Sparse Autoencoders Reveal Interpretable and Steerable Features in VLA Models

Aiden Swann[1], **Lachlain McGranahan**[3], **Hugo Buurmeijer**[3]
Monroe Kennedy III[1,2], **Mac Schwager**[3]
[1]Department of Mechanical Engineering, [2]Department of Computer Science
[3]Department of Aeronautics & Astronautics
Stanford University

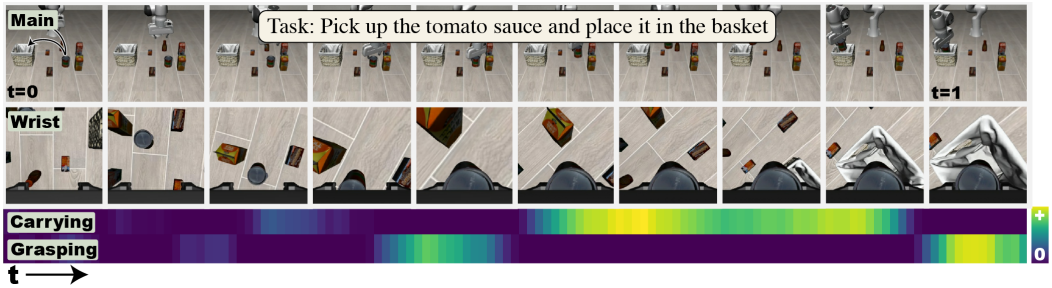


Figure 1: Activations for two general features related to grasping: F1902, F1129 in the $\pi_{0.5}$ model's Paligemma Layer 5 over an episode of LIBERO. F1129 and F1902 are general features for grasping and carrying behaviors respectively.

Abstract: Vision-Language-Action (VLA) models have emerged as a promising approach for general-purpose robot manipulation. However, their generalization is inconsistent: while these models can perform impressively in some settings, fine-tuned variants often fail on novel objects, scenes, and instructions. We apply mechanistic interpretability techniques to better understand the inner workings of VLA models. To probe internal representations, we train Sparse Autoencoders (SAEs) on hidden layer activations of the VLA. SAEs learn a sparse dictionary whose features act as a compact, interpretable basis for the model's computation. We find that the large majority of extracted SAE features correspond to memorized sequences from specific training demonstrations. However, some features correspond to interpretable, general, and steerable motion primitives and semantic properties, offering a promising glimpse toward VLA generalizability. We propose a metric to categorize features according to whether they represent generalizable transferable primitives or episode-specific memorization. We validate these findings through steering experiments on the LIBERO benchmark. We show that individual SAE features causally influence robot behavior. Steering general features induces behaviors consistent with their semantic meaning and can be applied across tasks and scenes. This work provides the first mechanistic evidence that VLAs can learn generalizable features across tasks and scenes. We observe that supervised fine-tuning on small robotics datasets disproportionately amplifies memorization. In contrast, training on larger, more diverse datasets (e.g., DROID) or using knowledge insulation promotes more general features. To facilitate future research in VLA mechanistic interpretability, we provide an open-source codebase and user-friendly interface for activation collection, SAE training, and feature steering. Our project page is located at drvla.github.io.

Keywords: Vision-Language-Action Models, Sparse Autoencoders, Mechanistic Interpretability, Robot Learning

correspondence to {swann, lachmcg}@stanford.edu

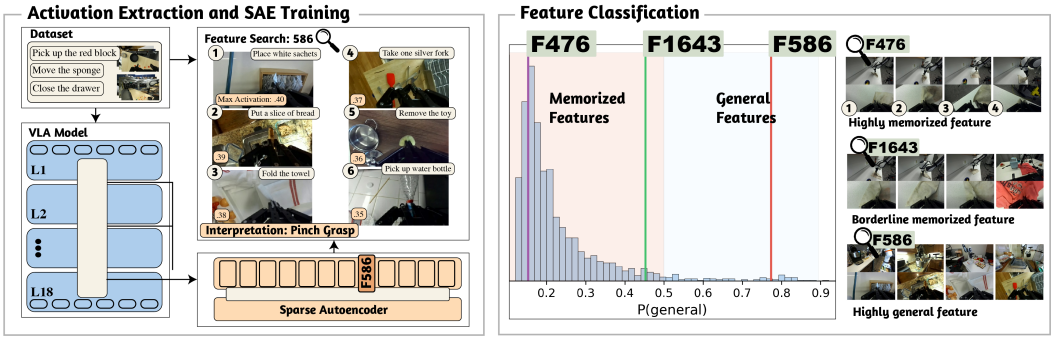


Figure 2: Overview of our mechanistic interpretability pipeline for VLA models. We collect internal activations from a VLA, train a Sparse Autoencoder (SAE), and obtain sparse features that represent both memorized episodes, as well as general semantic concepts, motion primitives and task structure. We propose a metric to categorize features by generality. Right: three grasp-related features on DROID, visualized by the top four activating episodes each. Memorization features fire on highly similar scenes, while general features activate across diverse scenes, tasks, and grasp types.

1 Introduction

The field of robot manipulation is increasingly shaped by research in generalist policies that combine visual inputs, natural language instruction, and continuous control outputs into a single learned system. The primary example of such a policy architecture is the Vision-Language-Action (VLA) model (e.g. RT-2 [1], OpenVLA [2] and Pi0 [3]). VLA models typically couple a pretrained vision language model (VLM) backbone with a separate action decoding head. These models are pretrained on large, heterogeneous, cross-embodiment robot datasets such as OpenX Embodiment [4] or DROID [5].

