The Hitchhiker's Guide to Mechanistic Interpretability of Vision-Language-Action Models

Aryan Goyal [1 2]

Abstract

The mechanistic interpretability toolkit for large language models (LLMs) was built around properties that autoregressive text transformers happen to have: one residual stream, one unembedding matrix, homogeneous layers, a single forward pass per output, a verifiable target token. These properties are specific to that architecture, not general features of neural computation. Vision-Language-Action (VLA) models violate between one and six of them depending on their design, and tools that rely on them still might run without error and return results that look valid but no longer measure what they did on an LLM. We name seven such properties as cautions **A1–A7**, classify VLAs into seven architecture types, and identify which caution each type breaks. The result is a map of where the LLM interpretability toolkit applies unchanged, where it needs a small adaptation, and where it must be rebuilt.

1. Introduction

Mechanistic interpretability seeks to reverse-engineer neural networks at the level of individual components: which attention heads implement which computations, how information flows through the residual stream, and which minimal circuit is causally responsible for a given behaviour (Elhage et al., 2021; Olsson et al., 2022). Progress on language models has been substantial, with circuits identified for indirect object identification, factual recall, and induction (Wang et al., 2023; Meng et al., 2022), and feature-level analyses scaling via sparse autoencoders (Cunningham et al., 2023; Bricken et al., 2023).

Vision-Language-Action (VLA) models now lead generalist robot control: RT-2 (Brohan et al., 2023), OpenVLA (Kim et al., 2024), π_0 (Black et al., 2024), and Octo (Octo Project Team, 2023) couple a language-model backbone with a robot action head and produce motor commands in a single differentiable system. The safety stakes are high: a robot acting on a misunderstood scene causes physical harm in ways a language model producing a wrong token does not.

VLA mechanistic interpretability is, however, almost entirely unexplored, and we argue that the tooling and techniques don't translate directly and infact it's a harder problem. The architectural diversity of VLAs means the LLM interpretability toolkit, applied without modification, ranges from fully applicable to silently and systematically incorrect depending on which VLA is studied. Applying Direct Logit Attribution (DLA) to a cross-attention fusion VLA gets a heatmap that looks plausible and is wrong — the cross-attention contributions are missing entirely, attributed to nothing. Applying SVD to a continuous-action VLA's residual stream identifies heads with zero causal control over the output while missing the heads that drive it. Neither tool raises an error.

This is a positioning paper aimed at the practitioner about to run a mech interp experiment on a VLAs. We check how much of the mech interp tooling and techniques translate to VLAs. We name the LLM-side properties the toolkit silently depends on, identify which VLA architectures preserve or remove each property, and describe the failure modes that follow.

2. Related Work

Häon et al. (2025) are the closest prior work in spirit. They apply LLM-style mech interp to OpenVLA and π_0 by projecting FFN activations onto the token embedding space, recover sparse "semantic" directions like "slow", and use these as inference-time steering vectors, reporting that under 25% of FFN neurons are "rewired" for action prediction.

Grant et al. (2026) did the first cross-VLA mech-interp study: six models, about 394k rollouts, using SAEs, linear probes, and activation injection. Their main findings are that vision contributes more than language, that language matters more on some tasks than others, and that the network splits into separate motor and goal pathways. They also note that activations have to be read one token at a time — averaging

[1]Independent Researcher [2]Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Aryan Goyal <aryangoy@andrew.cmu.edu>.

across the sequence destroys the signal. That last point is exactly what our **A3/A6** cautions predict, and we treat it as outside confirmation.

3. Background: The LLM Toolkit

A transformer with L layers and H heads per layer maintains a residual stream $h^{(l)} \in \mathbb{R}^{T\times d}$, and the central identity (Elhage et al., 2021) is

$$h_t^{(L)} = h_t^{(0)} + \sum_{l=1}^{L}\sum_{h=1}^{H} head_{l,h}(h^{(l-1)})_t + \sum_{l=1}^{L} \mathrm{MLP}_l(h^{(l-1)})_t. \tag{1}$$

DLA projects this decomposition through a shared unembedding $W_U \in \mathbb{R}^{d\times|V|}$:

$$\ell_{v^+} - \ell_{v^-} = \sum_{l,h} head_{l,h}\cdot(W_U[:,v^+] - W_U[:,v^-]) + \dots \tag{2}$$

Equation (1) is the substrate; (2) the readout. Activation patching, attribution patching (Nanda, 2023), path patching (Goldowsky-Dill et al., 2023), sparse autoencoders, and linear probes all use one or both. Their guarantees inherit the guarantees of (1) and (2): a single residual sum, a single linear readout, and a homogeneous stack of layers.

