert's output raises clearance by 10.4 mm [6.7, 13.7], also when the correction is predicted from the backbone's own features, while boosting the expert's attention to the obstacle has no effect ($+0.1$ mm [-0.4, 0.8] at the strongest boost). Trained later on a frozen backbone with little data, the expert takes the cheapest informative inputs, the robot state and, on familiar data, the episode's identity. Evaluations on training episodes therefore overstate perception; same-episode image shifts and state removal should be standard checks, and use of the scene should be audited during training (Appendix F). Task or safety information can instead be added to the expert's output without retraining the backbone.

5 Conclusion

Egocentric action pretraining reallocates the roles of a VLM's layers. The layers that turned images into words build a late, spatial, object-level scene code, the instruction loses its effect along the way, and the action expert trained afterwards bypasses that code: backbone and action serve different readers, so what the backbone represents the action need not use. Four recommendations follow. (i) Protect the language pathway during action training and verify it with causal tests as well as benchmarks. (ii) Steer through what the policy reads: action tokens, the expert's output or the image.

(iii) Supply identity and part-level content from outside the backbone, and train the action to depend on it, for example with counterfactual scenes and use gates. (iv) A safety correction added to the expert's output moves the action without retraining; it complements output filters. The diagnostics used here can serve as training-time checks (Appendix F); limitations are in Appendix B.

References

A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications, 2019.

S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge, W. Ge, Z. Guo, Q. Huang, J. Huang, F. Huang, B. Hui, S. Jiang, Z. Li, M. Li, M. Li, K. Li, Z. Lin, J. Lin, X. Liu, J. Liu, C. Liu, Y. Liu, D. Liu, S. Liu, D. Lu, R. Luo, C. Lv, R. Men, L. Meng, X. Ren, X. Ren, S. Song, Y. Sun, J. Tang, J. Tu, J. Wan, P. Wang, P. Wang, Q. Wang, Y. Wang, T. Xie, Y. Xu, H. Xu, J. Xu, Z. Yang, M. Yang, J. Yang, A. Yang, B. Yu, F. Zhang, H. Zhang, X. Zhang, B. Zheng, H. Zhong, J. Zhou, F. Zhou, J. Zhou, Y. Zhu, and K. Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

T. F. Banfi, L. Ortolani, M. Menzolini, G. Galbiati, and G. Voss. Agentic VLM for long-horizon planning in humanoid physical AI agent orchestration. In *2nd Workshop on Multimodal Reasoning and Slow Thinking in the Large Model Era: Towards System 2 and Beyond*, 2026.

Y. Bansal, P. Nakkiran, and B. Barak. Revisiting model stitching to compare neural representations. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 225–236. Curran Associates, Inc., 2021.

S. Belkhale, T. Ding, T. Xiao, P. Sermanet, Q. Vuong, J. Tompson, Y. Chebotar, D. Dwibedi, and D. Sadigh. RT-H: Action hierarchies using language, 2024.

L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, T. Unterthiner, D. Keysers, S. Koppula, F. Liu, A. Grycner, A. Gritsenko, N. Houlsby, M. Kumar, K. Rong, J. Eisenschlos, R. Kabra, M. Bauer, M. Bošnjak, X. Chen, M. Minderer, P. Voigtlaender, I. Bica, I. Balazevic, J. Puigcerver, P. Papalampidi, O. Henaff, X. Xiong, R. Soricut, J. Harmsen, and X. Zhai. Paligemma: A versatile 3b vlm for transfer, 2024.

K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026.

J. Chan and J. C. Kao. Steering autoregressive vision-language-action policies via action token intervention, 2026.

T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski. Vision transformers need registers, 2024.

F. Dorati and T. F. Banfi. Foundation ViTs can learn where to act: Task affordance segmentation and surface normals. In *ECCV 2026 Workshop on Open-World 3D Scene Understanding and Representations*, 2026.

D. Driess, J. T. Springenberg, B. Ichter, L. Yu, A. Li-Bell, K. Pertsch, A. Z. Ren, H. Walke, Q. Vuong, L. X. Shi, and S. Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better, 2025.

Y. Elazar, S. Ravfogel, A. Jacovi, and Y. Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals, 2021.

S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, L. Ji, X. He, S. Zhang, Z. Fei, J. Fu, J. Gong, and X. Qiu. Libero-plus: In-depth robustness analysis of vision-language-action models, 2025.

A. Ghandeharioun, A. Caciularu, A. Pearce, L. Dixon, and M. Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 15466–15490. PMLR, 2024.

B. Grant, X. Zhao, and P. Wang. Not all features are created equal: A mechanistic study of vision-language-action models, 2026.

A. Gromov, K. Tirumala, H. Shapourian, P. Glorioso, and D. A. Roberts. The unreasonable ineffectiveness of the deeper layers, 2025.

J. Gu, S. Kirmani, P. Wohlhart, Y. Lu, M. G. Arenas, K. Rao, W. Yu, C. Fu, K. Gopalakrishnan, Z. Xu, P. Sundaresan, P. Xu, H. Su, K. Hausman, C. Finn, Q. Vuong, and T. Xiao. RT-Trajectory: Robotic task generalization via hindsight trajectory sketches, 2023.

A. J. Hancock, X. Wu, L. Zha, O. Russakovsky, and A. Majumdar. Actions as language: Fine-tuning VLMs into VLAs without catastrophic forgetting. In *The Fourteenth International Conference on Learning Representations*, 2026.

B. Häon, K. C. Stocking, I. Chuang, and C. Tomlin. Mechanistic interpretability for steering vision-language-action models. In J. Lim, S. Song, and H.-W. Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 2743–2762. PMLR, 2025.

R. Hoque, P. Huang, D. J. Yoon, M. Sivapurapu, and J. Zhang. Egodex: Learning dexterous manipulation from large-scale egocentric video, 2026.

Y. Hou and L. Zhao. Langgap: Diagnosing and closing the language gap in vision-language-action models, 2026.

S. Jain and B. C. Wallace. Attention is not explanation, 2019.

N. Kachaev, M. Kolosov, D. Zelezetsky, A. K. Kovalev, and A. I. Panov. Don't blind your vla: Aligning visual representations for ood generalization, 2025.

O. Kaduri, S. Bagon, and T. Dekel. What's in the image? a deep-dive into the vision of vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14549–14558, 2025.

M. Kang, L. Shi, L. Liang, R. Gan, D. Liu, P. Zhang, S. Chen, S. Qin, Y. Zheng, J. Zheng, H. Wang, X. Zhan, and H. Su. X-tokenizer: A multimodal action tokenizer for vision-language-action pretraining, 2026.

A. Kassem, T. Jiralerspong, N. Rostamzadeh, and G. Farnadi. Delta-crosscoder: Robust crosscoder model diffing in narrow fine-tuning regimes, 2026.

J. Lindsey, A. Templeton, J. Marcus, T. Conerly, J. Batson, and C. Olah. Sparse crosscoders for cross-layer features and model diffing. Transformer Circuits Thread, 2024.

Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling, 2023.

J. Minder, C. Dumas, C. Juang, B. Chughtai, and N. Nanda. Overcoming sparsity artifacts in crosscoders to interpret chat-tuning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 106423–106474. Curran Associates, Inc., 2025. doi: 10.52202/085713-3551.

D. Morton and M. Pavone. Safe, task-consistent manipulation with operational space control barrier functions. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 187–194, 2025. doi: 10.1109/IROS60139.2025.11246389.

D. Morton, P. Mohnot, and M. Pavone. Constrained whole-body tracking for humanoid robots, 2026.

A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos. Affordance detection of tool parts from geometric features. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1374–1381, 2015.

D. N. Olivieri and A. F. P. Rodríguez. Transformer field theory: A response-theoretic approach to mechanistic interpretability, 2026.

K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models, 2025.

Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025.

L. X. Shi, Z. Hu, T. Z. Zhao, A. Sharma, K. Pertsch, J. Luo, S. Levine, and C. Finn. Yell at your robot: Improving on-the-fly from language corrections. In *Robotics: Science and Systems (RSS)*, 2024.

K. Sridhar, S. Dutta, D. Jayaraman, and I. Lee. RICL: Adding in-context adaptability to pre-trained vision-language-action models. In J. Lim, S. Song, and H.-W. Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 5022–5038. PMLR, 2025.

M. Sun, X. Chen, J. Z. Kolter, and Z. Liu. Massive activations in large language models, 2024.

A. Swann, L. McGranahan, H. Buurmeijer, M. Kennedy III, and M. Schwager. Sparse autoencoders reveal interpretable and steerable features in vla models, 2026.

A. M. Turner, L. Thiergart, G. Leech, D. Udell, J. J. Vazquez, U. Mini, and M. MacDiarmid. Steering language models with activation engineering, 2024.

S. Wei, H. Jing, B. Li, Z. Zhao, J. Mao, Z. Ni, S. He, J. Liu, X. Liu, K. Kang, S. Zang, W. Yuan, M. Pavone, D. Huang, and Y. Wang. ψ_0: An open foundation model towards universal humanoid loco-manipulation, 2026a.

S. Wei, Z. Ni, J. Liu, Z. Zhao, J. Ye, H. Jing, J. Xia, X. Liu, M. Leong, L. Heng, D. Huang, and Y. Wang. Simple: Simulation-based policy learning and evaluation for humanoid loco-manipulation, 2026b.

G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks, 2024.

Q. Yu, J. You, Y. Wang, J. Liang, B. Ping, Y. Tian, Y. Chen, M. Cai, Z. Gong, R. Wu, Y. Li, J. Liang, and Y. Chen. Affordancevla: A vision-language-action model empowering action generation through affordance-aware understanding, 2026.