The motivation for using VLA models is simple. Large language models (LLMs) and vision language models (VLMs) achieve impressive generalization across a wide variety of tasks [6, 7]. Specifically, these frontier models learn rich representations that enable generalization across text, objects, and spatial relations. VLAs attempt to leverage this widespread semantic-visual knowledge through a VLM backbone with the goal of obtaining broad generalization to a variety of robot tasks in diverse visual environments, commanded by open-vocabulary language prompts. However, current VLAs suffer from several pitfalls. Typically, VLAs must be fine-tuned on a specific task or embodiment to perform well. Despite rapid empirical progress in benchmarks like LIBERO [8] or Robocasa [9], these models often lose language following and generalization abilities during supervised fine-tuning (SFT). Furthermore, papers like LIBERO-PRO [10] have shown that models that exceed 90% success rate under the original protocol can collapse to near-zero under systematic perturbations, implying that these policies may rely on rote memorization of action sequences and environment layouts rather than generalizing to new perceptual inputs.

In order for robots to move from structured lab settings into the wild, these policies we must understand how to engineer VLA policies to be generalizable, reliable, and interpretable. Classically, "safety" has been defined by collision avoidance, force limits, and stability. As robots transition into the wild, notions of "safety" must also adapt to accommodate a broader safety paradigm. "Safety" should no longer be just geometric or control-theoretic, but also semantic, situational, and task-dependent. For example, a home robot must avoid contact with hot surfaces and handle sharp or fragile objects appropriately. In light of this paradigm shift, we believe that the first step toward building trustworthy robots is to look past empirical success rates and focus on mechanistic understanding of VLA policies. We hope this will unlock the ability to steer these policies toward more desirable behavior.

However, we still do not have a firm grasp on how these large models work. As a result, most explanations for such brittleness are behavioral and anecdotal rather than rooted in analysis. Specifically,

we currently lack the tools to understand which parts of the model encode transferable concepts or are memorizing the data, despite being able to observe failures. Drawing inspiration from the LLM community [11, 12, 13], this work uses a set of tools for understanding the inner workings of learned models called mechanistic interpretability. Specifically, we leverage Sparse Autoencoders (SAEs), an unsupervised learning technique that disentangles superimposed representations by projecting dense activations onto a higher-dimensional sparse latent space. This technique is known to find highly intepretable features in LLM models [14], and we show that this trend continues with VLAs.

In this work, we apply SAEs to the residual streams of VLA models. Crucially, we verify that steering individual features during inference can predictably modulate robot behavior. Empirically, we find that SAEs uncover an interpretable basis for VLA computation, including features aligned with motion primitives, task completion, and language-relevant semantics. We also observe that VLAs contain memorization features whose activation enforces task-specific action sequences, offering mechanistic evidence that supervised fine-tuning on small datasets can induce trajectory-level memorization rather than compositional skill learning. Our contribution is as follows:

- We propose a technique to extract **interpretable**, **general**, and **steerable** features from VLA models using Sparse Autoencoders.
- We propose several **generality quantification metrics** for SAE features using activation statistics on the fine tuning data, without running any policy roll-outs.
- We observe significant **episode-level memorization** in all cases. However, fine-tuning on diverse datasets such as **DROID** leads to comparatively more **general** features.
- We offer **Dr. VLA**, an **open-source** and user-friendly software package for **SAE training**, **feature evaluation**, and **policy steering** in VLAs.

2 Related Work

2.1 Mechanistic Interpretability of VLMs

Many interpretability methods have been proposed in the literature, including causal tracing, activation patching, and probes [15]. Among these are sparse autoencoders (SAEs), which are an unsupervised method for discovering latent features or patterns in the model's activations that are not known a priori. The modern SAE interpretability paradigm was popularized by *Towards Monosemanticity* [16], which applied SAEs towards a single-layer transformer, extracting 4000+ interpretable features. Further work has scaled this approach to production-grade models [12, 11, 14], including applications to vision-language models (VLMs) [17]. Central to this approach is the Linear Representation Hypothesis [18], which suggests that concepts are represented as directions in the model's activation space. However, direct inspection of these directions is hindered by superposition. The goal of these works is to decompose the highly superposed representations of the residual stream into monosemantic features. A prominent example of this ability is Golden Gate Claude, in which a single monosemantic feature for the Golden Gate Bridge was isolated [19] and used for feature-level steering. Since VLAs typically reuse the same VLM backbones, our work builds on this interpretability framework and adapts SAE-based analyses and interventions to the embodied control setting. We aim to show that VLAs exhibit many of the same interpretability features as LLMs.

2.2 Mechanistic Interpretability of VLAs

Mechanistic Interpretability of VLAs is a nascent field with only a handful of works released in the past year. Existing work falls into two categories: (i) *decoding* task-relevant information from internal representations and (ii) *intervening* on internal activations to causally steer behavior. On the decoding side, Lu et al. [20] train linear probes on OpenVLA activations to decode states such as object positions and actions. They find that probe accuracies exceed $>90\%$ for most layers. This suggests that, like in LLMs, VLAs often linearly encode structured information relevant to the task,

even when the end-to-end policy remains a black box. Building on this observation, Molinari et al. [21] move beyond static state decoding and probe toward an emergent "world model" structure by testing whether state-transition vectors —defined via embedding— are recoverable from intermediate activations. While they propose SAEs as a promising next step to interpretable planning, they do not implement them. In parallel, several other works pursue causal interventions. Häon et al. [22] interpret feed-forward network (FFN) directions by projecting them into token space, clustering directions by semantics (e.g., higher/lower, faster/slower), and show that manipulating the corresponding activations can steer actions. One issue is that the underlying FFN features remain highly superposed and thus may not scale cleanly to more abstract or compositional concepts without an explicit feature decomposition, such as SAEs. Khan et al. [23] propose using SAEs to steer VLA models, but they do not analyze whether the resulting SAE features are interpretable. Furthermore their steering method is based on contrastive interventions that can act across many latent directions simultaneously, rather than targeting a single isolated feature. Finally, Buurmeijer et al. [24] span both observation and intervention of VLAs using linear probes. The authors find they can reliably apply control theoretic techniques to steer outputs with minimal linear interventions while operating closed loop. Furthermore they propose leveraging SAEs to discover features without explicit labels as a direction of future work.