Section 4 names seven such properties the toolkit silently assumes; Section 5 defines seven VLA architecture classes and maps which properties each preserves, partially satisfies, or breaks (Table 1).

4. Seven Properties That Don't Carry Over from LLMs to VLAs

A1. Single residual stream. Every attention head and MLP writes into one residual stream additively, so the stream at any layer decomposes into a linear sum of their individual contributions ((1)). DLA, attribution patching and path patching, all exploit this linear decomposition: they attribute behavior by reading off or intervening on individual terms in the sum. Two architectural mechanisms write into the stream from outside this enumeration, so the decomposition is incomplete and any tool built on it silently misses part of the computation. Cross-attention fusion (RoboFlamingo; Type V, §5) injects visual information through extra sublayers that read directly from the vision encoder; these write into the residual stream but are not among the heads and MLPs of (1), so standard implementations drop their contribution. Separate-weight action experts (π_0; Type IV, §5) process action tokens with their own weights while reading the backbone's keys and values at every shared layer, splitting the action computation across two parameter sets so that no single stream contains it.

A2. Shared unembedding for the output of interest. A single W_U projects the final hidden state into output space, which DLA, SAE feature interpretation, and logit-lens analysis all assume. Discrete-token VLAs (RT-2, OpenVLA; Type I, §5) preserve W_U by discretising each action dimension into 256 bins placed in the text vocabulary; continuous action heads (π_0, GR00T N1, Octo; Types II–IV, §5) replace it with a regression MLP, a denoising network, or an ODE-integrated flow field — none reducible to a matrix multiply.

A3. Modules computing the same kind of thing throughout. Every layer of a standard transformer does the same operation on a single stream, so a tool that works at layer ℓ works elsewhere. A VLA is three coupled networks — vision encoder, language/planning backbone, action head — each representing information in a different geometry, and the action head is no longer a transformer layer at all (MLP, denoising U-Net, or flow-matching expert). Logit-lens curves and per-layer SAEs computed uniformly across the full VLA conflate three kinds of computation. Tools must be applied to one module at a time.

A4. Verifiable ground truth at the decision level. Correctness for an LLM is a single target token; ablations flip the top token or not. Robotic tasks admit multiple valid trajectories, and discrete-token VLAs (Type I, §5) predict action dimensions sequentially, so circuits for joint k attend to already-generated joints $1,\dots,k-1$ as conditioning. DLA on joint k mixes the vision-to-action circuit with the inter-dimension coordination circuit. Mitigation: run DLA on action token 1 only (no prior action context), and evaluate ablations in action space (KL, rollout success), not by token rank.

A5. Architectural regularity. A homogeneous stack means a uniform tool can walk it end-to-end. MoE and heterogeneous-expert architectures route different token types to different FFN experts; flow-matching designs give action tokens their own expert weights inside shared layers. The same "head 17 of layer 12" index corresponds to different sub-computations depending on the routing decision, and per-layer attribution averages over sub-computations that cannot meaningfully be combined.

A6. Low superposition in the relevant subspace. Features of interest are approximately linearly separable; SAEs and linear probes recover them. Action subspaces encode continuous physical quantities — Cartesian positions, joint angles, gripper apertures — whose natural geometry is a low-dimensional continuous manifold, not a discrete categorical code. Discretising that manifold into 256 bins packs a smooth structure into a 256-token vocabulary, correlating embedding rows and densely superposing the residual-space

representation.

A7. Single-pass determinism. One forward pass produces one next token, so causal attribution is single-pass. Diffusion VLAs decide over T denoising passes; flow-matching VLAs decide over an ODE integration; action-chunking VLAs emit K future motor commands per forward pass. A one-pass ablation measures a compounded multi-step effect, not a single-step causal contribution. The remedy is to run the analysis at each denoising step or chunk position and report a per-step circuit; the LLM toolkit does not do this out of the box.