Y. Yue, Y. Wang, B. Kang, Y. Han, S. Wang, S. Song, J. Feng, and G. Huang. DeeR-VLA: Dynamic inference of multimodal large language models for efficient robot execution, 2024.

Z. Zhang, S. Yadav, F. Han, and E. Shutova. Cross-modal information flow in multimodal large language models. *arXiv preprint arXiv:2411.18620*, 2024.

R. Zheng, Y. Liang, S. Huang, J. Gao, H. Daumé III, A. Kolobov, F. Huang, and J. Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies, 2025.

Contents

1 Introduction **1**

2 Related work **2**

3 Experimental setup **2**

4 Results **2**

4.1 Action pretraining reallocates the roles of the layers 2

4.2 Language becomes a second-class input 3

4.3 The scene code is spatial and object-level 3

4.4 The action expert bypasses the scene code 4

5 Conclusion **4**

A Implementation details **9**

A.1 Models and inputs 9

A.2 Data 9

A.3 Measuring representations 9

A.4 Measuring information flow 9

A.5 Interventions on the action 10

A.6 Statistics and compute 10

B Extended discussion and limitations **10**

C Generalisation: a second VLA and unseen episodes **12**

D What the action expert uses **12**

D.1 Interventions on the inputs 12

D.2 Backbone layers the action depends on 13

D.3 Output injection versus attention bias 13

E Additional results **14**

E.1 Attention to objects across layers 14

E.2 Part specificity 17

E.3 Dense probes on the visual tokens 17

E.4 Instruction content and naming 18

E.5 Geometry of the backbone 19

E.6 Crosscoder features 21

E.7 Flow expert dynamics 22

E.8 Activation editing 22

F From diagnosis to VLA training: structured latent priors **23**

A Implementation details

A.1 Models and inputs

The base model is `Qwen/Qwen3-VL-2B-Instruct` (28 decoder layers, hidden size 2048). The action-pretrained backbone is Ψ_0's stage-1 VLM with the same architecture. The simulated-humanoid analyses use the backbone inside the released SIMPLE bend-and-pick checkpoint of `USC-PSI-Lab/psi-model` (`g1wholebodybendpick-v0`, step 40k), whose configuration loads stage-1 checkpoint `pre.fast.mixed.1by1.2601091803` and keeps it frozen; the UMD analyses use the released stage-1 checkpoint `pre.fast.1by1.2601091803.ckpt.ego200k.he30k`, which carries the same run identifier. The action expert has six joint-attention blocks (hidden size 1536, 0.5B parameters) and samples with ten Euler steps of a flow-matching velocity field [Lipman et al., 2023]; it reads all 96 last-layer backbone states and one robot-state token and outputs 30 steps of a 36-dimensional command (hands, arms, waist, height, locomotion). The second VLA, $\pi_{0.5}$, is the `openpi` model fine-tuned on the same task; its expert reads the keys and values of every backbone layer, and the robot state enters as discretised text tokens.

Both backbones receive the same pixels and the same chat template. In simulation the egocentric image yields 77 visual tokens (a 7×11 grid after the 2×2 merger); on UMD the image is resized to 512×384, giving a 12×16 grid. Image tokens precede the text, so under causal attention they cannot depend on the instruction. The decision token is the last prompt position. All attention analyses use eager attention and average over heads unless stated.

A.2 Data

The UMD part-affordance dataset [Myers et al., 2015] provides tool images with pixel-level part labels for seven affordances (grasp, cut, scoop, contain, pound, support, wrap-grasp). We pair its frames with 340 goal-level instructions, ten per object-affordance pair, none of which names the affordance; affordance readouts are split by instruction and spatial readouts by tool instance. The simulated data are frames from the Isaac-rendered training episodes of the bend-and-pick task, from teleoperated episodes of the same task that were not used for training, and from MuJoCo re-renders of the same scenes. The counterfactual obstacle set contains 280 queries in 30 scenes; for each query the obstacle is moved by ±6 or ±12 cm, removed or replaced by a distractor, or the target is moved, and for every variant we store backbone states, attention, the action, the CBF barrier and the CBF correction obtained by solving a quadratic program (CBF-QP). Splits are by scene.

A.3 Measuring representations

Linear probes are ridge or logistic regressions with regularisation chosen by cross-validation inside the training scenes and evaluated on held-out scenes. A property is *kept*, *gained* or *lost* when the 95 % interval of the paired difference between the two models lies within ±0.05, above $+0.05$ or below -0.05, and *unresolved* otherwise. To read states as language we use three tools: a logit lens restricted to the action vocabulary (final norm and the FAST rows of the unembedding applied to a layer's decision-token state), Patchscopes [Ghandeharioun et al., 2024] (a state written into an identity few-shot prompt of the base model at the same layer, decoded as the next word and counted as visual when it names a colour, an object or a shape), and stitching [Bansal et al., 2021] (the base model's upper layers continue from the action-pretrained states, scored by the increase in caption negative log-likelihood, NLL). Both run through the base model's own head and layers, so they do not depend on the action-pretrained head. The text NLL is the mean per-token negative log-likelihood of one English sentence without image, computed with each model's tied embedding as the unembedding; it is 9.0 and 11.8 at 25 and 50 % of the linear interpolation between the weights of the two models. KL and NLL values are not compared across models; each contrast is within a model, against a random direction of equal norm or against the unablated run. Model diffing uses a BatchTopK crosscoder with a decoder-norm penalty trained jointly on both models' image-token states at layers 16 and 28 [Lindsey et al., 2024, Minder et al., 2025].

A.4 Measuring information flow

Image knockouts set the attention from the decision token and the text positions to all image tokens to zero, either in layers $1..k$ or from layer k to the last layer, and report the Kullback–Leibler divergence

(KL) of the next-token distribution. Attention enrichment is the share of the decision token's attention on a region divided by the region's share of image area, so that 1 means uniform. A few image tokens receive attention in every frame regardless of content (visual sinks, as with registers in vision transformers [Darcet et al., 2024]); we identify them at layer 4 as tokens with more than five times the uniform share averaged over frames and exclude them from the attention maps of Appendix E.1. A layer skip replaces one decoder layer by the identity.

A.5 Interventions on the action

Every action effect is expressed in units of the noise floor: for each frame we sample two actions with different flow noise and take the mean absolute difference of the normalised action. Activation steering adds a direction to the residual stream at one layer and compares it with a random direction of equal norm; next-token readouts use the final-norm output. For the safety study, the barrier is the signed distance between the hand spheres and the obstacle, computed with forward kinematics of the predicted joint targets, and the correction is the CBF-QP solution for the chunk [Ames et al., 2019], with the barrier defined on the hands in operational space as in Morton and Pavone [2025]. Output injection adds the correction to the expert's velocity output at the second-to-last of ten flow steps (step 8, counting from 0; equivalent to a change of its final residual). A velocity injection at steps 0–7 survives to the final chunk with gain 0.02–0.08, against 1.0 at step 8 and 0.73 at step 9. Attention boosting adds a constant to the attention logits from the action tokens to the obstacle tokens. The distilled variant fits a linear map from layer-16 obstacle features and the state to the correction on training scenes.

A.6 Statistics and compute

Intervals are 95 % cluster bootstraps with 2,000 resamples over the independent units (scenes, tool instances or instructions); paired comparisons use randomisation tests with 5,000 permutations, with Benjamini–Hochberg correction over layers and heads. All forward passes, probes, singular value decompositions and bootstraps run on a GPU: a single RTX 4050 (6 GB) for the simulated analyses and about 25 A40 GPU-hours for the UMD analyses. Code, probing scripts and the counterfactual queries are released with this paper.

B Extended discussion and limitations

One backbone, two readers. The results fit a single account, which we did not test by ablating the pretraining data or objective. Egocentric action pretraining trains the backbone for one reader: the stage-1 action-token head, which predicts human hand motion in the camera frame. That objective needs the layout of objects relative to the hands at the last layer, and does not need their names or language. The backbone accordingly moves scene integration to its last quarter, builds a spatial object code there, and lets language and identity decay. The robot's action expert is a second reader, trained later on a frozen backbone with far less data. It does not read the code built for the first reader and takes a cheaper solution: the robot state, the global structure set up in the first layers, and, on training episodes, the identity of the episode. Representation and use come apart because they were optimised for different readers. The same failure appears in another architecture: a structured head trained on this VLA behaved as a state-only policy (Appendix F).

Steering. Instructions, corrections and in-context examples assume that text changes internal states in a specific way. Here the instruction is still encoded but has lost its causal effect: removing or replacing it moves the action barely more than sampling noise (1.5–2.3$\times$ nf), and steering the backbone along the instruction direction is no different from steering along a random one. In these models language is a second-class input. This is consistent with the behavioural finding that VLAs ignore their instructions [Fei et al., 2025, Hou and Zhao, 2026, Grant et al., 2026], and it indicates where steering should act instead: where the policy reads. In action space, this means intervening on action tokens [Chan and Kao, 2026], on backbone directions that have a causal effect on the action [Häon et al., 2025], or on the expert's output, as our CBF injection does (Appendix D.3). In visual space, it means marking the image, for example with sketched trajectories or visual traces [Gu et al., 2023, Zheng et al., 2025]; the action-pretrained backbone attends to objects and the image drives the action on familiar scenes, but we did not test visual steering directly. Methods that protect the

language pathway during action training [Driess et al., 2025, Hancock et al., 2026] should be judged by internal measures such as the instruction-direction test as well as by benchmarks.

