Mechanistic Interpretability for Steering Vision-Language-Action Models in Simulation

Aryan Panda
UC Berkeley
Berkeley, CA
aryanp123@berkeley.edu

Shoumik Roychowdhury
UC Berkeley
Berkeley, CA
sroychow@berkeley.edu

Anjo Pagdanganan
UC Berkeley
Berkeley, CA
anjopag31@berkeley.edu

Sufjan Fana
UC Berkeley
Berkeley, CA
sufjan@berkeley.edu

Abstract

Recent work by [6] demonstrates that activation-level interventions can steer the behaviors of vision–language–action (VLA) policies without retraining, revealing promising avenues for interpretable control in robotics. Their method identifies steerable directions by projecting feed-forward network (FFN) activations into token space, but this approach may overlook the richer semantic structure present in a model's internal residual streams. In this work, we reproduce the steering framework of [6] and investigate whether fine-tuned Sparse Autoencoders (SAEs) provide a more disentangled and expressive latent basis for activation editing in VLA models. A practical constraint of existing tooling is that publicly available SAE probes exist only for π_0-based architectures, but not for OpenVLA. As a result, our study naturally bifurcates into two complementary analyses: (1) we apply the zero-shot, no-training steering method of [6] to OpenVLA, and (2) we evaluate SAE-driven interventions within the $\pi_{0.5}$ architecture, where layer-wise activations and pretrained dictionaries are accessible. Although this creates an asymmetry between the models we test, it also offers insight into how different interpretability tools interface with distinct VLA architectures in practice.Using $\pi_{0.5}$, we fine-tune SAEs on Layer 13 of the Knowledge Expert—an identified semantic bottleneck where abstract intent is consolidated prior to motor decoding—to extract monosemantic, orthogonal features suitable for precise steering. We present our system design, simulation-driven data collection pipeline, and preliminary validation, and outline ongoing comparisons between token-space and SAE-derived steering directions. Our findings aim to clarify the conditions under which representation-learning probes can enhance the controllability and interpretability of modern VLA policies.

1 Introduction

The recent emergence of Vision-Language-Action (VLA) models has marked a significant shift in robot learning, enabling agents to integrate semantic reasoning with low-level control. Architectures such as $\pi_{0.5}$ and Magma demonstrate that large language models (LLMs) can serve as effective backbones for embodied policies, allowing robots to generalize to open-world environments by leveraging the vast knowledge embedded in pre-trained text corpora. However, as these models grow in complexity, their internal decision-making processes remain largely opaque, creating significant challenges for safety, interpretability, and precise control. To address this opacity, recent works have adapted mechanistic interpretability techniques from the language domain to robotics.

For instance, [1] demonstrated that Sparse Autoencoders (SAEs) can identify steerable features—such as ”open gripper”—within the residual stream of a Magma policy. Similarly, Haon et al. proposed extracting Feed-Forward Network (FFN) vectors and clustering them in token space to identify neurons to steer behaviors like transport velocity.

While these methods provide a proof of concept for latent intervention, they face distinct limitations. [1] observed that using off-the-shelf SAEs (trained on the base LLM) resulted in ”imperfect disentanglement” and ”inadvertent activations,” where steering one feature unintentionally triggered others. Conversely, clustering methods that rely on projecting activations to the token space may fail to capture the rich, high-dimensional semantic nuance present in the model’s internal residual streams.

In this work, we investigate whether fine-tuning SAE probes specifically on the internal activations of a VLA policy offers superior feature separability compared to token-space clustering. We hypothesize that while VLA backbones retain the semantic structure of their base LLMs, the domain shift introduced by robotic fine-tuning necessitates a specialized probe to accurately disentangle intent from execution.

We test this hypothesis using $\pi_{0.5}$, a VLA architecture that explicitly decouples high-level reasoning (the ”Knowledge Expert”) from motor generation (the ”Action Expert”). By fine-tuning an SAE on Layer 13 of the Knowledge Expert—a distinct ”semantic bottleneck” where we believe abstract intent is finalized before action decoding—we aim to extract cleaner, more orthogonal steering features than previously possible.

Our contributions are as follows:

1. **Latent-Space SAE Finetuning**: We propose a method to adapt pre-trained LLM SAEs to the specific distribution of VLA policies, targeting the residual stream of instruction tokens.
2. **Architectural Analysis of $\pi_{0.5}$:** We identify Layer 13 of the $\pi_{0.5}$ Knowledge Expert as a critical locus for semantic planning, validating its utility for high-level behavioral steering.
3. **Comparative Evaluation:** We provide a comparative analysis against token-space clustering baselines on the LIBERO benchmark, evaluating the disentanglement and separability of the learned features.

2 Literature Review

To contextualize our investigation into fine-tuned SAE probes for VLA policies, we review the architectural foundations of modern VLA models, specifically the modular design of $\pi_{0.5}$ ([3]) and OpenVLA ([7]). We then discuss recent advances in mechanistic interpretability, focusing on Sparse Autoencoders (SAEs) and activation steering techniques derived from contrastive pairs.

Vision Language Action Models. Recent advances in *vision–language–action* (VLA) models have demonstrated a new paradigm for generalist robotic agents that integrate visual perception, natural language understanding, and action generation. Early pioneering work on the RT-2 model [9] showed that a massive VLA policy (55B parameters) could transfer web-scale vision and language knowledge to control robot tasks. This foundation enabled zero-shot generalization to novel objects and instructions. However, RT-2 remained a closed model and suffered from high inference latency due to its autoregressive action decoding. To address openness and efficiency, *OpenVLA*—a 7B-parameter open-source VLA model trained on 970k robot demonstrations—was introduced [7]. It builds on a LLaMA-2 backbone and achieves strong performance on multi-task manipulation while supporting parameter-efficient adaptation. While early VLA models often adopted monolithic structures fusing visual and linguistic modalities into a single transformer backbone, recent architectures have embraced modularity to improve generalization and training stability, such as $\pi_{0.5}$.