2.3 Improving Generalization of VLA models

Many works have sought to address the apparent brittleness of VLA policies beyond the training data, especially after supervised fine-tuning (SFT) on continuous control tasks. One hypothesis is that the lack of generality after fine-tuning arises from a degradation of the model's visual representations. To test this hypothesis, Kachaev et al. [25] propose a visual representation regularizer that aligns the model representations with a frozen teacher model. They find that this helps the fine-tuned model generalize better to spatial reasoning tasks. Similarly, Knowledge Insulation [26] is a method that prevents fine-tuning from degrading internal model representations. It proposes mechanisms such as stop gradients and architectural separation, so that the VLM backbone retains semantics while an expert handles continuous control efficiently. Grover et al. [27] take a different approach to improving generalization. They augment the standard VLA architecture with a dual-vision encoder in which one encoder is frozen, and the other is trainable for adaptation. They find that this enhances visual robustness and instruction generalization. However, one of the biggest challenges in improving generalization is in providing scenarios that are truly out-of-distribution. LIBERO-PRO [10] finds that the standard LIBERO benchmark [8] overestimates performance. Instead, they propose systematic perturbations that force genuine generalization.

3 Methods

Our overall method is shown in Figure 2 and has four main components: SAE training (section 3.1), generality metrics (section 3.2), feature classification (section 3.3), and feature steering (section 3.4). First, we extract activations from the VLA policy and train SAEs on several layers within the model. SAEs decompose superimposed features into a sparse dictionary. Next, in order to process the large quantity of features, we utilize a number of feature-wise metrics that can be computed automatically. We then use these metrics to classify features as general or memorized. Many of these features show promising generalization. Finally, we perform steering on these features to further confirm their causal impact on the model.

3.1 SAE AuxK

To extract interpretable features from the VLA's internal representations, we train sparse autoencoders (SAEs) on residual stream activations following the TopK architecture with AuxK auxiliary loss introduced by Gao et al. [12]. This architecture provides direct control over feature sparsity through the TopK activation function [28] and addresses the dead latent problem through an auxiliary reconstruction loss computed over inactive features [12, 16].

Given an input activation $\mathbf{x} \in \mathbb{R}^d$, the SAE first applies per-sample normalization. A learned pre-bias $\mathbf{b}_{\text{pre}}$, initialized to the geometric median of training activations, is subtracted, followed by per-sample mean subtraction and ℓ_2 normalization:

$$\tilde{\mathbf{x}} = \frac{(\mathbf{x} - \mathbf{b}_{\text{pre}}) - \mu}{\|(\mathbf{x} - \mathbf{b}_{\text{pre}}) - \mu\|_2}, \tag{1}$$

where μ is the scalar mean over the d_{model} dimension. The normalized input is encoded into a sparse representation:

$$\mathbf{z} = \text{ReLU}\big(\text{TopK}(\mathbf{W}_{\text{enc}}(\tilde{\mathbf{x}})\big), \tag{2}$$

where the TopK function retains the k largest pre-activation values and zeros the rest, and a post-selection ReLU ensures non-negative feature coefficients. The reconstruction in normalized space $\hat{\tilde{\mathbf{x}}} = \mathbf{W}_{\text{dec}}\mathbf{z}$ is un-normalized by restoring the saved ℓ_2 norm, mean, and pre-bias. Decoder columns are constrained to unit norm, ensuring that each feature's contribution is determined purely by its activation coefficient. Neither the encoder nor the decoder uses bias terms.

The total loss combines normalized reconstruction error with the AuxK auxiliary loss:

$$\mathcal{L} = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}{C_{\text{MSE}}} + \alpha \cdot \frac{\|\tilde{\mathbf{e}} - \hat{\tilde{\mathbf{e}}}_{\text{aux}}\|_2^2}{C_{\text{MSE}}}, \tag{3}$$

where C_{MSE} is the variance of centered activations computed at initialization with $\alpha = 1/32$. $\tilde{\mathbf{e}} = \tilde{\mathbf{x}} - \hat{\tilde{\mathbf{x}}}$ is the reconstruction residual in normalized space and $\hat{\tilde{\mathbf{e}}}_{\text{aux}}$ is the auxiliary reconstruction from the top-k_{aux} dead latents applied to this residual. A latent is considered dead if it has not activated within the last 500 optimization steps. Encoder weights are initialized as a scaled transpose of the decoder: $\mathbf{W}_{\text{enc}} = \mathbf{W}_{\text{dec}}^\top \sqrt{k/n}$, where n is the residual stream dimension. Decoder gradients are projected onto the tangent plane of the unit-norm constraint [16], and gradient norms are clipped at 1.0. We train SAEs with a $1\times$ expansion ratio (i.e., the latent space has the same dimensionality as the input[1]) on eight layers of the $\pi_{0.5}$ model [29]: PaliGemma layers $0, 5, 11, 17$ ($d = 2048$, $k = 100$) and action expert layers $0, 5, 11, 17$ ($d = 1024$, $k = 64$). We similarly train SAEs for the OpenVLA model [2]. Prior SAE work has typically focused on a single middle layer [16, 19], or a narrow range of adjacent middle layers [12]; we instead select four approximately evenly-spaced layers within each subnetwork (first, early-middle, late-middle, last) to provide coverage across all stages of the representation hierarchy, but our main analysis focuses similarly on the early and late middle layers. Additional hyperparmeter details can be found in Section A.2 and Table 4.