Grounding. The action-pretrained backbone encodes object positions better than the VLM it came from and attends to objects in the right layers; it lacks part-level and identity information, and a reader that uses what it has. Grounding a VLA therefore has two requirements: the right content in the backbone (an explicit affordance model can supply the parts: a decoder on frozen visual features exceeds 80 Jaccard on UMD [Dorati and Banfi, 2026], against 49 for a linear readout of this backbone's tokens) and a training signal that makes the action depend on it, such as counterfactual scenes or supervision of the representation the action reads (Appendix F).

Safety. An external safety signal helps only if the policy reads it. In the flow-matching expert, adding the correction to its output works; redirecting attention does nothing. Because a linear readout of the backbone's own features suffices, safety corrections can be added without retraining the VLA, complementing safety layers that act on the commanded motion, such as operational-space CBFs [Morton and Pavone, 2025] or constrained whole-body tracking [Morton et al., 2026]. System-level stacks add gates between planning and actuation, for example a VLM that checks each planned tool call and an anomaly detector on the controllers' inputs that halts the robot before balance, joint-limit or collision violations [Banfi et al., 2026]. Such gates can block or stop an action; output injection is complementary, because it corrects the action from within the policy.

Implications for training. Each recommendation follows from a measurement and states what retraining should change; we did not run these experiments. (i) *Protect where the scene code forms.* The scene code forms in layers 20–28, and the vision tower changes little (Section 4.1); freezing or regularising those layers, or insulating their gradients, should keep the instruction-direction effect and the words' coupling to objects (Appendix E.4); the open question is the cost in action quality. (ii) *Make the expert depend on the scene.* The expert takes the state and, on familiar data, the episode (Appendix D); counterfactual scenes (obstacle moved or removed) and state dropout should raise its response to the obstacle above $1.3\times$ nf, which the permutation gates of Appendix F measure. (iii) *Add what the backbone lacks.* Identity and parts are absent (Section 4.3); an explicit affordance model, or part and identity labels as auxiliary targets at the layers where the scene code forms, should supply the part-level information that a linear readout of the backbone does not (49 vs. above 80 Jaccard on UMD). (iv) *Deliver signals where the policy reads.* Safety and task information should enter the expert's output, which worked for safety without retraining (Section 4.4), whereas an attention bias on the same tokens did nothing. (v) *Audit with counterfactual interventions.* State removal, same-episode image shifts, the instruction-direction test and the output/attention separation expose each failure above during training.

Limitations. We study one action-pretrained pair in detail, so we cannot separate the effect of egocentric data from that of action-token training in general; the second VLA is architecturally different and shows the same language loss and state dominance, but we did not repeat every analysis on it. Decision-token analyses use the stage-1 next token, not the deployed policy; the action expert is studied on one simulated task with a single trained checkpoint and a single instruction, so ignoring it is not wrong there (the backbone tests do not depend on this), and the safety study rests on 70 violating chunks in eight scenes. Output injection is an intervention with access to the CBF correction, not a deployed safety layer, and it is injected at single flow steps (Figure 5) and makes 20–24 % of the 70 chunks safe; it is tested on one predicate (obstacle clearance). The unseen episodes are teleoperated, so their states are also off the training distribution, and we cannot say whether the expert would use the scene code if trained with more data or with counterfactual scenes. Attention blocking is a lower bound on a region's influence because visual tokens mix before the block, and a layer skip also changes the normalisation of the residual stream. Identity and parts were probed linearly, so information present but not linearly readable is not excluded, and the UMD instructions contain no negations or spatial prepositions. Continuous action heads and prior-preserving training were not evaluated; the instruction-direction and output/attention tests apply to them unchanged. All models use the FAST action tokenizer, so we cannot say whether a semantically aligned tokenizer [Kang et al., 2026] would keep the language pathway or object identity. Causal patching also has response-theoretic accounts [Olivieri and Rodríguez, 2026] that we do not use. The training recommendations are consequences we draw from the diagnostics; we did not test them by retraining.

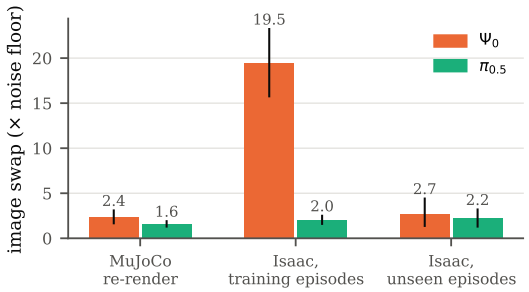

Figure 3: Action change when the image is replaced by one from another episode, in units of the noise floor, by data source.

C Generalisation: a second VLA and unseen episodes

The loss of language and the primacy of the robot state hold across a second architecture and three data sources; the recognition of training episodes does not, and depends most on the training data.

A second VLA. The $\pi_{0.5}$ checkpoint is trained on this task from its DROID release with the VLM frozen, as for Ψ_0. It differs from Ψ_0 in its backbone (PaliGemma, Beyer et al., 2024), in how the expert reads it (keys and values of all layers) and in how the state enters (as text tokens). Its language weights changed less than Ψ_0's (cosine 0.77; vision 0.89), but text NLL still rises from 3.9 to 14.9 nats, above a uniform guess. Its action is dominated by the state in the same way: removing the state tokens' keys and values changes the action by 60–115$\times$ nf across MuJoCo, training and unseen Isaac frames (60 frames per source), removing the image tokens' by 2.7–2.9$\times$ nf, and removing the instruction's by 0.4–0.5$\times$ nf, less than sampling noise. Per layer, removing the state's keys and values changes the action by 0.4–10.6$\times$ nf (peaks at layers 6, 8, 10 and 11) while the image changes it by at most 1.4$\times$ nf and the instruction by at most 0.2$\times$ nf at any layer (24 training frames). Unlike Ψ_0, it does not use the image to recognise training episodes: an image from another episode changes its action by 1.6–2.2$\times$ nf on every source (Figure 3).

Unseen episodes. On 80 training frames, Ψ_0's action error is 0.0031 (mean-action baseline 0.079) and replacing the image changes the action by 19.5$\times$ nf $[15.8, 23.2]$. On 80 unseen frames the error is 0.132, worse than the mean action (0.092), and replacing the image changes neither the action (2.7$\times$ nf $[1.4, 4.4]$) nor the error. On training frames, another episode's image moves the action closer to *that* episode's ground truth than to its own (error 0.20 vs. 0.31, Δ $+0.11$ $[0.05, 0.16]$), an image ten frames away in the same episode changes it by 9.4$\times$ nf, and Ψ_0 is far more accurate than nearest-neighbour retrieval on the state (0.012 vs. 0.046), so the image complements the state. Unseen episodes are teleoperated, so their states are off-support and explain part of the error.

A real-robot checkpoint. A third Ψ_0 checkpoint, trained on real episodes of a G1 picking up a bottle and pouring, shows the same pattern on 60 frames of its own task. Replacing the instruction by an empty one changes the action by 0.6$\times$ nf, below sampling noise, while replacing the image changes it by 4.4$\times$ nf $[3.9, 4.8]$, graying it by 5.5$\times$ nf and removing the state by 7.6$\times$ nf. The state still dominates but far less than in simulation (51$\times$ nf), and the image matters for accuracy: the open-loop error is 0.024 with the true image, 0.088 with another episode's, 0.113 with a gray one and 0.190 for the mean action. Language is thus unused in all three data sources; image reliance grows with the variety of the data. Because training and evaluation frames come from the same episodes, we cannot tell whether the image is used to perceive the scene or to recognise the episode.

D What the action expert uses

D.1 Interventions on the inputs

The expert is a state-conditioned policy with a weak visual correction. We perturb one input group at a time on 80 frames from training episodes and measure the change of the action chunk in units of the noise floor (Figure 4a). Removing the robot-state token changes the action by 51$\times$ nf $[47, 54]$, more than removing every backbone token at once (22$\times$ nf). Removing the image tokens (1.8$\times$ nf)

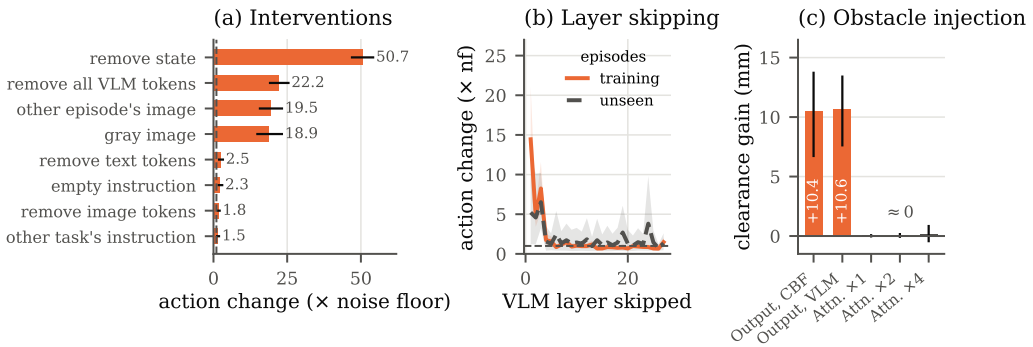

Figure 4: (a) Action change per input intervention, training-episode frames. (b) Action change when one backbone layer is skipped, on training and unseen episodes (dashed: noise floor). (c) Clearance gain from a CBF correction added to the expert's output, the same correction predicted from the backbone's features, or an attention bias of 1, 2 or 4 towards the obstacle.

or the text tokens (2.5× nf), emptying the instruction (2.3× nf) or replacing it with another task's instruction (1.5× nf) all stay close to the noise floor. The exceptions are a gray image (18.9× nf) and an image from another training episode (19.5× nf). Both alter the cue that identifies the episode, which is the effect analysed in Appendix C: on unseen episodes, swapping the image changes the action by only 2.7× nf and does not change its error. The expert therefore reads the state, and on familiar data uses the image to identify the episode and its phase.