What compounding error means for interpretation. LLM next-token prediction and VLA behavioural cloning are the same supervised problem. Both minimise a per-step loss

$$\hat{\pi} \approx \arg\min_{\pi \in \Pi} \sum_{i,t} \mathrm{loss}\big(s_t^{(i)}, a_t^{(i)}, \pi\big)$$

over expert state-action pairs, and both are autoregressive: the output at t becomes the input at $t{+}1$. A per-step error moves the model into states it was not trained on, and these errors accumulate over the rollout.

Ross and Bagnell (2010) give a bound on this accumulation. A behavioural-cloning policy with per-step error ε under the expert distribution has $J(\pi) - J(\pi^*) \leq \varepsilon H^2$, and the H^2 factor is tight for behavioural cloning. This bound is the same order for LLMs and VLAs; the two settings differ in what an error costs. A VLA executes $H{=}100$–200 physical steps and has no expert data to recover from once it leaves the expert distribution, whereas an LLM's errors remain in token space and carry no physical consequence. Action chunking lowers the accumulation by committing to actions only at chunk boundaries, which leaves roughly K times fewer points at which error can compound (Zhao et al., 2023).

The H^2 rate applies to discrete actions. A continuous-action VLA uses a squared loss, so a per-step error is a magnitude that propagates through the system dynamics rather than a single discrete mistake, and its growth is governed by the stability of the closed-loop dynamics. Simchowitz et al. (2025) show that this dependence is fundamental: the demonstrations do not reveal whether the dynamics are stable, and without an algorithmic mechanism that enforces stability the imitation error can compound exponentially in the horizon rather than polynomially. The same per-step error is therefore mild on a discrete-token VLA and can be far more damaging on a continuous one.

This accumulation affects interpretation only when an interpretation is used to act on the policy. A steering vector applied at every step, or a head ablated throughout a rollout, changes the policy by some $\varepsilon_{\text{interp}}$ per step, and this change compounds through the same dynamics: in the discrete case it is bounded by $\mathcal{O}(\varepsilon_{\text{interp}} H^2)$, which a $K{=}8$ chunk reduces by about an order of magnitude, and in the continuous case it grows at the rate the dynamics set. An attribution that is computed incorrectly but never used to intervene does not compound, because there is no rollout for it to act through.

Two consequences follow for evaluation. A successful rollout does not establish that a circuit is correct, since other heads can compensate, and a failed rollout does not establish that it is wrong, since a roughly correct intervention can be amplified by the dynamics. Rollout evaluation therefore measures behaviour, not the correctness of an attribution.

Practitioner pitfalls (instrumentation analogues). Even on a VLA whose architecture passes A1–A7, two pitfalls surface at instrumentation time. **P1: Constrained-decoding subvocabulary.** Inference-time sampling is restricted to a small action token range (e.g. IDs 31744–31999 on OpenVLA). Logit-space tools that softmax over the full $|V| = 32{,}064$ vocabulary normalise the action-token mass by ~ 31.8k irrelevant text-token logits and report $P(\text{correct action}) \approx 0$ at every layer (a false null). Fix: restrict the softmax denominator to the action subspace. **P2: Embedding-level visual injection.** The Prismatic projector injects 256 visual patch features at the embedding level, before tokenisation. Tools that read `input_ids.shape[1]` measure 25 tokens (text only) and report 0% visual attention. Fix: read the true sequence length from `hidden_states[0].shape[1]` $= 281$.

5. VLA Architecture Taxonomy

A VLA maps a visual observation o_t and a language instruction ℓ to an action a_t. The architecture divides into three components: vision encoder, language/action backbone, and action head. The interpretability surface is determined almost entirely by the action head type and the modality fusion strategy. We identify seven non mutually exclusive classes, illustrated for four representative cases in Figure 1.

Type I: Discrete-token VLAs (RT-2, OpenVLA). Pre-trained LLM backbone, visual patch tokens prepended to language tokens, physical actions discretised into N bins per dimension as vocabulary tokens sharing W_U. A1 and A2 hold cleanly. A6 fails: the ordered action vocabulary correlates adjacent unembedding rows, densifying the action-block residual and frustrating SAE/SVD decompositions.