3.2 Generality Quantification Metrics

Each SAE dictionary contains many thousands of individual features. Manually inspecting all of these across many models and layers is intractable, so we define a set of per-feature activation statistics that summarize how each feature behaves within episodes and across the dataset. Together they provide a basis for distinguishing general features from memorized features, enabling automated classification at scale (Section 4).

Let $f_j(\mathbf{x}_t^{(e)}) \in \mathbb{R}_{\geq 0}$ denote the activation of SAE feature j at timestep t in episode e, and let $T^{(e)}$ be the number of timesteps in episode e. Let $E = \{e_1, e_2, \ldots, e_N\}$ denote the set of all episodes in the fine-tuning dataset and $E_j^+ = \{e : \exists\, t,\ f_j(\mathbf{x}_t^{(e)}) > 0\}$ be the set of episodes in which feature j fires at least once.

Episode Coverage. Episode coverage is the fraction of episodes in the fine-tuning dataset where the feature activates at least once:

$$c_j = \frac{|E_j^+|}{|E|}. \tag{4}$$

Higher episode coverage indicates the feature is active across diverse tasks and therefore has greater generality.

[1] We find that larger expansion ratios lead to far greater dead feature percents while offering similar interpretability. See Section A.2 for more discussion and ablations on expansion ratio.

Mean Onset Count. An onset of a feature is the transition of that feature from inactive to active. Rather than defining an active feature as simply $f_j(\mathbf{x}_t^{(e)}) \neq 0$ for a specific timestep, we instead add an activation threshold τ_{on} to suppress the effects of noise. Each feature j maintains a binary state s with $s_0 = 0$. The state transitions are:

$$s_t = \begin{cases} 1 & \text{if } f_j(x_t) > \tau_{\text{on}} \\ 0 & \text{if } f_j(x_t) = 0 \\ s_{t-1} & \text{otherwise} \end{cases}, \tag{5}$$

with activation threshold $\tau_{\text{on}} = 0.1$. An onset is counted at each $0 \to 1$ transition, giving the per-episode onset count:

$$o_j = \sum_{t=1}^{T} \max(0, s_t - s_{t-1}). \tag{6}$$

The mean onset count averages over active episodes only, decoupling it from episode coverage:

$$\bar{o}_j = \frac{1}{|E_j^+|} \sum_{e \in E_j^+} o_j^{(e)}. \tag{7}$$

Since every active episode has at least one onset, $\bar{o}_j \geq 1$ for any feature with $c_j > 0$. General features typically exhibit $\bar{o}_j \gg 1$, indicating bursty, event-driven activation.

Mean Activation Magnitude. For each active episode $e \in E_j^+$, we record the maximum activation of feature j across all timesteps. The mean activation magnitude averages these per-episode maxima:

$$\bar{a}_j = \frac{1}{|E_j^+|} \sum_{e \in E_j^+} \max_t f_j(\mathbf{x}_t^{(e)}). \tag{8}$$

This metric captures the typical peak intensity of a feature when it fires, averaged across all episodes in which it is active.

Relative Run Length. For each active episode, the run length is the mean number of consecutive active timesteps per onset:

$$r_j = \frac{1}{o_j} \sum_{t=1}^{T} s_t. \tag{9}$$

Normalizing by episode length yields the relative run length, which expresses the average activation duration as a fraction of the episode:

$$\bar{\ell}_{r,j} = \frac{1}{|E_j^+|} \sum_{e \in E_j^+} \frac{r_j^{(e)}}{T^{(e)}}. \tag{10}$$

Values near 0 indicate brief, transient activations; values near 1 indicate sustained activation across the full episode. General features tend to have low $\bar{\ell}_r$ (bursty), while memorized features often exhibit high $\bar{\ell}_r$ (sustained).

3.3 Feature Classification

Using the metrics proposed above we can more effectively sort through the large number of SAE features. We hand label a random subset of SAE features as *general* or *memorized*, and train a classifier that uses the above metrics in a logistic regression model to determine for each SAE feature whether it is general or memorized.

3.3.1 Categories of Features

We first must rigorously define what a general, memorized or interpretable feature is. Prior work on SAE interpretability in language models evaluates features primarily through top-activating examples: a feature is interpretable if a human can identify a common pattern among the inputs that most strongly activate it [16, 14, 19]. In VLAs, the analogous unit of analysis is the observation–action trajectory rather than a text sequence. We adapt the interpretability criterion accordingly: a feature is *interpretable* when its temporal activation pattern, both within and across episodes, consistently aligns with an identifiable sensorimotor event or visual state.

We identify two functionally distinct categories of interpretable features that emerge from SAE training on VLA residual streams:

General features. A feature is *general* if it activates across a diverse set of episodes in response to a semantically coherent event. For example, grasping an object, placing it at a goal location, or a specific object class entering the field of view. General features exhibit three characteristic properties in their activation statistics: (i) *high mean onset count* ($\bar{o} \gg 1$), indicating bursty, event-locked firing rather than sustained activation; (ii) *broad episode coverage* (c), indicating the triggering event occurs across many episodes; and (iii) *high mean active fraction* ($\bar{\phi}$), indicating the feature activates on a large portion of timesteps within the dataset. Empirically, general features tend to exhibit $\bar{o} > 1.0$ and episode coverage ranging from 0.2 to 0.9, depending on dataset diversity.