D.2 Backbone layers the action depends on

Replacing a single backbone layer by the identity (Figure 4b) changes the action by 5–15× nf for layers 1–3 and by at most 1.8× nf for every layer from 4 to 26, on training and unseen episodes alike, in line with the redundancy of deep layers in language models [Gromov et al., 2025] and with early-exit VLAs [Yue et al., 2024]. An affine map from the layer-4 states to the expert's input reproduces the action (1.3× nf), and the exact Jacobian of the action with respect to the last-layer states puts 90 % of its energy in 1.6 directions. Within those directions, an obstacle's tokens have the influence of an average image token, and per-token attention and per-token knockout effects are uncorrelated ($r = -0.09$ over 40 queries × 77 tokens): where the expert attends is not what it uses [Jain and Wallace, 2019]. Layers 1–3 are also where the action-pretrained backbone builds a massive-activation channel [Sun et al., 2024] whose first principal component carries 79 % of the image-token variance by layer 4 and 85 % at layer 28 (channel 1999 alone 75 % and 44 % in simulated scenes). Zeroing it changes the action by 5–30× nf, as zeroing the next highest-variance channel does (19.5–21.5× nf), and restoring it does not rescue a skipped layer (43 vs. 6× nf), so the dependence is on the global structure of the residual stream and not on this channel alone. In language models such channels act as near-constant biases on which later layers rely [Sun et al., 2024]. The scene content computed in layers 1–3 is not used.

D.3 Output injection versus attention bias

The obstacle tests whether the action uses what the backbone encodes, since decodable information need not be used [Elazar et al., 2021]. Its position is linearly decodable from the backbone at every layer (R^2 0.47–0.68). Yet moving it by 6–12 cm changes the action by 1.3× nf, and the clearance gain near the obstacle is $+0.3$ mm [-0.2, 0.9] over 818 chunks. The information is present, and the action does not use it.

We compare two interventions: adding a correction to the expert's output, which supplies what the action needs, and biasing its attention towards the obstacle tokens, which changes where it reads. We use 70 action chunks that violate the CBF barrier in 8 held-out scenes (mean barrier -38 mm) for both, and inject at the second-to-last flow step (Figure 5). Adding the CBF-QP correction to the expert's output raises clearance by $+10.4$ mm [6.7, 13.7] and makes 20 % of the chunks safe; one correction step cannot remove a deep violation, so this is the ceiling of the intervention. Predicting

the same correction linearly from the backbone's layer-16 obstacle features and the state, with the map fit on training scenes, gives $+10.6$ mm [7.6, 13.4] and 24 % safe chunks, so the backbone holds everything needed to compute the correction. Adding a bias of 1, 2 or 4 to the attention logits from the action tokens to the obstacle tokens (the tokens the obstacle covers by more than a quarter, or any token it overlaps) changes clearance by $+0.02$ mm [-0.03, 0.07], $+0.03$ mm [-0.06, 0.16] and $+0.11$ mm [-0.43, 0.84] and makes no chunk safe. Injections before step 7 are undone by the flow

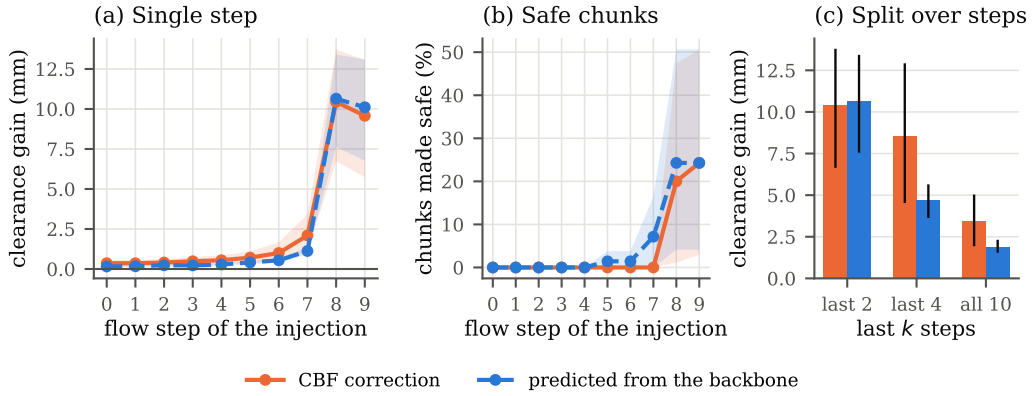

Figure 5: (a) Clearance gain and (b) share of chunks made safe when the correction is injected at one flow step (steps 0–9). (c) Gain when the correction is split over the last k steps.

(below 1 mm), steps 8 and 9 give the full gain, and splitting the correction over the last 2, 4 or 10 steps gives 10.4, 8.5 and 3.4 mm. Steering the backbone itself along the obstacle-displacement direction leads to the same conclusion: it moves the decision token more than a random direction at layers 12–16, but never moves the action beyond $0.25\times$ nf, against $2.1\times$ nf for moving the real obstacle in the same frames (Appendix E.8; $1.3\times$ nf over the full counterfactual set). The expert looks at the obstacle, but nothing it takes from those tokens depends on it, so redirecting attention changes nothing, whereas a correction added to the output does.

Figure 6 follows one unsafe chunk from the image to the action. The expert's attention on the obstacle (1 %, the same as its share of the area, panel a) is not the problem, since the information is in the backbone: the share of the obstacle's displacement that can be decoded from its states is 74 % at layer 4 and still 51 % at layer 28, the only layer the expert reads, but only 0.1 % reaches the hands (b). The layer-28 change caused by moving the obstacle or the target lies in directions the expert is insensitive to: its overlap with the expert's sensitive directions (exact Jacobian through the ten flow steps, 40 frames) is 0.69 and 0.71 of a random direction's (c). Along the example chunk the barrier falls to -28 mm; boosting attention leaves it there, adding the CBF correction to the output brings it to the boundary and predicting the correction from the backbone recovers part of it (d); over all 70 unsafe chunks the output correction adds about 10 mm and the attention boost nothing (e). The information is in the expert's input and the action does not use it; shifting attention does not change that, and a correction added to the output does.

E Additional results

E.1 Attention to objects across layers

The decision token's attention shows when each model consults the image and on what. In the base model about half of the decision token's attention goes to the first token, an attention sink, and little to the image (Figure 8); in the action-pretrained backbone there is no sink, and from layer 4 most of the attention goes to the image, with the instruction receiving a small, late share. Object enrichment mirrors this (Figure 7b): it rises above uniform only in layers 23–28 (1.4–$3.7\times$ on UMD, $6.3\times$ on the simulated targets), peaking at layer 27 on both datasets, and image knockouts agree that the image is still needed there (Figure 7c).

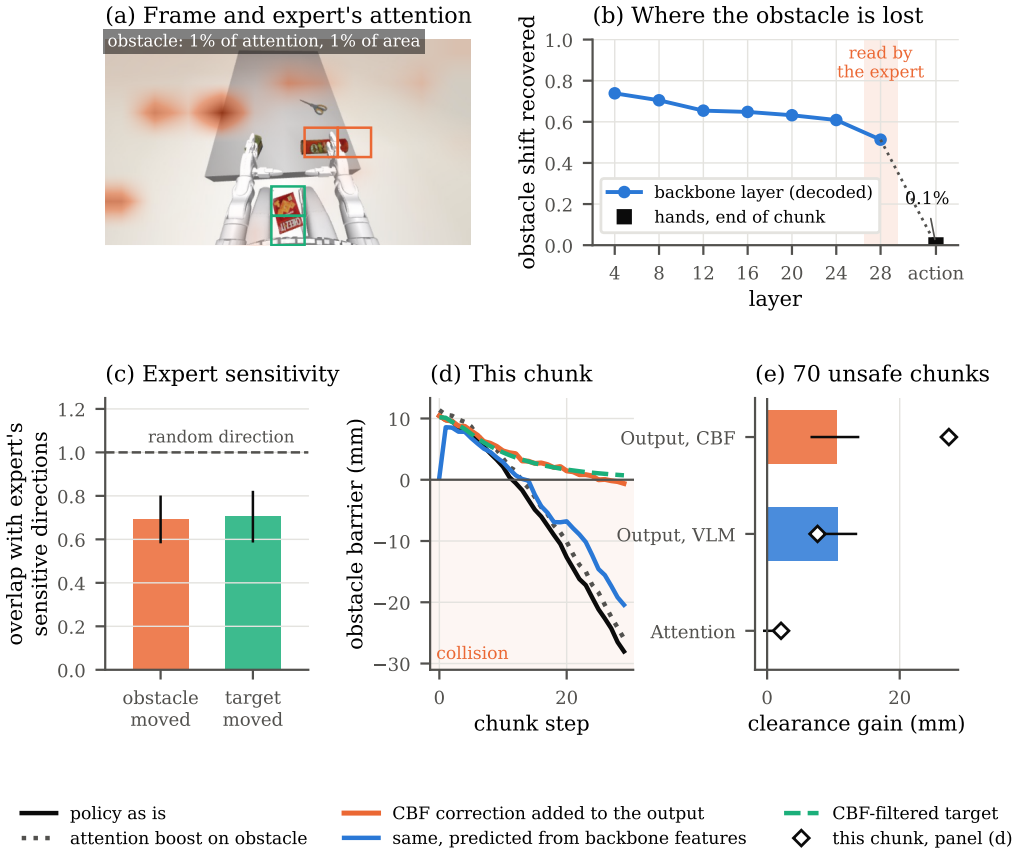

Figure 6: (a) Frame with the obstacle (orange cells), the target (green) and the expert's attention. (b) Share of the obstacle's displacement decoded from each backbone layer, and reaching the hands. (c) Overlap of the layer-28 change with the expert's sensitive directions. (d) Obstacle barrier along one chunk for the policy as is, an attention boost, the CBF correction added to the output, the same correction predicted from the backbone, and the CBF-filtered target. (e) Clearance gain over 70 unsafe chunks (bars: mean and 95 % interval; diamonds: the chunk in d).