Architecture and Information Flow in $\pi_{0.5}$. The $\pi_{0.5}$ architecture addresses the challenge of open-world robotic generalization by decoupling high-level semantic reasoning from low-level motor control through a hierarchical “mixture-of-experts” design. The model integrates a *Knowledge Expert*, realized as a large pre-trained Vision-Language Model (VLM) backbone (specifically PaliGemma),

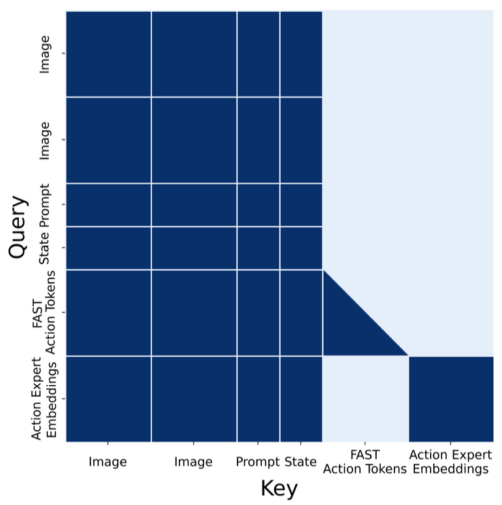


Figure 1: $\pi_{0.5}$ attention mask illustrating token structure: image, prompt, robot state, action tokens, action expert embeddings [3]

which is responsible for processing multimodal inputs—including image sequences, language instructions, and proprioceptive states—to infer high-level semantic subtasks. This backbone leverages web-scale pre-training to handle diverse visual and semantic contexts, allowing the system to reason about complex, long-horizon tasks by breaking them down into simpler components.

Complementing this is the *Action Expert*, a smaller, specialized transformer module (approximately 300M parameters) dedicated to high-frequency motor generation. Unlike the discrete token output of the VLM backbone, the Action Expert operates on continuous action representations using a flow-matching objective, enabling it to model fine-grained, continuous control sequences efficient for real-time execution.

A critical innovation in $\pi_{0.5}$ is the strict enforcement of unidirectional *Information Flow* between these components. The model employs a masked attention mechanism where the Action Expert attends to the embeddings generated by the VLM (the "prefix" containing images, text, and state), but the VLM is explicitly prevented from attending to the Action Expert's embeddings. This design ensures that semantic "knowledge" flows downstream to guide motor execution without "information leakage" flowing back upstream, effectively isolating the high-level reasoning process from the noise of low-level action generation.

This "semantic bottleneck" between the two experts presents a unique opportunity for interpretability: broadly, we hypothesize that the Knowledge Expert's residual stream contains abstract intent (e.g., "pick up the red mug") that a probe pretrained on a text corpus would only have to fit on.

Activation Steering and Contrastive Addition. A central challenge in deploying Large Language Models (LLMs) and Vision–Language–Action models (VLAs) is controlling their behavior without incurring the substantial cost and engineering complexity of retraining. Steering methods address this challenge by directly modifying a model's internal activations during inference, thereby shifting its output distribution away from undesirable behaviors such as hallucination, sycophancy, or unsafe robotic actions. As model scales continue to grow, such techniques increasingly rely on insights from *mechanistic interpretability*, which aims to identify structured, causally meaningful components inside networks—often represented as directions in activation space. However, this endeavor is complicated by the *polysemanticity* of neurons, wherein a single unit may encode multiple unrelated features [5].

Contrastive Activation Addition (CAA), introduced by [8], constructs a "steering vector" by taking the difference in residual-stream activations between carefully chosen pairs of positive and negative examples (e.g., honest vs. sycophantic responses). Injecting this vector into the model at inference time systematically modulates the likelihood of the targeted behavior, enabling fine-grained control without altering model weights. Prior work has shown its effectiveness in reducing sycophancy and enforcing refusal behaviors in Llama-2-Chat. Analogous techniques have recently been extended to robotic control, where contrastive activations derived from prompt variations (e.g., “move fast” vs. “move slow”) can reliably steer VLA policies toward desired motion styles ([6])

Sparse Autoencoders (SAEs) for Feature Extraction. While CAA operates on the dense residual stream, directions in this space are often ”polysemantic,” with a single activation direction encoding multiple unrelated concepts. To address this, Sparse Autoencoders (SAEs) have emerged as powerful tools for disentangling dense representations into sparse, interpretable features.

An SAE is trained to reconstruct activation vectors from a sparse set of latent features. Recent work by [1] demonstrated the utility of SAEs in VLA policies, showing that they can identify features corresponding to discrete actions like ”open gripper” within a Magma-style model. By isolating these features, SAEs allow more precise steering than raw activation addition, as individual semantic components (e.g., ”target object: mug”) can be amplified or suppressed without inadvertently triggering correlated but irrelevant behaviors. However, applying standard LLM-pretrained SAEs to robotic policies often results in domain mismatch, motivating the need for fine-tuning probes on the specific distribution of the VLA, as explored in this work.

The Cost of Embodied Data. A primary bottleneck in robotic learning is the prohibitive cost of data acquisition. Unlike the web-scale text corpora available for LLM training, high-quality robotic demonstrations require expensive teleoperation, physical safety constraints, and real-time human labor. Consequently, training interpretable probes—such as Sparse Autoencoders—from scratch on these limited datasets is not only computationally expensive but also risks overfitting to the narrow distribution of collected trajectories. By contrast, the underlying semantic concepts (e.g., distinguishing ”open” from ”close”) are already robustly learned in the base LLM on vast text datasets. Therefore, a transfer learning approach—initializing probes on the ”easier,” abundant text domain and fine-tuning them for the ”harder” embodied domain—offers a compelling strategy to maximize sample efficiency while retaining rich semantic priors. Most recently, Haon et al. [6] developed a zero-shot activation steering framework. By projecting FFN activations onto token embeddings, they identified semantically meaningful neurons in π_0 and OpenVLA. Injecting sparse vectors aligned to concepts (e.g., “slow”) causally changed robot behavior—without retraining. This method enables interpretable, fine-grained control of embodied policies. Their experiments demonstrated that these sparse interventions could reliably steer real-world robots without modifying model weights or requiring new labeled trajectories. Our work aims to reproduce these findings and extend them by integrating sparse autoencoder probes into the activation editing process.