These features are practically significant because their decoder weight vectors serve as effective steering directions: adding the vector to the residual stream at inference time reproducibly modulates the corresponding behavior across episodes (Section 4).

Memorized features. A feature is *memorized* if it is tuned to a specific episode, visual scene, or narrow task variant. Memorized features are characterized by: (i) *low mean onset count* ($\bar{o} \approx 0$–1), indicating sustained rather than bursty activation; (ii) *low episode coverage* (c), often concentrated in a single episode or a small cluster sharing the same scene; and (iii) a steep drop-off in max activation magnitude between the top episode and subsequent episodes. Although narrowly scoped, memorized features remain interpretable.

We observe that these features lie on a spectrum and do not always fall neatly into the binary classification of *memorized* or *general*. For example, we find that some features primarily memorize a visual scene yet also encode task-level affordances (e.g., DROID F322 activates most strongly on chess-piece grasping in a specific office scene, but also fires on chess boards in unrelated scenes at reduced magnitude). Conversely, some features exhibit general semantics but are classified as memorized by our metrics due to dataset composition effects (Section 6).

3.3.2 Manually Labeling Features

We manually label a subset of features and use these to train a automatic classifier. We label these features through a three-stage visual inspection pipeline that examines activation patterns at the episode, cross-episode, and dataset levels.

Stage 1: Episode-Level Screening. Starting from a randomly selected episode, we examine a feature activation heatmap that displays the top 50–100 features at a given layer. Each row of the heatmap shows the activation magnitude of a single feature across all timesteps, aligned with the corresponding camera frames, as shown in Figure 3. We refer to this visualization as the *Activation Viewer*. We identify candidate general features by their *bursty* activation signature, meaning short, high-magnitude peaks coinciding with identifiable behavioral events (e.g. grasps, placements, object appearances). We identify candidate memorized features by their *sustained* signature, meaning near-uniform activation across most timesteps without alignment to behavioral transitions.

Stage 2: Cross-Episode Validation. For each candidate, we query a precomputed index over all SAE activations in the dataset, which we call the *Feature Search* index. This returns the globally top-activating timesteps and the top-10 diverse episodes ranked by maximum activation. General

features should exhibit peak activations spanning multiple episodes and tasks with consistent, even alignment; memorized features should show activations concentrated in one or a few episodes, with a steep drop-off in magnitude. We additionally verify that this new episode subset exhibits the same per-episode temporal pattern in the Activation Viewer (bursty vs. sustained), ensuring consistency with the original randomly selected episode.

Stage 3: Labeling Criteria. A feature receives the label *general* if: (1) activation bursts align with semantically consistent events across the top diverse episodes, (2) the activation pattern reproduces in held-out episodes viewed in the *Activation Viewer*, and (3) the global per-feature metrics (mean onset count, mean active activation, episode coverage) indicate this feature is persistent across a large portion of the training dataset. A feature receives the label *memorized* if: (1) activation is sustained within the top episode(s), (2) the top episodes cluster around ≤ 2 visual scenes, and (3) the global per-feature metrics indicate this feature activates on a small subset of the training dataset. If there is any ambiguity in the interpretation of the Feature Search results, then the given feature is excluded.

3.3.3 Automated Classification

Manual labeling is precise but does not scale to thousands of features across multiple models and layers. To classify features at scale, we fit a logistic regression classifier on the four temporal metrics defined in Section 3.2, using 30 manually-labeled features (15 general, 15 memorized) from a single reference layer as training data. The classifier takes the form:

$$P(\text{general} \mid \mathbf{m}) = \sigma\big(\beta_0 + \beta_1 \bar{o} + \beta_2 c + \beta_3 \bar{a} + \beta_4 \bar{\ell}_r\big)\,, \tag{11}$$

where $\bar{o}$ is mean onset count, c is episode coverage, $\bar{a}$ is mean activation magnitude, $\bar{\ell}_r$ is relative run length, and σ is the logistic sigmoid. The classifier operates on unnormalized metric values, so the learned decision boundary can be applied across layers within the same model without per-layer normalization. We train one classifier per fine-tuning dataset (LIBERO and DROID) and apply it to all models trained on that dataset; the OpenVLA classifier reuses the LIBERO boundary since OpenVLA is also fine-tuned on LIBERO.

3.4 Feature Steering

In order to further validate our interpretation of a specific feature we can directly steer that feature and observe the model output. Given a trained SAE with encoder f_{enc} and decoder f_{dec}, we extract the steering vector for feature i as the i-th column of the decoder weight matrix:

$$\mathbf{v}_i = \mathbf{W}_{\text{dec}}[:, i] \in \mathbb{R}^d, \tag{12}$$

where d is the residual stream dimension (2048 for PaliGemma layers, 1024 for action expert layers, 4096 for OpenVLA). The decoder columns are unit-normalized by construction, so the steering magnitude is controlled entirely by a scalar coefficient α such that

$$\mathbf{y}' = \mathbf{y} + \alpha \cdot \mathbf{v}. \tag{13}$$

The steering vector is broadcast over the sequence dimensions, applying the same perturbation to every token position at each denoising step. This is another area where our approach differs from standard LLM steering approaches [19], since our model does not autoregressively predict actions.

4 Results

We present results demonstrating that the SAE features extracted using our method are **interpretable**, **general**, and **steerable** across multiple models and datasets.