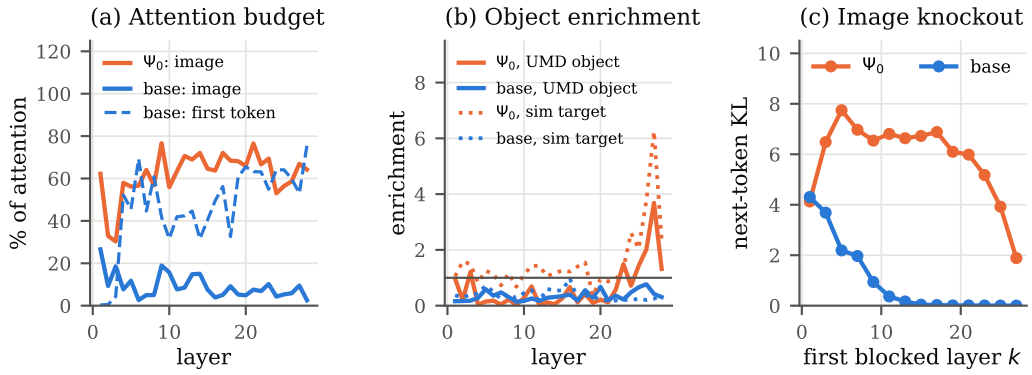

Figure 7: (a) Share of the decision token's attention on the image tokens and on the first token (UMD; base dashed). (b) Attention enrichment, the share of attention over the share of area (1 = uniform, line), on the object in UMD frames and on the target in simulated scenes. (c) Next-token KL when the image is blocked from layer k on.

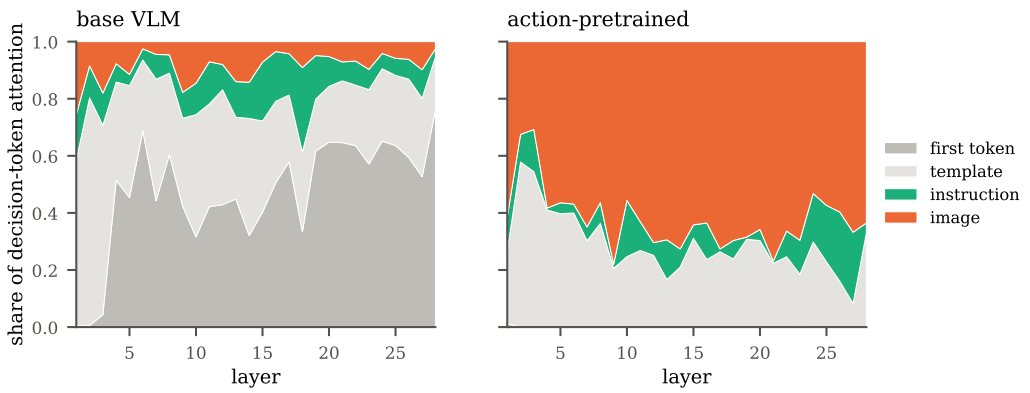

Figure 8: Where the decision token's attention goes, per layer, stacked to 1 (12 UMD frame–instruction pairs): the first token of the prompt (an attention sink), chat-template tokens, instruction tokens and image tokens.

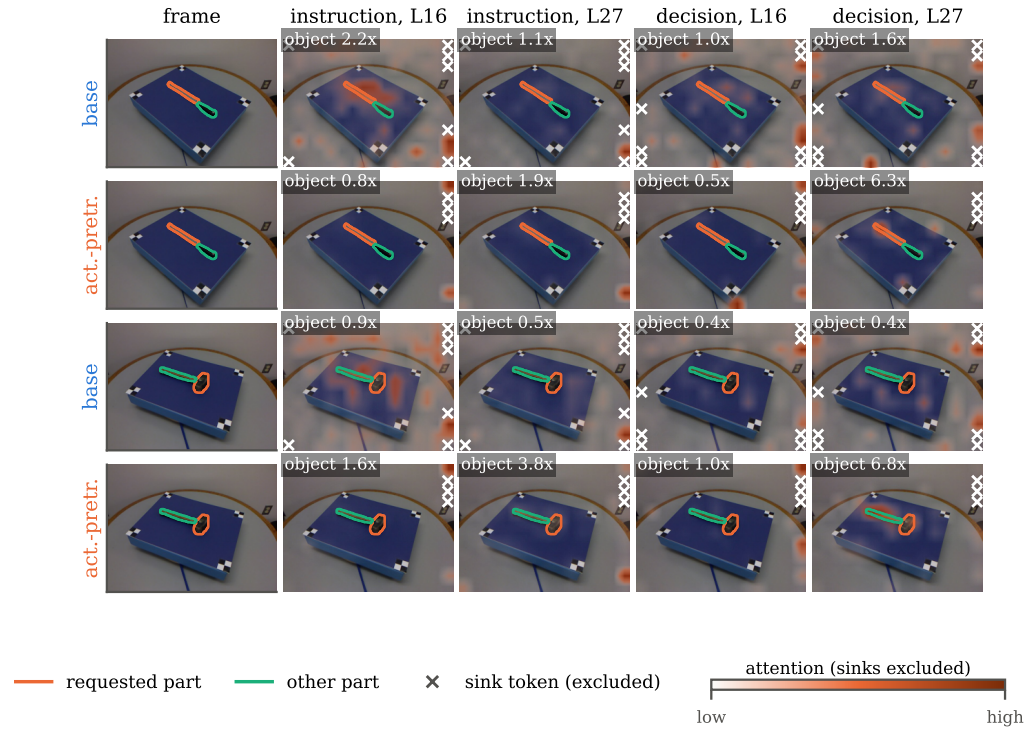

Figure 9: Rows: base and action-pretrained, for two frames. Columns: the frame with the requested part (orange) and the other part (green); attention of the instruction tokens (mean over their positions) and of the decision token at layers 16 and 27, with the enrichment on the object as label. Visual sink tokens are excluded (×).

The maps show who looks at the object (Figure 9). In the base only the instruction tokens do (1.4–1.8× above uniform over 12 frame–instruction pairs) and the decision token hardly does (0.7–1.2×); after action pretraining both converge on the object, with 2.7–3.1× for the instruction tokens and 5.4× for the decision token at layer 27. A late band of heads shows this: in layers 24–28 single heads reach 11× object enrichment, and no head in either model shifts consistently towards the part an instruction asks for (Figure 10). Attention selects objects, and no tested head selects parts.

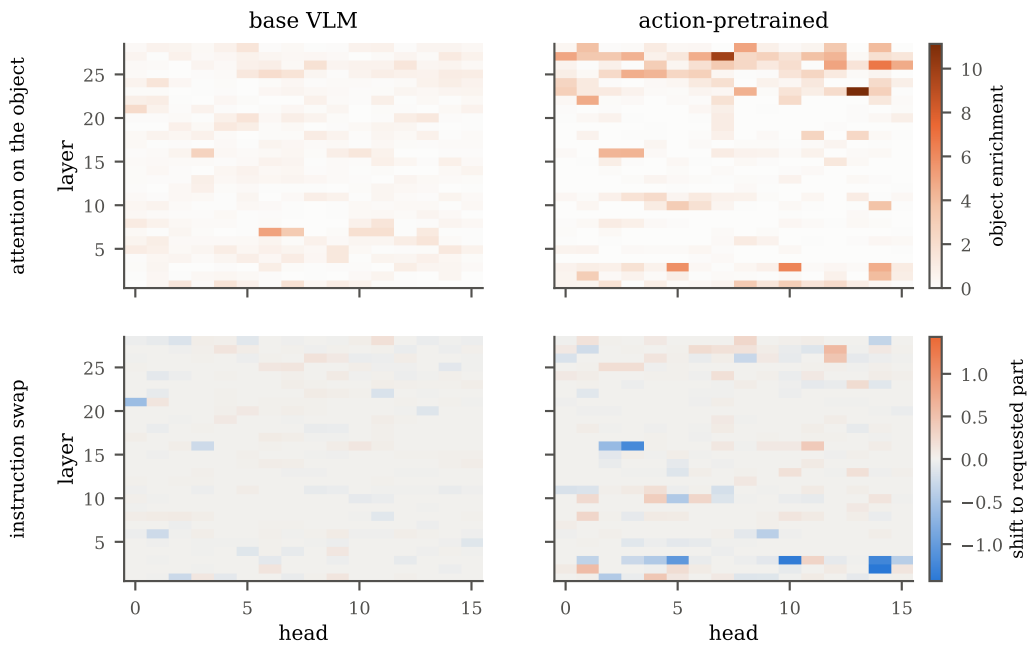

Figure 10: Per head (28 layers $\times$ 16 heads), decision token, 4,080 UMD rows. Top: attention enrichment on the object. Bottom: shift of attention towards the requested part when the instruction asks for the other part of the same object (instruction swap).