Type II: Continuous-regression VLAs (GR-1). Backbone identical to Type I; the action head is an MLP mapping the final hidden state to a continuous action vector. A2 fails (DLA needs a probe substitute). A6 fails downstream: continuous actions keep SAEs and SVD unreliable. Causal

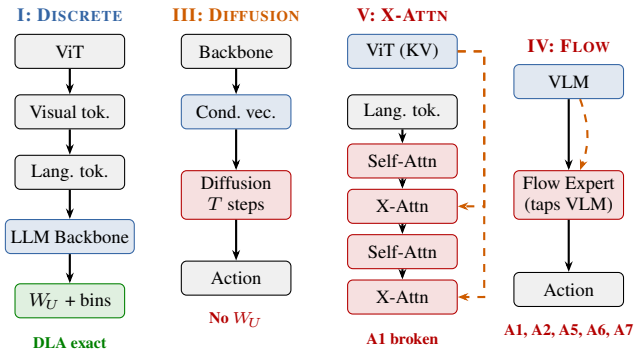

Figure 1. **Four representative VLA classes vs. A1 (single residual stream) and A2 (shared W_U).** Type I prepends visual tokens, keeping a single stream and a shared unembedding (DLA exact). Type III runs a separate diffusion head over T passes (no W_U, A7 violated). Type V injects vision via cross-attention sublayers (dashed orange) read at every other layer (A1 violated). Type IV routes action tokens through a separate-weight flow-matching expert that attends to backbone keys/values within shared layers (A1, A2, A5, A6, A7 violated).

tools (path patching, activation steering) transfer.

Type III: Diffusion-based VLAs (Octo, Diffusion Policy). The backbone produces a conditioning vector; a separate diffusion network iteratively denoises an action from Gaussian noise. A1 fails (diffusion network sits outside the backbone residual stream); A2 fails (no W_U); A7 fails (T denoising passes, none of which is the decision).

Type IV: Flow-matching VLAs (π_0, $\pi_{0.5}$). A pre-trained VLM backbone and a smaller action expert with separate weights process one joint sequence: action tokens attend to backbone keys and values within shared transformer layers under blockwise masking, and the expert generates actions via flow-matching ODE integration. A1 fails (the expert maintains its own residual stream with separate weights, so (1) written over the backbone omits the action computation), A2 fails (no W_U on the action path), A5 fails (per-token-type expert weights mean the same layer/head index names different sub-computations), A6 fails downstream, as in Types II and III, since the action expert represents a continuous action manifold rather than a categorical code, A7 fails (ODE distributes the decision over many steps).

Type V: Cross-attention fusion VLAs (RoboFlamingo). Visual features are not prepended; cross-attention sublayers query the vision encoder at every other layer. A1 fails structurally:

$$h_t^{(L)} = h_t^{(0)} + \sum_{l,h} head_{l,h} + \sum_{l} \underbrace{\mathrm{XA}_l(h^{(l)}, V_{\text{vis}})}_{\text{invisible to standard DLA}} + \dots \tag{3}$$

[1]If cross-attention carries fraction α of the action signal, every DLA score is deflated by $(1-\alpha)$. Standard tools without correction interpret a self-consistent shadow with the visual pathway removed.

Type VI: Heterogeneous-expert VLAs (π_0 action expert, SpatialVLA). Specialised sub-networks handle different modalities; in MoE variants, a router sends token types to different FFN experts. A5 fails (head index is no longer a single thing); A6 fails downstream. π_0 falls under both Type IV and Type VI — flow-matching and heterogeneous-expert decomposition are orthogonal.

Type VII: Temporal-chunking VLAs (ACT). The model predicts K future actions simultaneously; in ACT (Zhao et al., 2023) this is a conditional VAE with cross-attention at every decoder layer. A1 fails (chunked prediction violates single-stream accumulation), A4 fails (no single target token), A7 fails (decision distributed across the chunk).

6. How the Toolkit Breaks: Five Failure Modes

F1: Silent DLA underestimation (A1). If out-of-stream contributions carry fraction α of the action logit, every DLA score is deflated by $(1-\alpha)$. Within-stream rankings *survive* (heads deflated by the same factor preserve ratios), but absolute claims and ablation predictions do not: "head h accounts for 30% of the output" silently means "30% of $(1-\alpha)$", and ablations overstate the actual drop because the model routes around them through the unmeasured pathway.

F2: DLA against a causally disconnected readout (A2, Types II–IV). When no W_U exists for the action, substituting the text W_U runs without error but returns scores measuring contributions to a distribution the model does not use for control, ablating the top heads identified this way leaves rollout performance essentially unchanged. The valid substitute is a supervised probe, $\mathrm{ProbeDLA}_{l,h} = head_{l,h} \cdot w_{\text{probe}}$; it costs the elegance of training-free DLA but its numbers correspond to action-space behaviour.