We organize our results as follows. We begin by estimating the overall fraction of interpretable features across several models and layers. Next, we describe the metrics used to distinguish general from memorized features and report model-wide classification results. We then highlight specific

general features identified in both LIBERO and DROID, presenting activation visualizations and feature search results that demonstrate their activations across episodes. We then present closed-loop steering experiments on LIBERO, demonstrating that individual PaliGemma-layer features causally and predictably influence robot behavior.

4.1 Interpretable Features

We find that the vast majority of the SAE features produced by our primary model show strong interpretability. To estimate the number of interpretable features within SAE dictionaries, we take a sampling-based approach. We take a random sample of 20 features from Pi05 LIBERO PG5 and PG11, Pi05 DROID PG5 and PG11, and OpenVLA Layers 8 and 24. Here, PG refers to PaliGemma, the base VLM in Pi05, with the number indicating the corresponding transformer layer (e.g., PG5 denotes PaliGemma layer 5), whereas AE refers to the action expert. Notably interpretability is distinct from generality and has a broader scope. For each of these features interpretability is hand labeled according to the following criteria: a feature is *interpretable* when its temporal activation pattern, both within and across episodes, consistently aligns with an identifiable sensorimotor event or visual state. We show the general statistics in Table 1. Furthermore, we include detailed explanation of 10 features from LIBERO on the Interactive Feature Browser on our website under tag "labeled".

Model	Dataset	Layer	n sampled	# interp.	# non-interp.	% interp.
Pi0.5	LIBERO	PG5	20	18	2	90%
Pi0.5	LIBERO	PG11	20	16	4	80%
Pi0.5	DROID	PG5	20	17	3	85%
Pi0.5	DROID	PG11	20	14	6	70%
OpenVLA	LIBERO GOAL	Layer 8	20	16	4*	80%
OpenVLA	LIBERO GOAL	Layer 24	20	14	6	70%
Total			**120**	**95**	**25**	**79.17%**

$*$ Two of the four non-interpretable features were inactive.

Table 1: **Sampling-based estimate of SAE feature interpretability.** We estimate the fraction of interpretable features in each SAE dictionary using a sampling approach: for each listed model/dataset/layer, we evaluate a sample of $n = 20$ features (taken as a random subset of all features) and label each as interpretable or not according to our interpretability criteria (Section 3.3.3).

4.2 Model-Wide Classification

We apply the per-dataset classifiers described in Section 3.3.3 to every feature across all layers of each model. Table 2 summarizes the results.

LIBERO classifier. The fitted coefficients are $\beta_0 = -4.20$, $\beta_1 = 1.89$, $\beta_2 = 1.80$, $\beta_3 = 0.52$, $\beta_4 = -0.36$, achieving 100% leave-one-out cross-validation (LOO-CV) accuracy on the 30 labeled examples. Mean onset count and episode coverage contribute nearly equally ($\beta_1 \approx \beta_2$), indicating that in LIBERO, where many tasks share common scenes, both bursty activation and broad episode presence are required to distinguish general from memorized features. The negative coefficient on relative run length ($\beta_4 < 0$) confirms that sustained, long-duration activations are characteristic of memorized features. Because the OpenVLA model is also fine-tuned on the LIBERO dataset, we apply this same classifier to its feature dictionaries.

DROID classifier. The fitted coefficients are $\beta_0 = -1.78$, $\beta_1 = 0.74$, $\beta_2 = 2.36$, $\beta_3 = 0.35$, $\beta_4 = -1.04$, achieving 96.7% LOO-CV accuracy. In contrast to LIBERO, episode coverage dominates (β_2 is $3.2\times$ larger than β_1), reflecting the substantially greater scene and task diversity in DROID (1545 unique tasks vs. 130 in LIBERO): high episode coverage is a much stronger signal of generality when the dataset itself is diverse. The stronger negative weight on relative run length

($\beta_4 = -1.04$ vs. -0.36 in LIBERO) further penalizes features with sustained, episode-spanning activation patterns. This is consistent with the dataset characteristics: DROID episodes are substantially longer and more variable ($\mu = 283.5$, $\sigma = 219.2$ timesteps, CV $= 0.77$) than LIBERO episodes ($\mu = 161.5$, $\sigma = 68.2$, CV $= 0.42$), making relative run length a stronger differentiator between transient general features and sustained memorized ones.

Model	Layer(s)	# Features	# General	# Memorized	% Memorized
Pi0.5 – LIBERO					
	PG5	2044	32	2012	98.43%
	PG avg (0, 5, 11, 17)	7175	188	6987	97.37%
Pi0.5 – DROID					
	PG5	2046	104	1942	94.92%
	PG avg (0, 5, 11, 17)	6649	719	5930	89.19%
OpenVLA – LIBERO Goal					
	Layer 8	1775	8	1767	99.55%
	LM avg (0, 8, 16, 24, 31)	9389	42	9347	99.55%

Table 2: **Feature classification results across models and layers.** We report classifier performance for distinguishing general from memorized SAE features using activation-pattern metrics. Individual layers (PG5, Layer 8) serve as baselines; averages are computed across all layers within each subnetwork.

Overall we find that only a small fraction of features are general across these models. For LIBERO studies [10] have shown that VLA models can strongly visually overfit to scenes. We find that models fine-tuned on DROID exhibit a relatively lower portion of memorized features than those fine-tuned on LIBERO, but the absolute portion of memorized features remains high. Although DROID is large by robotics standards, it is still relatively small compared to the scale of language model training datasets, with only intermediate task and scene diversity. We feel the best way to interpret these results is to compare the relative portion of memorized features between models and datasets. In this we find a clear and definitive pattern, increasing the size and diversity of the fine-tuning dataset increases the portion of general features. In particular, moving from LIBERO-GOAL to full LIBERO and then to DROID, we observe a steady increase in the share of general features. Please see Section A.5.1 for a discussion of common misclassifications and Section 4.5 for the impact of training steps and knowledge insulation on the proportion of general features.