The base reads the object through its instruction tokens, as a language model does to answer a question about it; the action-pretrained backbone reads it with the decision token itself, to compute layout for the action token. This is the visible side of the reassignment of layers, and it is easy to monitor: whether a late object band appears, and whether it follows the instruction, can be checked on any VLA during training.

E.2 Part specificity

The decision token selects objects and does not select parts: asking for the blade or the handle of a knife produces almost the same map (Figure 11). On UMD tools, every test of whether it selects the part an instruction asks for is negative. When the instruction changes on the same frame, the decision token's attention does not shift towards the newly requested part (at most $+0.06$ in enrichment over 112 pairs), and only 15 of 448 heads per model show a significant shift, mostly in early layers and towards the other part. Blocking attention to the instructed part changes the next token as much as blocking an equal-size other part (KL 0.041 vs. 0.033, p 0.05), and graying it in the pixels has the same effect as graying the other part (1.141 vs. 1.144, p 0.95), whereas blocking the whole object has five times the effect of an equal background area (0.114 vs. 0.022, p$<$0.001). A part map decoded from the decision token follows the instruction on the same frame (assignment 0.78), but a prior that knows only the object category and the requested affordance does better (0.87), and the decoded map stays below an oracle given only the object mask and the category (R^2 0.36 vs. 0.47). On our scenes the corresponding swap test is fully explained by a role prior (0.98 for both). The one instruction-specific trace is in the affordance readout: graying the instructed part lowers its accuracy more than graying the other part (-0.05, p$<$0.001). A VLA that must act on a part therefore needs part information from outside the backbone.

E.3 Dense probes on the visual tokens

Linear segmentation heads trained on each layer's visual tokens (UMD, 51 test tool instances, threshold calibrated on validation, three probe seeds) reach a Jaccard index of 49.4 in the base and 47.1 in the action-pretrained backbone (Δ -2.3 [-5.2, 0.4]); pixel average precision is lower

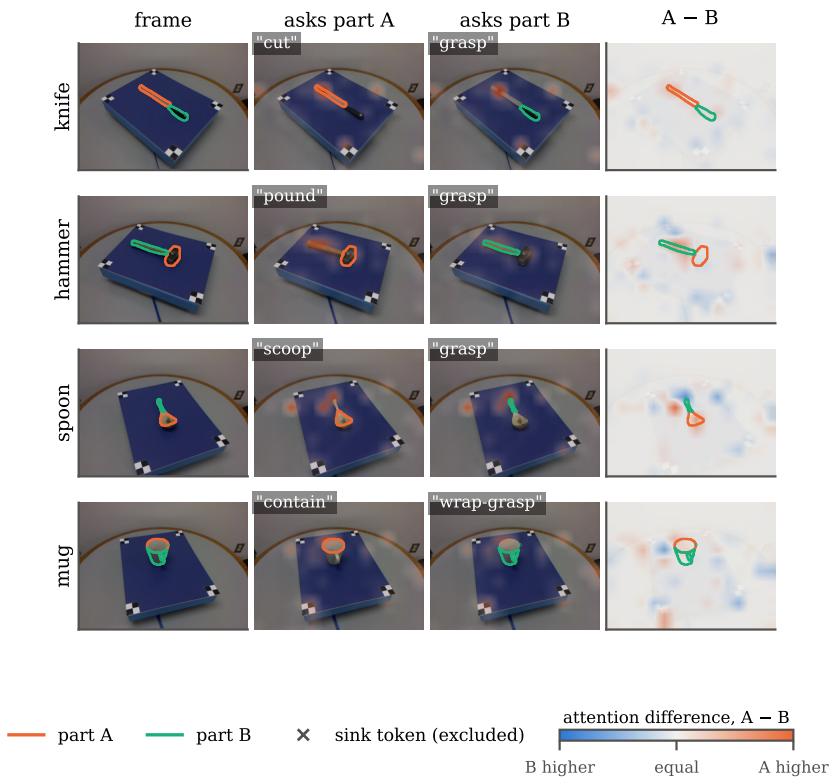

Figure 11: Same frame with the instruction asking for part A or part B (action-pretrained decision token, layer 27) and their difference.

(-0.056 [-0.090, -0.041]) and surface normals are slightly better ($-1.25°$ [-2.58, -0.24]). At 896 px the base gains 11.6 Jaccard and the action-pretrained backbone 2.0, so the gap widens to -10.0 [-14.3, -4.8] Jaccard and -0.108 in average precision (one probe seed; mostly support, 20 vs. 61, and pound, 50 vs. 63): extra pixels help only the base, which fits a backbone whose visual tokens were optimised for coarse, object-level layout. The action-pretrained vision tower's output is better (layer-0 average precision $+0.071$), but its language layers add less detail (layers 0 to 4: $+2.6$ vs. $+6.2$ Jaccard) and lose more with depth: dense part information is less accessible after action pretraining, consistent with a scene code of objects and positions without parts.

E.4 Instruction content and naming

We ask how much each modality adds to the instruction's affordance label (which part of an object to use), as the accuracy of a linear router on the decision token with both inputs minus the accuracy with one of them removed. The image's contribution is both minus language alone, with the image blocked in every layer up to the readout; the language's contribution is both minus image alone, with no instruction in the prompt. The image's contribution is read in two ways: with a router trained on clean states (frozen), which measures what a fixed readout expects, and with one retrained on the blocked states, which measures what the states still contain.

The image adds little (Figure 12a). Language alone, read with a router retrained on the blocked states, reaches 0.78 in the action-pretrained backbone and 0.94 in the base, against 0.84 and 0.97 with the image, so the image adds only 0.06 and 0.03 (a router trained on text only agrees: $+0.07$ [-0.02, 0.17], p 0.14, and $+0.04$, p 0.21; the two gains do not differ, $+0.04$ [-0.06, 0.14], p 0.51). A router trained on clean states (frozen) would suggest the opposite: with the image blocked it falls to 0.44 and 0.18, an apparent image contribution of 0.40 and 0.79. Almost all of this reflects the frozen readout's expectation of image-mixed states and carries no information about the label; the same holds for partial windows, where blocking layers 1–12 costs the frozen router 8–14 points in

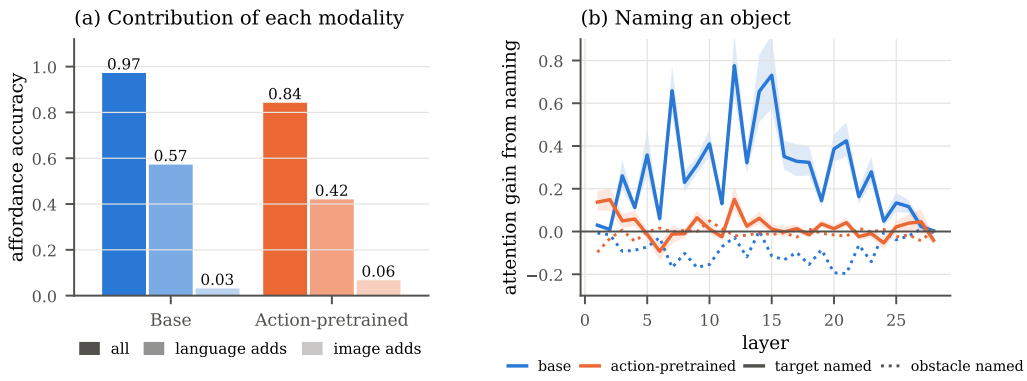

Figure 12: (a) Accuracy of the affordance router with image and text, and the contribution of each modality (routers retrained per condition; 340 instructions, 102 held out per split). (b) Gain in attention on an object when the instruction names it rather than the other object, per layer (blue: base; orange: action-pretrained; solid: target named, dotted: obstacle named; 59 frames, bands: 95 % intervals).

the action-pretrained backbone while a retrained one reaches 0.82–0.86. The image changes how the label is encoded at the text positions; the label is present without it.

Language adds far more, and less after action pretraining (Figure 12a). Removing the instruction costs 0.57 in the base and 0.42 in the action-pretrained backbone (image-only routers reach 0.40 and 0.42), a quarter less, in line with language-alone accuracy falling from 0.93 to 0.77; the intervals of a single split are about ± 0.09, so the difference is descriptive. The base is more accurate in absolute terms under every condition, so the action-pretrained backbone's larger relative gain from the image reflects a weaker language-alone readout and not a better use of the image.

What the instruction no longer does is point at the object it names (Figure 12b). Naming the target raises the attention index of the post-image tokens on it by up to $+0.78$ [0.63, 0.93] at layer 12 in the base, significantly in 26 of 28 layers, against $+0.15$ [0.10, 0.21] and 10 of 28 layers after action pretraining; naming the obstacle does so in neither (at most $+0.05$). In the base, language and image meet at the objects the words name, which is the mechanism prompt-based steering relies on; after action pretraining this coupling is five times weaker, as in the instruction-direction test (Section 4.2).

E.5 Geometry of the backbone

here,mv2main

Standardised to remove the massive channel described below, the representations of the two models drift apart towards the output: linear centred kernel alignment (CKA) between base and action-pretrained states falls from 0.49 at layer 4 to 0.20 for image tokens, and from 0.63 to 0.40 for the decision token (Figure 13a). The action-pretrained image-token states have a raw participation ratio of 1.4 at layer 28 (base 19.0), because their first principal component, dominated by a single channel (index 1999, mean -2816, s.d. 3120), carries 85 % of their variance, a massive activation [Sun et al., 2024]. Such a channel makes raw rank and similarity statistics near-degenerate, so they should be standardised or masked before models are compared. Without the five highest-variance dimensions, the action-pretrained states have a higher rank than the base's at layers 4–16 (36–90 vs. 21–27) and a comparable one later. The action-pretrained decision token spans many more dimensions across frames (robust participation ratio 40.8 vs. 5.1 at layer 28, Figure 13b): it varies with the scene, while the base's barely does. An affine map from the base's image-token states predicts 87 % of the action-pretrained variance at layer 4 but only 45–54 % at layers 16–28 (Figure 13c), in line with a new late-layer code. In the weights, the action-pretrained attention projections have flatter spectra (higher entropy rank in 25 of 28 layers for the query, key and output projections) but larger top singular values (stable rank of the query projection 32 vs. 57).