3 Methodology

To investigate the hypothesis that fine-tuned SAE probes offer superior feature separability compared to token-space clustering in VLA policies, we conduct a comparative analysis using the $\pi_{0.5}$ architecture on the LIBERO benchmark.

3.1 Model and Environment

We utilize $\pi_{0.5}$-LIBERO, a Vision-Language-Action model fine-tuned on the LIBERO framework. The model is built on a PaliGemma backbone, utilizing a Gemma-2B decoder for reasoning and action generation. We select the short-horizon tasks of LIBERO-100 (henceforth referred to as “LIBERO-90”).

3.2 Baseline: Token Clustering

(Note: Baseline implementation details to be added by collaborators.)

3.3 Proposed Method: Latent-Space SAE Finetuning

We propose a mechanistic interpretability framework to steer the high-level intent of the $\pi_{0.5}$ VLA policy. Our approach intervenes on the semantic bottleneck between the model’s vision-language backbone and its action generation module using a finetuned Sparse Autoencoder (SAE).

3.3.1 SAE Pretraining and Domain Adaptation

To effectively capture the sparse features of the VLA's residual stream, we employ a two-stage training pipeline for the SAE:

Stage 1: Text-Only Pretraining. We initialize the SAE parameters θ_{SAE} by pretraining on the residual stream activations of a standard Gemma-2B model trained on a text-only corpus (e.g., OpenWebText). The SAE consists of an encoder $W_{enc} \in \mathbb{R}^{d_{model} \times d_{lat}}$ and a decoder $W_{dec} \in \mathbb{R}^{d_{lat} \times d_{model}}$. We optimize the standard SAE loss, minimizing reconstruction error with an L_1 sparsity penalty on the latent activations z:

$$\mathcal{L}_{SAE} = ||x - \text{Dec}(\text{Enc}(x))||_2^2 + \lambda ||z||_1$$

This initialization leverages the fact that the $\pi_{0.5}$ backbone (PaliGemma) initializes its weights from Gemma-2B [3], suggesting that many semantic features in the residual stream are preserved from the language-only domain.

Stage 2: VLA-Specific Finetuning. We subsequently finetune the SAE on activations collected from the $\pi_{0.5}$ Knowledge Expert during robot instruction following. We freeze the $\pi_{0.5}$ weights and collect residual stream vectors $x \in \mathbb{R}^{2048}$ from the instruction tokens. We continue optimizing $\mathcal{L}_{SAE}$ on this domain-specific distribution. This step is crucial because, while the backbone is initialized from Gemma, the activations in $\pi_{0.5}$ are conditioned on multimodal inputs (images + text) and must encode robotic affordances not present in pure text [3, 1]. More information on finetuning is described in the finetuning section FIXME

Feature Selection via Contrastive Steering. Following [1], we identify steering features by collecting "contrastive pairs" of inputs (I, p_{pos}) and (I, p_{neg})—scenarios where the visual observation I is identical, but the prompts p imply opposing high-level behaviors (e.g., "Pick up the cup aggressively" vs. "Pick up the cup gently"). We compute the difference in mean latent activations $\Delta z = \bar{z}_{pos} - \bar{z}_{neg}$ to isolate the feature f_i responsible for the behavioral shift.

3.3.2 Model Architecture and Probe Placement

Our method exploits the unique "bifurcated" architecture of $\pi_{0.5}$, which explicitly separates semantic reasoning from motor control [3].

The Knowledge Expert (Semantic Backbone): The core of the policy is a PaliGemma VLM (based on Gemma-2B) that processes the visual history $I_{t-k:t}$ and language instruction L. This module is responsible for high-level scene understanding and subtask prediction (e.g., "open the drawer") [3].

The Action Expert (Motor Control): This is a smaller, specialized module (approx. 300M parameters, width 1024) initialized with random weights [3]. It autoregressively generates discrete action tokens (or flow-matching steps) by attending to the embeddings produced by the Knowledge Expert.

Information Bottleneck: Crucially, $\pi_{0.5}$ enforces a strict unidirectional information flow via a "prefix mask" [3]. The Action Expert attends to the Knowledge Expert's embeddings, but the Knowledge Expert *cannot* attend to the Action Expert. This architectural constraint creates a natural "semantic bottleneck": the Knowledge Expert must compress all necessary planning information into its final layer representations before the hand-off. We target this bottleneck for SAE intervention to steer the policy's intent before it is grounded into kinematics.

3.3.3 Layer Selection and Hypothesis

We focus our intervention on **Layer 13** of the 18-layer Knowledge Expert backbone. Our rationale is twofold:

1. **The Semantic-Syntax Transition:** Research in mechanistic interpretability suggests that mid-to-late layers in LLMs (like Layer 13 in a 2B model) typically encode high-level abstract concepts and intent, whereas earlier layers process syntax and local texture, and the final layers specialize in next-token prediction logic [**khan˙et˙al**].