4.3 General Features

4.3.1 Highlighted General Features LIBERO

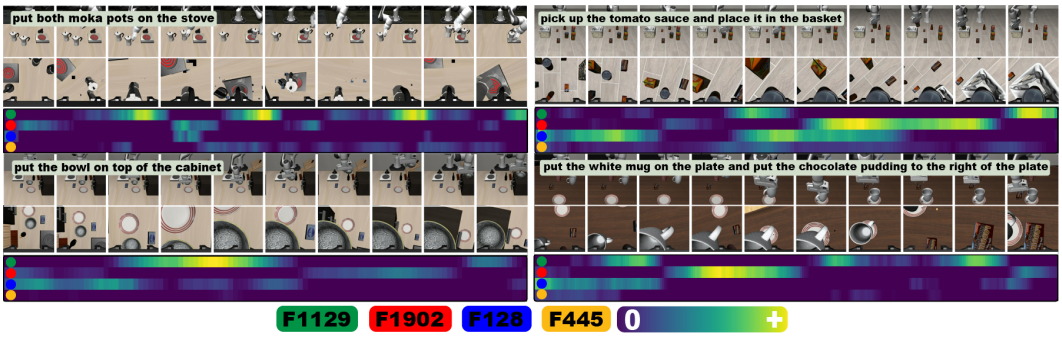


Figure 3: We present 4 general features identified across a range of LIBERO episodes. For each task, we show the wrist and main camera images equally spaced between t_0 and t_f. Below these images, we show the activations of four general features over the episode. The feature activations and camera images are in chronological order.

All four features below have episode coverage > 0.99 and are invariant to the scene, the goal object, and the number of grasps. These features represent some of the most general features in our dataset and were classified as follows: $P(\text{general} \mid \text{F1129}) = 0.91$, $P(\text{general} \mid \text{F1902}) = 0.89$, $P(\text{general} \mid \text{F128}) = 0.92$, and $P(\text{general} \mid \text{F445}) = 0.58$. We also show the top 10 activating episodes for each of these features on our website.

1. **F1129 (grasp/place):** Activates during initial object grasp and placement, with onset count scaling with the number of pick-and-place sub-goals (two onsets for single-object tasks, four for two-object tasks).
2. **F1902 (transport):** Activates between the onset pairs of F1129, corresponding to the carrying phase. Activation magnitude increases approximately linearly toward the goal position, suggesting the feature also encodes proximity to the placement target.
3. **F128 (pre-grasp alignment):** Activates when the end-effector is positioned above the target object prior to each F1129 onset. Magnitude ramps as the object enters and centers in the wrist camera frame. Also reactivates post-placement if the arm returns to a vertically aligned position.
4. **F445 (task completion):** Activates when the end-effector approaches the goal placement location, predominantly on the final sub-goal. In compound tasks, it activates at a lower magnitude and preferentially on the second object placement, consistent with encoding overall task success rather than sub-task completion

4.3.2 Highlighted General Features DROID

These features show high episode coverage despite DROID's substantially greater scene and task diversity (1545 unique tasks vs. 130 in LIBERO). Mean activations are lower than their LIBERO counterparts, consistent with encoding more abstract manipulation concepts. These features were classified as follows: $P(\text{general} \mid \text{F158}) = 0.91$, $P(\text{general} \mid \text{F586}) = 0.78$, $P(\text{general} \mid \text{F165}) = 0.83$, and $P(\text{general} \mid \text{F399}) = 0.79$.

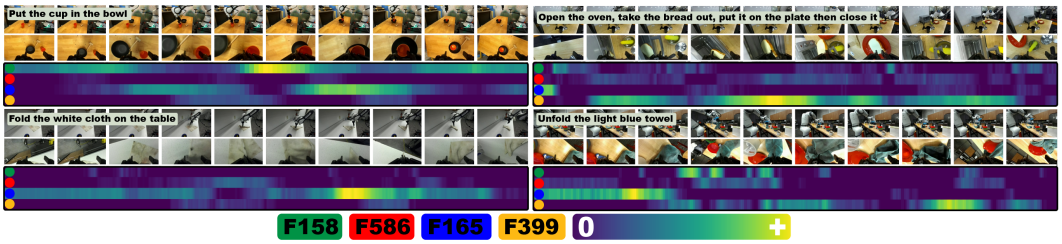


Figure 4: DROID general features across diverse tasks. For each episode, we show evenly spaced frames from the main and wrist cameras from t_0 to t_f, with per-timestep feature activations plotted below. Rows correspond to example tasks, and the color bars map to features F158, F586, F165, and F399 (legend at bottom).

1. **F158 (sub-task checkpoint):** The highest-$\bar{o}$ feature across all LIBERO and DROID PG5 SAEs. Activates at transitions between manipulation sub-phases: approaching the grasp position, acquiring the object, and approaching the goal. In episode 472 (8-bottle pick-and-place), the feature produces 16 distinct onsets, each aligned with a bottle entering the wrist camera or the goal plate becoming visible during grasp.
2. **F586 (pinch grasp):** Activates during precision grasps of thin objects (plate edges, sugar packets, utensil handles, towels) and persists through the grasp duration. Activates at reduced magnitude ($\sim 0.5\times$) for wider grasps (cans, blocks), consistent with encoding grip aperture.
3. **F165 (open gripper over target):** Activates when the target object is visible between the open gripper jaws in the wrist camera. Onsets typically follow F158 activations. Spurious

early-episode activations occur when the gripper passes over scene objects before the task begins.