The patch colours of Figure 14 show what changed. The base model's patches form a fine mosaic with little spatial grouping at every layer, like a recogniser of local appearance. After action pretraining

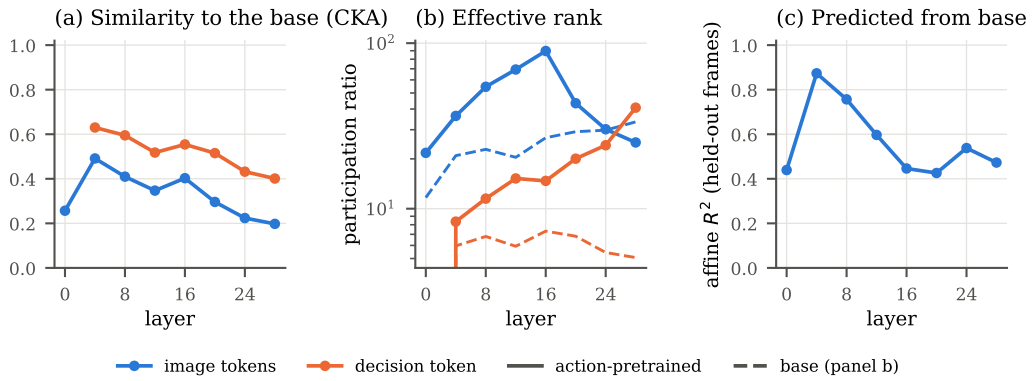

Figure 13: (a) Linear CKA between base and action-pretrained states, for image tokens (blue) and the decision token (orange). (b) Participation ratio without the five highest-variance dimensions (solid: action-pretrained; dashed: base). (c) Share of the action-pretrained image-token variance predicted by an affine map from the base (held-out frames).

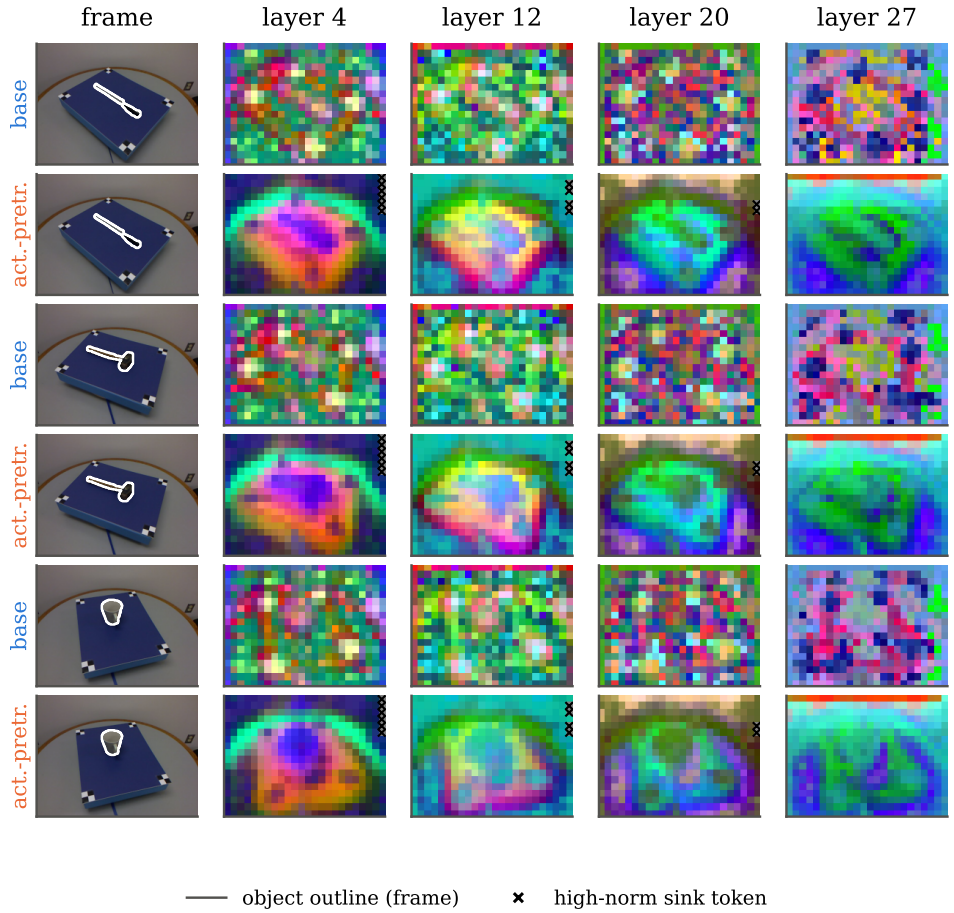

Figure 14: First three principal components of the image-token states as RGB, fitted per model and layer, for three UMD frames (768×576, 18 × 24 tokens); outline: object; ×: high-norm sink token.

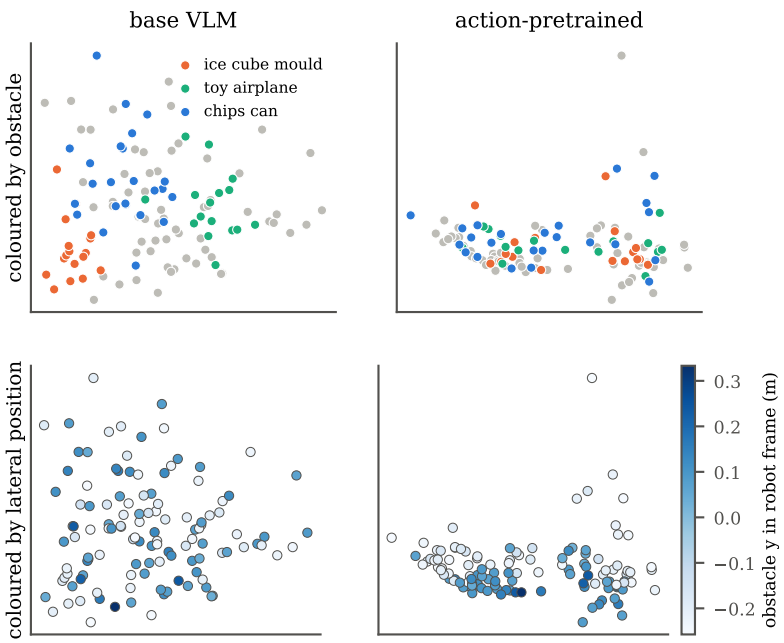

Figure 15: First two principal components of the standardised last-layer obstacle states (120 simulated frames), coloured by obstacle (top; three most frequent, others grey) and by lateral position (bottom).

they organise into coherent regions (table, background, object) from layer 4, and the object keeps its own colour up to layer 27: the backbone carries a spatial layout of the scene.

At the last layer of the simulated scenes (Figure 15), the base clusters frames by semantic content, the identity of the obstacle, whereas after action pretraining the identity clusters disappear and frames are arranged by the obstacle's lateral position. A k-means with one cluster per obstacle recovers the identity in the base (NMI 0.55 [0.50, 0.65], 0.18 for shuffled labels) and not after action pretraining (0.14, at chance); the five nearest neighbours of a frame in other scenes show the same obstacle in 56 % of cases in the base and 17 % after action pretraining (chance 11 %). Lateral position is organised in both models (Spearman correlation with the neighbours' position 0.29 [0.11, 0.43] in the base, 0.42 [0.28, 0.54] after action pretraining).

E.6 Crosscoder features

A crosscoder [Lindsey et al., 2024, Minder et al., 2025] trained on the paired image-token states of both models at one layer (BatchTopK, 2,048 features) learns a dictionary, and the norms of each feature's two decoders tell whether the base uses it (lost by action pretraining), the action-pretrained model uses it (new) or both do. At layer 16 most features are shared (about 1,070 of 1,860), 168 ± 3 are lost and 46 ± 3 are new. The new ones are selective for what the hands need, with $2.6 \pm 0.3\times$ the average activation on obstacle tokens, $1.8\times$ on target tokens and $1.5\times$ on hand tokens, whereas the lost ones are, if anything, anti-selective (0.65, 0.80, 0.75). At layer 28 the picture changes: 392 features are lost and 15 are new, and neither group is selective for obstacles or targets (1.1–1.2×); the new ones are mildly hand-selective (1.5×). Action pretraining thus adds a few spatial features in the middle of the network and mostly removes base-specific features at the end, in line with the loss of identity and language. The counts are stable across seeds (s.d. $\leq$3 at layer 16) but depend on the decoder-norm thresholds (0.1 and 0.9), so they describe this dictionary.

The action-pretrained backbone also builds a coordinate. Moving the obstacle sideways by $\pm$12 cm changes its pooled states at layer 28 along a direction that is shared across scenes (mean $|\cos|$ 0.58 over 85 frames; 0.48 when the target moves by $\pm$6 cm, 119 frames), against 0.12–0.15 in the base and 0.07–0.17 at every other layer of both models (random directions: 0.02; Figure 16c). The last layer thus holds a position axis shared across scenes. The editing experiment moves it without moving the action (Appendix E.8), so the axis is present and the action does not read it.