F3: Variance-aligned direction-finding (A6). SVD returns the directions of greatest activation variance and offers them as the axes of interest. On robot data that objective is misaligned with the question being asked, because the action components differ in scale: end-effector translation varies by roughly $\sim 100\times$ the range of gripper aperture, so the top singular directions align with translation while the gripper-control axis falls into tail components carrying $<1\%$ of the variance. A head ranked first by SVD can be nearly orthogonal to the variable it appears to track and have

[1]The elided terms are the standard MLP and embedding contributions of (1).

	A1	A2	A3	A4	A5	A6	A7
Type I	✓	✓	∼	✓	✓	×	✓
Type II	✓	×	✓	✓	✓	×	✓
Type III	×	×	✓	✓	∼	×	×
Type IV	×	×	✓	∼	×	×	×
Type V	×	×	∼	✓	×	×	✓
Type VI	✓	✓	✓	✓	×	×	✓
Type VII	×	×	✓	×	×	×	×

Table 1. LLM-side conditions for each VLA class. ✓ pass, × fail, ∼ partial. A1: single residual stream. A2: shared W_U. A3: homogeneity. A4: ground truth. A5: regularity. A6: low superposition. A7: single-pass.

near-zero causal effect on it. The fix is to stop ranking by variance: supervised linear regression selects directions by how well they *predict* the action, not by how much variance the residual happens to carry along them.

F4: Token-match evaluation hides action-space errors (A4). Standard mech interp judges a circuit by whether ablation flips the predicted token. On a discretised action vocabulary this is the wrong granularity: bin $31 \rightarrow 32$ flips the token but is behaviourally identical (both encode "slow right"), so an action-space-complete circuit can fail the ablation test; conversely, a circuit that preserves the token can still produce 5-bin shifts that compound into task failure.

F5: Heterogeneous-sequence ablation bias (A3, A6). The VLA token sequence mixes $\sim$256 visual, $\sim$25 language, and 1 action positions with very different activation statistics. Sequence-mean ablation substitutes a single value across all of them, biasing every head's importance by an OOD offset. Noise-based ablation is in-distribution at high-variance positions and wildly OOD at low-variance ones, so head rankings can flip with the noise scale chosen. Position-preserving mean ablation — $\mathbb{E}_{\text{dataset}}[x_{l,h,p}]$ at each position p separately — substitutes only values the model has seen at p, removing both biases.

7. Discussion

This paper does two things. First, it sets out cautions for transferring LLM mech-interp tooling to VLAs. The conveniences of that toolkit hold only for a single architectural choice, discrete autoregressive tokens over one residual stream, and most VLA architectures quietly break it. Second, it argues that VLA mech interp needs tooling built for the setting rather than adapted from LLMs, and it supplies the context that such tooling will need: which constraints break, where, and how to detect the breakage. We frame the paper this way to point the community at the gap with the pitfalls already mapped.

The architectures that are hardest to analyze are the ones being deployed. Toolkit applicability is highest for the discrete-token design and degrades as the action head becomes continuous and multi-pass: Type I breaks one caution, Type III four, Type IV five. The field is moving toward that hard end, since flow-matching and diffusion action heads now lead generalist control.

References

Black, K., Brown, N., Driess, D., et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Bricken, T., Templeton, A., Batson, J., et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Brohan, A. et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Elhage, N., Nanda, N., Olsson, C., et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

Goldowsky-Dill, N., MacLeod, C., Sato, L., and Arora, A. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

Grant, B., Zhao, X., and Wang, P. Not all features are created equal: A mechanistic study of vision-language-action models. In *ICLR Workshop on Multimodal Intelligence*, 2026.

Häon, B., Stocking, K., Chuang, I., and Tomlin, C. Mechanistic interpretability for steering vision-language-action models. *arXiv preprint arXiv:2509.00328*, 2025.

Kim, M. J., Pertsch, K., Karamcheti, S., et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Nanda, N. Attribution patching: Activation patching at industrial scale. Blog post, 2023.

Octo Project Team. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2023.

Olsson, C., Elhage, N., Nanda, N., et al. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

Ross, S. and Bagnell, J. A. Efficient reductions for imitation learning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Simchowitz, M., Pfrommer, D., and Jadbabaie, A. The pitfalls of imitation learning when actions are continuous, 2025. URL https://arxiv.org/abs/2503.09722.

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

Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.