2. **Pre-Action Integration:** In $\pi_{0.5}$, the embeddings passed to the Action Expert serve as the "context" for motor generation. By intervening at Layer 13, we aim to modify the *semantic plan* (e.g., the "mood" or "speed" of the task) while allowing the remaining layers (14–18) and the Action Expert to handle the consistent integration of this plan into valid robot actions. This allows for better generalization across different embodiments, as semantics, scene understanding, and robot proprioception are insulated within the Knowledge Expert, away from the specifics of translating this understanding to movement in the Action Expert. As a result, our intervention to remain closer to the semantic understanding of the network and away from action tokens, instead passing a stronger instruction-level signal of our concept for interpretation by the Action Expert.

3.4 Evaluation Metric: Concept Separability

To quantify the precision of our steering interventions, we evaluate *Concept Separability*: the degree to which the model disentangles semantically distinct instructions within its internal representations. We measure the "entanglement" between two theoretically orthogonal concepts: a manipulation primitive (e.g., *"Open"*) and a spatial trajectory (e.g., *"Front"*). Our hypothesis is that a clean semantic representation should treat these as separate axes, whereas a noisy representation will exhibit high correlation (entanglement) between them.

Specifically, we test

We define the entanglement score $\mathcal{E}$ as the absolute cosine similarity between the direction vectors $\mathbf{v}_{open}$ and $\mathbf{v}_{front}$ in the residual stream of Layer 13:

$$\mathcal{E}(\mathbf{v}_{open}, \mathbf{v}_{front}) = \frac{|\mathbf{v}_{open} \cdot \mathbf{v}_{front}|}{||\mathbf{v}_{open}||_2 ||\mathbf{v}_{front}||_2} \tag{1}$$

A lower score indicates better separability (orthogonality). We compare $\mathcal{E}$ across two methods of deriving these direction vectors:

3.4.1 Baseline: Token-Space Centroid Mapping

For our baseline, adapted from [6], we derive steering vectors by reverse-mapping the target concept from the output vocabulary into the residual stream.

1. **Token-Space Clustering:** We first identify the set of discrete action tokens $\mathcal{T}_{open}$ corresponding to the "Open Gripper" command. We compute the *token-space centroid* c_{open} by aggregating the embedding vectors of these tokens from the model's final embedding layer.
2. **Activation Isolation:** We then map this centroid back to the residual stream to find the network states responsible for generating it. Specifically, we iterate through the dataset and collect the residual stream activations $x_t^{(13)}$ at Layer 13 that are "most relevant" to c_{open}—defined as the activations that result in a high log-probability for the tokens in the target cluster.
3. **Vector Computation:** We compute the baseline vector $\mathbf{v}_{base}^{open}$ as the mean of these isolated Layer 13 activations. This represents the average internal state of the network when it is effectively primed to output the "Open" token centroid.

3.4.2 Proposed Method: Latent-Space Decoding

Our proposed method derives vectors by isolating specific features learned by the Sparse Autoencoder (SAE). This approach aims to strip away the noise and contextual interference present in the dense residual stream.

1. **Contrastive Steering:** To derive steering directions, we process contrastive manipulation segments identified in our manifest. For each entry, we replay the corresponding episode through $\pi_{0.5}$, extracting the residual stream activations at Layer 13 of the Knowledge Expert. These residuals are encoded using our trained SAE to obtain sparse latent representations.

   Following the methodology of **?**, we aggregate these latents by task label and compute their averages, denoted as $\boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_B$ for a generic pair of contrastive tasks (e.g., *open*

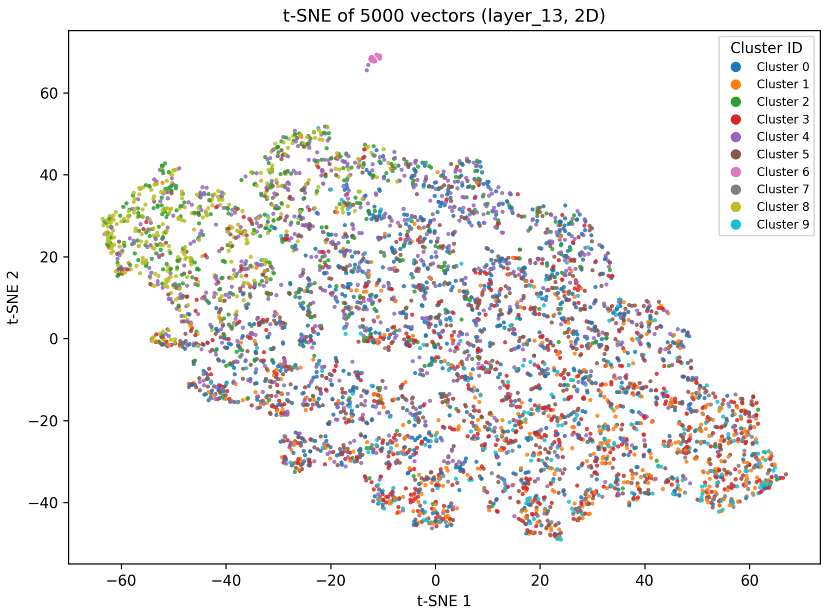


Figure 2: t-SNE visualization of 5k sampled layer-13 FFN vectors from OpenVLA, colored by the ten k-means clusters used in the interpretability-to-steering pipeline; the fast/quick cluster (cluster 4) is the direction injected during the LIBERO Spatial fast-layer-13 intervention

vs. *close*). This aggregation is essential to mitigate the significant trajectory-level variance inherent in the demonstrations available to us in the LIBERO set. We define the latent steering vector $\mathbf{v}_{\text{steer}}$ as the difference between these means:

$$\mathbf{v}_{\text{steer}} = \boldsymbol{\mu}_A - \boldsymbol{\mu}_B. \tag{2}$$

In subsequent sections, we evaluate the disentanglement and steering efficacy of these SAE-derived directions against baselines such as token-space clustering.