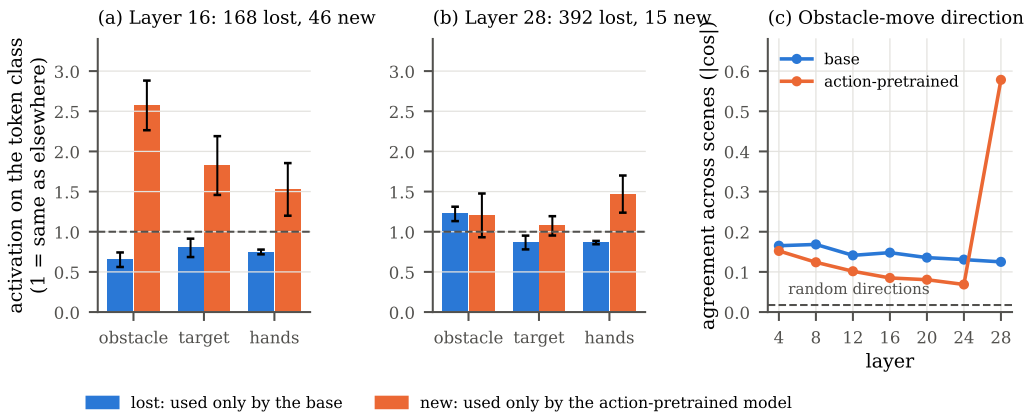

Figure 16: (a, b) Mean activation of lost and new crosscoder features on obstacle, target and hand tokens at layers 16 and 28 (mean and s.d. over five seeds; 1 is no selectivity). (c) Agreement across scenes of the change in the obstacle's pooled states when it moves sideways by ± 12 cm (dashed: random directions).

E.7 Flow expert dynamics

The flow expert computes a direct, low-dimensional map from inputs to action, so extra inference compute does not buy safety. Sixteen noise seeds give nearly identical actions for a frame, while different frames give very different ones (the spread between frames is 40–110$\times$ the spread within a frame). Its residual stream has a participation ratio of 1.1–2.7 at every step and block, matching the 1.6-dimensional motor workspace of the Jacobian. Decisions bind late: switching the robot state at flow step 5 still realises 54 % of its effect and switching the scene 62–67 %, and a velocity perturbation is undone unless it is injected in the last two steps. More computation does not help: two and fifty Euler steps give the same clearance, and repeating backbone layers or expert blocks never improves avoidance ($\leq$2 mm, intervals include 0). The expert's residual encodes the clearance from the first step (R^2 0.86) while its explicit action estimate does not (0.50), another instance of information that is represented but not used. Appending the base model's chain-of-thought tokens changes clearance by $+2.8$–3.0 mm whether the reasoning is the frame's own, the opposite conclusion or another scene's, and length-matched Gaussian tokens change it by -2.4 mm, so extra tokens act as generic context and not as reasoning.

E.8 Activation editing

Steering experiments on 24 frames separate what changes the decision token from what changes the action. Adding the difference between the decision-token states under two instructions changes the base model's next token as much as the real instruction change (KL 0.072 vs. 0.073 at layer 24) and four times more than a random direction (0.017), whereas in the action-pretrained backbone it equals a random direction at every layer and moves the action by 0.34–0.43$\times$ nf, the same as a random direction, against 1.27$\times$ nf for the real instruction change. Adding the obstacle-displacement direction to the obstacle's tokens moves the action-pretrained decision token more than a random direction at layers 12–16 (KL 0.042 vs. 0.018 at layer 16), but moves the action by 0.19$\times$ nf (random 0.16$\times$ nf) against 2.11$\times$ nf for moving the real obstacle. Blocking the decision token's attention to the obstacle at any single layer changes the next token by at most KL 0.009 and the action by 0.13$\times$ nf, less than blocking an equal background area (0.33$\times$ nf). Further directions confirm this (Figure 17). Adding the hand-selection (left minus right), object-identity and obstacle-displacement directions to the decision token at layers 16–27 moves the action-pretrained policy by at most 0.15$\times$ nf more than a random direction of equal norm. At the next token, the identity direction exceeds random at layers 16–24 (KL $+2.8$, $+2.4$ and $+0.8$ nats) and the hand direction does so marginally; in the base model the leverage depends on the layer (identity 3.2$\times$ random at layer 16, 0.2$\times$ at layer 24). Edits in the backbone do not steer this policy, whereas edits at the expert's output do.

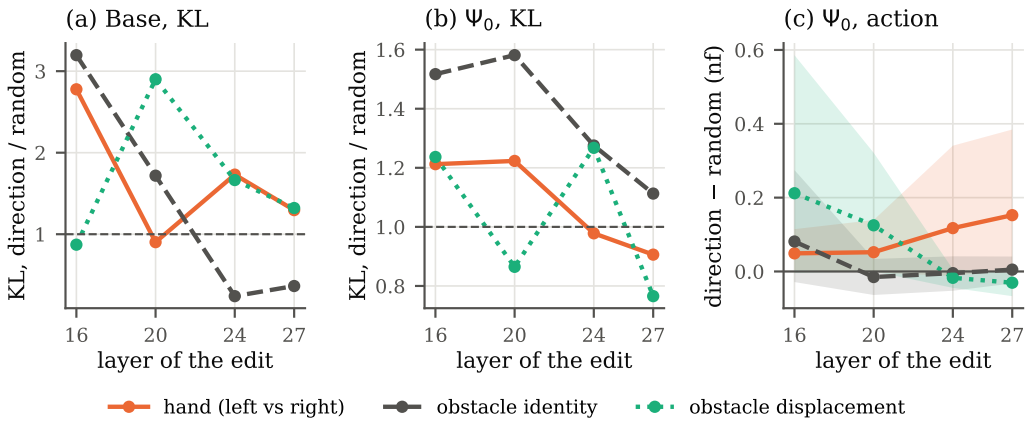

Figure 17: Next-token KL of each direction over that of random directions, per layer of the edit, in (a) the base model and (b) the action-pretrained backbone. (c) Action change of each direction minus random (nf; bands: 95 % intervals).

F From diagnosis to VLA training: structured latent priors

The analyses in this paper are diagnostic, but the purpose of diagnosing a VLA is to change how it is trained. We are building such an extension on top of the same VLA. Instead of letting an action expert read the backbone directly, a flow-matching planner produces a structured latent packet (one slot per future time step and per body part), and a body-side controller turns the packet into joint targets from proprioception and touch alone. The controller never sees the task, so task meaning can reach the robot only through the latent, which is supervised by probes to carry the target, contacts, the held object and the goal. Every attention layer also receives declared relational priors (relative geometry, kinematic structure, contact and task relations) as additive terms on its logits. The design targets the two gaps we identify: the supervised latent replaces the lost language pathway as the channel for task meaning, and the relational priors add structure that the backbone does not deliver to the action.

Our initial results with this extension, which are preliminary and not part of the claims of this paper, reproduce the failure we describe in Section 4.4. A structured head trained on top of the VLA first scored 0/20, because its controller ignored the latent and acted as a state-only policy. Adding state dropout and a gate that requires the action error to grow when the latent is permuted made the head pass the gate, but it remained below direct fine-tuning on one task (7/20 vs. 19/20) and was not distinguishable from it on another (18/20 vs. 15/20). Our analysis suggests why this shortcut is the default: a reader trained on a fixed representation with little data takes the cheapest informative input, which is the robot state, as Ψ_0's own expert does. It also suggests that a single use gate is not enough. The interventions used here (state and image swaps, same-episode image shifts, and the instruction-direction test), can audit during training whether the controller depends on the latent, on the scene and on the task, and catch episode recognition before it is mistaken for perception.

Our results also indicate what the latent should be supervised on. Semantic supervision of the packet raised arm success from 0.65 to 0.89, so supervision matters; the question is its content. The action-pretrained backbone already encodes object positions and spatial relations, and the action ignores them (Section 4.3, Appendix D): these are the quantities worth making the latent carry and the controller depend on, because they are available from the backbone at no cost. Object identity and part-level affordance are absent from the backbone, and the packet also encodes which hand is active only weakly (0.61–0.67 agreement), so they have to come from labels or explicit affordance model.

Finally, our CBF study bears on how relational priors should enter the model. Priors added to attention logits change where a token reads; in the action expert, changing where it reads had no effect, while adding a correction to its output did (Appendix D.3). Relational priors added to attention logits alone may therefore have no effect, and should be checked with the same output/attention separation. The diagnostics, supervision targets and output injection outlined here are the elements of training VLAs with explicit structured priors, the direction our initial results point to.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The four findings in the abstract are measured in Section 4; the scope is in Appendix B.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Appendix B covers one action-pretrained pair, one simulated task, 70 violating chunks and untested recommendations.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: No theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 3 and Appendix A. Models and the UMD dataset are public.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Code, probing scripts and the counterfactual queries are released with the paper. Checkpoints and UMD are public.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 3 and Appendix A.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: 95 % cluster bootstraps and permutation tests (Appendix A). Variance across trained checkpoints is not covered (Appendix B).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix A reports one 6 GB GPU for the simulated analyses and about 25 A40 GPU-hours for UMD. Nothing is trained.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Analysis of public models on public and simulated data; no human subjects or personal data.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Diagnostic study. The simulated safety result is not a safety guarantee (Appendix B).

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper is a diagnostic study of existing public models and poses no particular risk of this kind.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Qwen3-VL, Ψ_0, $\pi_{0.5}$ and UMD are cited and used as released (Appendix A).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: No new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: No human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: No human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: The base model decodes states in Patchscopes (Appendix A).

    Guidelines:

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