2. **Decoding:** We obtain the residual vector $\mathbf{v}_{SAE}$ by decoding this single feature vector back into the residual space using the SAE's decoder weight matrix W_{dec}:

$$\mathbf{v}_{SAE}^{open} = W_{dec}[:, k_{open}]$$

By repeating this process for the positive vs. negative contrastive pair, we obtain $\mathbf{v}_{SAE}^{open}$ and calculate the entanglement $\mathcal{E}$ between decoded directions of unrelated concepts.

4 Experiments

4.1 Preliminary LIBERO Spatial Results (fast-layer-13 steering)

Reproduction of Haon's pipeline We completed the interpretability-to-steering loop for OpenVLA and kicked off the LIBERO Spatial evaluation with the fast layer-13 steering vector, but we paused the full sweep when it became clear the run would take too long and inflate compute costs. Instead, we focused on a targeted slice—108 steered episodes and three contemporaneous unsteered controls —that captures the core behavioral effects we care about and keeps the experiment tractable. These intentionally small samples, explicitly labeled as such in the accompanying table, already show that the steering intervention behaves as expected. The figure visualizes the decomposition into ten groups of OpenVLA's layer-13 feedforward projections that grounds this fast-steering analysis. Each point represents a sampled FFN vector embedded via t-SNE and colored by its k-means assignment; the cluster we steer corresponds to cluster 4 region whose centroid's top logits emphasize *fast/quick* tokens, while neighboring clusters capture bowl semantics, spatial references, and neutral motion cues. Consistent with the findings of Haon et al.[6], where FFN-derived semantic clusters naturally overlap rather than forming rigid partitions, our layer-13 geometry exhibits the

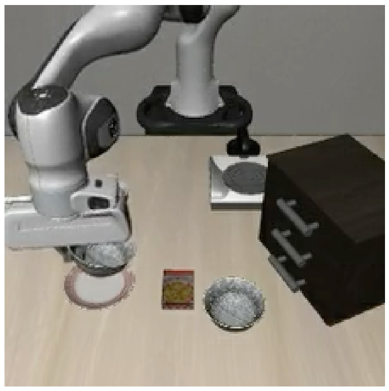


Figure 3: Still frame of control rollout

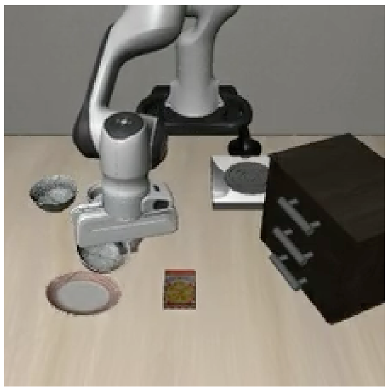


Figure 4: Still frame of rollout with steering

same pattern. This overlap reinforces that the activation direction used in our 108 steered LIBERO Spatial episodes originates from a semantically meaningful—but not artificially isolated—region of the model's value-vector space.

Steering Inject and Rollouts To make the behavioral delta tangible, we paired two GIFs—one from baseline episode=99 (control) and one from steered episode=13. The control clip shows the unmodified policy grasping the black bowl cleanly and placing it beside the ramekin, matching the 3/3 success rate. In contrast, the steered clip shows the injected fast-layer-13 direction pushing high-velocity tokens into the action stream earlier: the gripper accelerates across the plate, overshoots the rim, and repeatedly bumps the bowl, mirroring the 1/108 success count from the steered log slice. These two trajectories clarifies what the fast cluster actually does in practice—it reliably induces the "go faster" behavior predicted by Figure2, even though the added aggressiveness currently reduces task completion.

Despite the limited data, this section already demonstrates (1) the full interpretability pipeline is operational, (2) the injected direction measurably alters the action token distribution, and (3) we have the logging and visualization artifacts needed to expand into the complete set of experiments described in the paper.

4.2 Fine-Tuned Sparse Autoencoder on $\pi_{0.5}$ Residuals

To ensure convergence, we conduct a preliminary hyperparameter sweep, specifically ablating normalization schemes and decoder weight freezing strategies to identify a stable training configuration.

After identifying a stable hyperparameter configuration, we fine-tuned a pretrained Gemma-2B sparse autoencoder (SAE) specifically on the Layer 13 residual stream of the $\pi_{0.5}$ Knowledge Expert. We obtain these pretrained checkpoints from SAE Lens, a package created by [4] containing a repository of pretrained LLM SAE probes. This layer acts as a semantic bottleneck in the policy, receiving fused visual-language representations and producing task-relevant latent structure just before the Action Expert decodes motor commands. Because the pretrained SAE originates from a pure language model, its feature basis does not naturally align with the geometry of a VLA model trained on robotic demonstrations. Fine-tuning the SAE on actual $\pi_{0.5}$ activations is therefore essential for recovering monosemantic features that faithfully reflect embodied semantics.

Training procedure. We fine-tuned only the encoder parameters while freezing the decoder weights and biases. This strategy is recommended by [2] for distribution-shifted domains, as the pretrained decoder contains a well-structured, high-quality dictionary of monosemantic features. Adjusting only the encoder allows the model to remap $\pi_{0.5}$ activations into this feature basis without distorting it. The training was performed on $10{,}074{,}706$ instruction-token residuals using mixed-precision (AMP), pinned-memory loading, 128-sample batches, and Adam optimization. The process converged smoothly within 10–20 epochs, producing stable sparsity and reconstruction curves.

Effect on representation geometry. Fine-tuning significantly altered how $\pi_{0.5}$ activations map into latent space. The pretrained SAE assigned many residual vectors to dense, overlapping factors, while the fine-tuned SAE learned sharper, more orthogonal latent codes. We observe:

- **Higher sparsity.** The average number of active units increased from ~ 1700 to ~ 3000 (for a 16k dictionary), indicating that the encoder is selectively utilizing a broader set of meaningful features.
- **Lower reconstruction error.** MSE decreased from ~ 60 to ~ 11, confirming that the fine-tuned encoder captures geometry specific to the VLA domain.
- **Improved cluster separability.** Latent vectors corresponding to "open gripper," "close gripper," and "transport" instructions form cleaner, more linearly separable clusters compared to those obtained from the pretrained SAE.

This improved structure directly benefits the downstream interpretability analysis.

4.3 Constructing Steering Directions Using the Fine-Tuned SAE

Having trained a sparse autoencoder (SAE) specifically on the Layer 13 residual stream of the $\pi_{0.5}$ Knowledge Expert, we next use the model to extract steering directions that correspond to high-level behavioral concepts. Following the methodology laid out by [1], we focus on constructing an "open–close" steering direction, as the gripper state provides a clean, binary behavioral attribute with clear causal effects on downstream actions.

Using Layer 13 as the semantic locus. Layer 13 is a late residual layer just before action decoding and serves as a semantic bottleneck where intent is resolved into motor-relevant structure. Prior work on activation steering in VLA models suggests that such late-residual layers contain linearly editable representations of affordances and object interactions. Because the SAE was fine-tuned on the true Layer 13 distribution, it provides a faithful basis for decoding these structural features.

5 Results

See plots next page.

We report a negative result regarding the geometry of the learned latent space. Contrary to our hypothesis that the SAE would disentangle tasks into orthogonal subspaces (as observed in the token-space clustering baseline), we find that contrastive tasks become antipodally aligned (characterized by high negative cosine similarity). In fact, the token-clustering method establishes near orthogonal directions between the tokens.

We theorize that this phenomenon stems from the hierarchical nature of the $\pi_{0.5}$ architecture. Since we probe the Knowledge Expert—which is architecturally insulated from the high-frequency dynamics of the Action Expert—the extracted features reflect semantic polarity (treating 'open' and 'close' as linguistic antonyms) rather than geometric distinctness. The SAE effectively recovers the high-level instructional structure, where tasks are logical opposites, rather than the physical manifold structure, where trajectories would be distinct but not necessarily opposed.

6 Limitations

Compute. A practical limitation of our current work is the availability and stability of compute resources for activation logging and sparse autoencoder training. We initially attempted to use the shared GPU cluster provided by the OCF, but resource contention and queue variability made it difficult to run experiments reproducibly. We then explored several cloud providers, including AWS and Lambda Labs, and ultimately relied on these services because we already had credits available and they provided more predictable access to GPU instances.

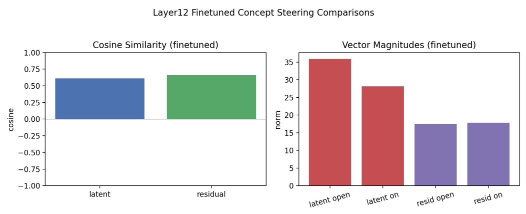


Figure 5: Cosine similarity of ”Open”, ”On” through finetuned probe

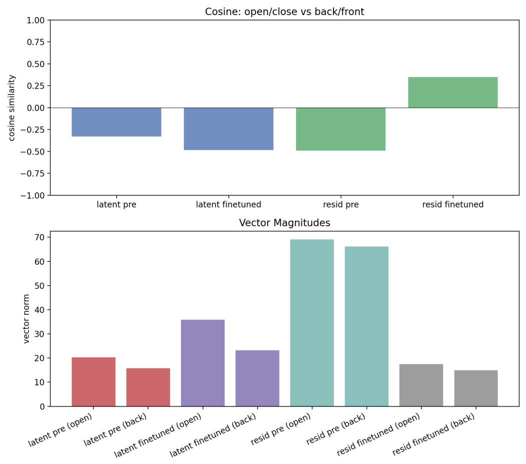


Figure 6: Cosine similarity of ”Open”, ”Back”

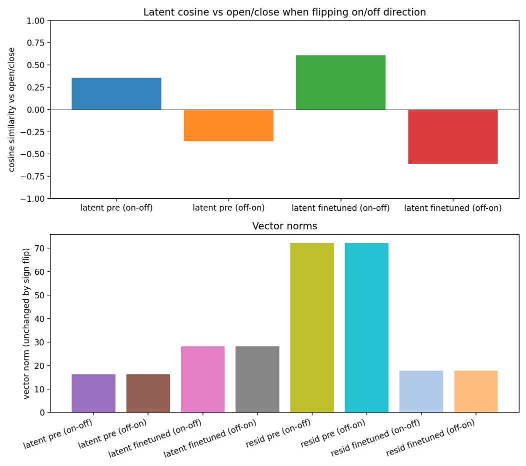


Figure 7: Cosine similarity of ”Open” between ”Off”, ”On”

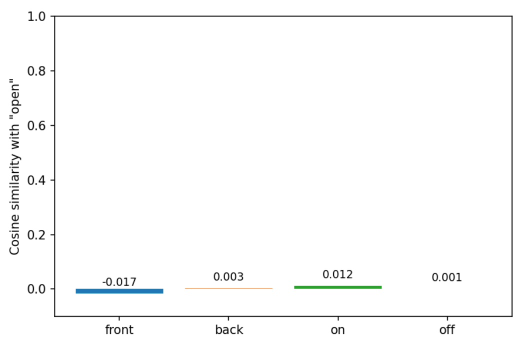


Figure 8: Cosine similarities computed using baseline method

7 Conclusion

In this work, we reproduced and extended the activation-level steering framework of [6], examining whether fine-tuned Sparse Autoencoders can provide a semantically meaningful basis for controlling VLA policies. Due to the availability of pretrained dictionaries, our SAE analysis focuses on the $\pi_{0.5}$ architecture, while zero-shot token-space steering is applied to OpenVLA, yielding a complementary evaluation across two distinct model families. Notably, when evaluated under the original Haon et al. protocol on LIBERO, injected steering vectors provided negative or inconsistent effects for OpenVLA, suggesting that token-space directions may not transfer reliably across VLA architectures or tasks.

In contrast, our analysis of $\pi_{0.5}$ reveals a rigid semantic bottleneck: while SAEs successfully recover task-relevant directions, we find these features exhibit **antipodal alignment** rather than orthogonal disentanglement. This suggests that the model's architectural decoupling of "Knowledge" and "Action" experts fosters a representation of semantic polarity (encoding tasks as logical opposites) rather than independent control primitives. This could be because of the architecture of our model insulating instruction/task-level semantic understanding in the Instruction Expert, where we probe, away from motor control understanding in the Action Expert. These findings underscore that activation-level steering depends not only on the intervention method but also on the specific representational geometry imposed by the model's hierarchy. Future work will complete the comparative evaluation on real and simulated benchmarks, investigate whether this antipodal structure limits or simplifies control, and explore architecture-agnostic SAE dictionaries for fine-grained steering.

We welcome peer feedback on:

- correctness of the environment reconstruction,
- data-collection methodology,
- missing implementation details from the original paper,
- reproducibility assumptions.

References

[1] Momin Ahmad Khan et al. "Controlling Vision–Language–Action Policies through Sparse Latent Directions". In: *NeurIPS 2025 Workshop on Mechanistic Interpretability*. 2025.

[2] Anthropic. *Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet*. Tech. rep. Anthropic Technical Report, 2024.

[3] Kevin Black et al. $\pi_{0.5}$*: a Vision-Language-Action Model with Open-World Generalization*. 2025. arXiv: `2504.16054 [cs.LG]`. URL: `https://arxiv.org/abs/2504.16054`.

[4] Joseph Bloom et al. *SAELens*. `https://github.com/decoderesearch/SAELens`. 2024.

[5] Tim Bricken et al. "Sparse Autoencoders Learn Monosemantic Features in Vision–Language Models". In: *Anthropic Research* (2024).

[6] Benjamin Haon et al. "Mechanistic Interpretability for Steering Vision–Language–Action Models". In: *arXiv preprint arXiv:2509.00328* (2025).

[7] Myung Jin Kim et al. "OpenVLA: An Open-Source Vision–Language–Action Model". In: *arXiv preprint arXiv:2406.09246* (2024).

[8] Nina Panickssery et al. "Steering Llama 2 via Contrastive Activation Addition". In: *arXiv preprint arXiv:2312.06681* (2023). URL: `https://arxiv.org/abs/2312.06681`.

[9] Brianna Zitkovich et al. "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control". In: *Proceedings of The 7th Conference on Robot Learning*. Vol. 229. Proceedings of Machine Learning Research. 2023, pp. 2165–2183. URL: `https://robotics-transformer2.github.io/`.

8 Appendix

Justification for Mean Subtraction. Unlike prior work (e.g., Khan et al.) that utilizes physical hardware to generate perfect counterfactuals by freezing the environment state, we are constrained to pre-collected demonstrations from the LIBERO dataset. To approximate this isolation, we select task pairs that differ by a single semantic instruction. However, because these are distinct trajectories rather than controlled state interventions, the resulting latent representations contain significant variance due to trajectory styles and incidental correlations.

We model the latent vector $\mathbf{z}$ as an additive superposition of shared environmental features $\mathbf{z}_{\text{env}}$, task-specific semantics $\mathbf{z}_{\text{task}}$, and trajectory noise $\boldsymbol{\epsilon}$. By averaging over the intra-class variance of each task, we effectively marginalize out the nuisance noise $\boldsymbol{\epsilon}$. Furthermore, because the environmental context remains semantically consistent across the contrastive pair, the subtraction operation cancels out the shared $\mathbf{z}_{\text{env}}$ component:

$$\mathbf{v}_{\text{steer}} \approx (\mathbf{z}_{\text{env}} + \mathbf{z}_{\text{task}}^{(A)}) - (\mathbf{z}_{\text{env}} + \mathbf{z}_{\text{task}}^{(B)}) = \mathbf{z}_{\text{task}}^{(A)} - \mathbf{z}_{\text{task}}^{(B)}. \tag{3}$$

This isolates the axis of variation corresponding strictly to the semantic difference between the tasks, removing activation components common to both behaviors (e.g., object identity, scene layout) while preserving the differential representation responsible for the task switch.

8.1 Contrastive Steering Episodes

[1] construct steering directions from contrastive pairs of trajectories that differ only in a targeted attribute (e.g., "open gripper" versus "close gripper"). We adapt this principle to the LIBERO-90 benchmark by extracting contrastive manipulation segments from pairs of tasks that operate on the same object but in opposite directions. Rather than treating entire episodes as positive or negative samples, we focus specifically on the time interval where the relevant object coordinate moves monotonically. This approach better aligns with the localized steering assumption inherent to contrastive activation addition.

State Dimension Selection. Data processing commences with the raw LIBERO-90 HDF5 demonstrations for two binary task pairs: *opening versus closing the top cabinet drawer* and *turning a stove burner on versus off*. Each demonstration consists of a sequence of low-dimensional robot states $s_t \in \mathbb{R}^{47}$. For each task, we compute the terminal state difference $\Delta s = s_T - s_0$ per demonstration and average this difference across all demonstrations.

We rank the state dimensions by the absolute difference between the means of the two contrasting tasks. We then select the first index where the means exhibit opposite signs and exceed a minimum displacement threshold. This procedure identifies:

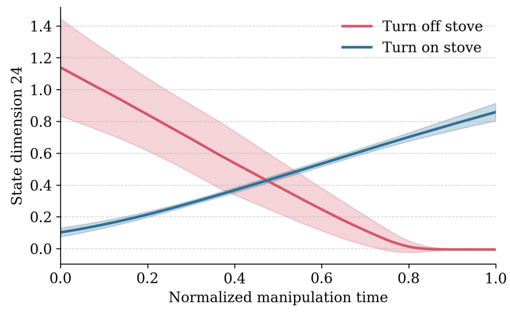
- **Dimension 38:** A signed drawer displacement coordinate.
- **Dimension 24:** A stove-knob rotation coordinate.

In both cases, the selected coordinate cleanly separates the two behaviors while leaving the remainder of the state vector largely unchanged.

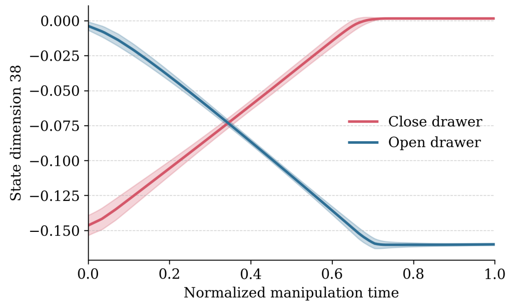
Temporal Segmentation. Given the axis index for a task pair, we isolate the specific manipulation window within each demonstration. Let k_t denote the chosen state coordinate for a single demo at time t. We compute the cumulative displacement $d_t = k_t - k_0$ and identify the start (t_{start}) and end (t_{end}) timesteps as the points where $|d_t|$ exceeds a fixed fraction of the total displacement $|d_T|$. This "progress band" isolates the interval where the drawer or knob is actively moving. We pad this interval by a small buffer on both sides to capture contact initiation and release. For demonstrations where the terminal displacement is negligible, we utilize a fallback heuristic based on velocity, selecting the region with the largest finite differences $|k_{t+1} - k_t|$. We compile the task label, file metadata, and raw axis values for each segment into a contrast manifest to drive subsequent analysis.

Validation of Trajectories. To validate the semantic consistency of these segments, we normalize each manipulation window to unit length in the temporal domain and linearly resample it to a fixed-length grid before averaging (see Figure 9).

- **Drawer Axis (Dim 38):** The mean trajectories for "open" and "close" form nearly linear, oppositely directed curves with relatively low variance during the motion phase. Higher variance is observed at early normalized timesteps for the closing task, as the drawer begins from varying partially open configurations and may undergo corrective pushes before the primary closing stroke.
- **Stove Axis (Dim 24):** The "turn on" and "turn off" trajectories appear approximately linear and symmetric. We observe greater dispersion early in the turn-off motion, attributable to heterogeneous approach strategies before the counter-clockwise twist is applied.



(a) Stove knob axis (Dim 24)



(b) Drawer axis (Dim 38)

Figure 9: **Mean normalized trajectories of state dimensions identified by our contrastive procedure. (a)** "Turn on stove" vs. "turn off stove" on state dimension 24 (knob rotation). **(b)** "Open drawer" vs. "close drawer" on state dimension 38 (drawer displacement). Each curve averages 50 demos resampled to a fixed-length window; shaded bands indicate ± 1 standard deviation.

8.2 Hyperparameter Search for Sparse Autoencoder Fine-Tuning

To adapt pretrained language-model SAEs to the residual distribution of the $\pi_{0.5}$ Knowledge Expert, we conducted a structured hyperparameter sweep following methodological guidance from **??**. Unlike classical language-model residuals, the activations in $\pi_{0.5}$ arise from a VLA policy trained jointly on visual features, spatial reasoning, and robot-control signals, introducing a significant distributional shift. Our sweep therefore evaluated both standard SAE hyperparameters and input-normalization strategies specific to this domain.

Learning rate sweep. We evaluated the learning rates $\{3\times10^{-5},\ 1\times10^{-4},\ 3\times10^{-4}\}$ using batch size 128 and identical initialization. Consistent with prior SAE literature, very small rates underfit, while large rates caused instabilities in the encoder update dynamics. The choice $\mathbf{1\times10^{-4}}$ yielded the fastest and most stable decrease in reconstruction error without overshooting, and is the setting recommended in **?** for adapting pretrained SAEs to shifted activation distributions.

Sparsity penalty (λ). We swept $\lambda \in \{10^{-5},\ 3\times10^{-5},\ 10^{-4}\}$ following **?**, who emphasize that sparsity penalties that are too low lead to dense, uninterpretable features, while penalties that are too high suppress useful activations. We found that $\mathbf{3\times10^{-5}}$ produced the most stable balance between reconstruction quality and latent sparsity. Higher values amplified training variance, while lower values significantly reduced the number of active units.

Normalization strategy. We evaluated three normalization schemes:

1. no normalization,
2. per-token ℓ_2 normalization,
3. the expected-average-input normalization used by Anthropic for language-model SAEs.

Surprisingly—and in contrast with classical LLM activation modeling—the best performance for $\pi_{0.5}$ residuals was obtained with **no normalization**. Both normalization-based schemes increased reconstruction error and produced less stable sparsity patterns, as indicated by the ℓ_0 count (number of active SAE features per sample).

This divergence from the behavior observed in **?** is consistent with the domain-shift hypothesis: residuals in a VLA policy reflect multimodal grounding, geometry, proprioception, and visuomotor fusion, rather than the token-distribution-driven statistics found in language models. Because normalization implicitly encodes assumptions about the distributional scale of activations, applying LLM-normalization procedures distorted the SAE's learned geometry. We therefore adopt **no input normalization** for all downstream experiments.

Full fine-tuning on $\pi_{0.5}$ activations. Using the chosen hyperparameters, we fine-tuned the SAE encoder on all 10,074,706 residual activations from Layer 13.

Final hyperparameters.

Hyperparameter	**Value**
Learning rate	1×10^{-4}
Batch size	128
Sparsity penalty λ	3×10^{-5}
Normalization	**None** (best for $\pi_{0.5}$ activations)
Decoder parameters	Frozen
Epochs	10–20
Precision	Mixed (AMP)