4. **F399 (grasp acquisition/placement):** Activates during the closing phase of a grasp and during object placement; deactivates during transport. Activation peaks overlap with F586 in episodes involving pinch grasps. Functionally analogous to LIBERO F1129 but with noisier temporal boundaries.

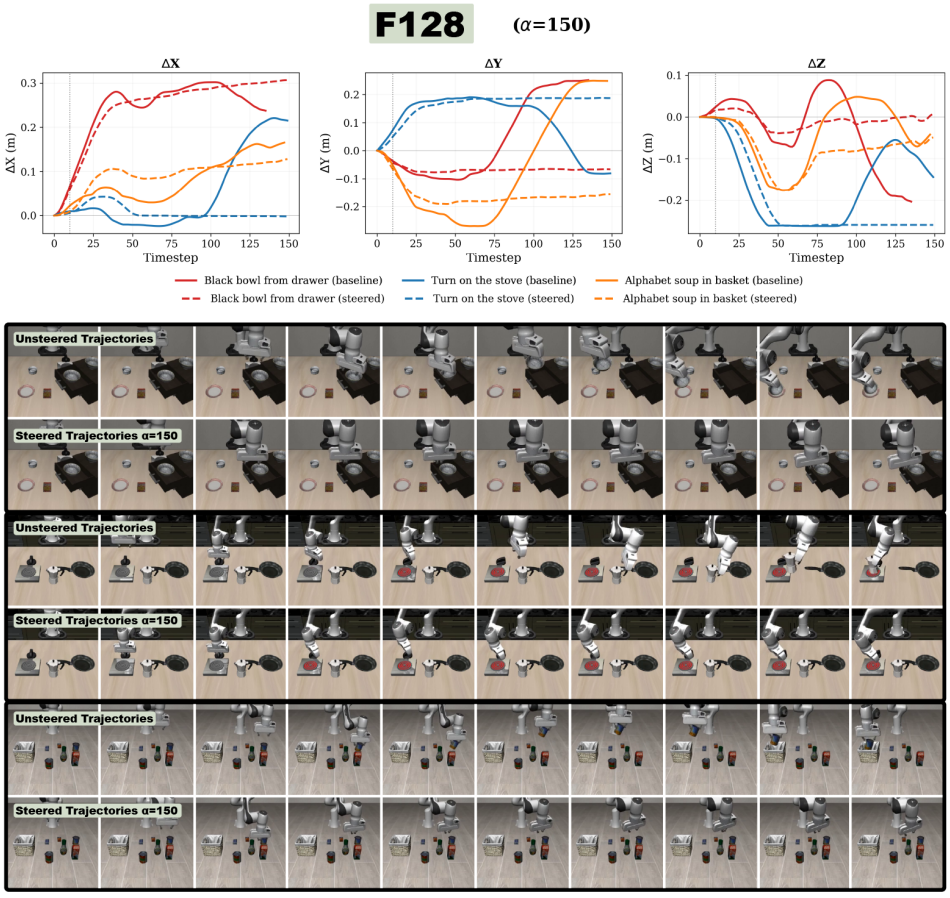


Figure 5: Closed-loop steering results for F128 of Pi05 LIBERO. We show XYZ trajectory plots of the unsteered and steered trajectories (top) and images across timesteps during steering (bottom)

4.4 Feature Steering

In this section, we apply two general features, F128 and F1902, to steer closed-loop behavior across several episodes in LIBERO. One observation that holds throughout our experiments is the impressive robustness of VLA models to activation-level perturbations. Even when steering with a single feature at high magnitude, the policy continues to exhibit goal-directed behavior, generally attempting to approach relevant objects, grasp, or pursue alternative subtasks rather than simply producing degenerate or random motion.

For these closed-loop evaluations, steering is applied starting at the 10th timestep and continues throughout the episode. We execute five steps from the Pi0.5 action horizon before updating the observation and inferencing the model. Additional details can be found in Section A.6.

F128 (LIBERO PG5) As discussed in Sec. 4.3.1, F128 correlates strongly with pre-grasp alignment, activating when the end-effector is positioned above the target object prior to grasp. We perform steering across three tasks: *move the black bowl from the drawer to the plate*, *turn on the*

stove and put the mocha pot on the stove, and *pick up the alphabet soup and place it in the basket*. When we steer with this feature, we observe that the robot consistently approaches the target grasp object and, rather than picking it up, simply hovers above it. This behavior is fully consistent with our expected interpretation of the feature as a pre-grasp alignment representation.

F1902 (LIBERO PG5) As discussed in Sec. 4.3.1, F1902 is a feature relating to the transport of the object to the goal region. We find that this feature activates near the goal position and encodes proximity to the placement target. We steer this feature across three tasks: *move the black bowl from the drawer to the plate*, *put the white mug on the plate*, and *pick up the alphabet soup and place it in the basket*. In all three of these examples, the robot completely skips the relevant object grasps and moves directly to the goal location. In the *pick up the alphabet soup and place it in the basket* trajectory, the robot even collides with the basket, pushing it out of the way. Videos of these results are shown on our website.

Overall, we find that the steering results for these features reinforce our findings in Sec. 4.3.1, based on the feature search and activation viewer investigations. However, steering is not perfect and still shows some evidence of superpositional effects with a given feature. We also preform steering on memorized features. We find that memorized steering has varying effects, sometimes driving the robot towards a specific direction, while other times seemingly regurgitating memorized behaviors. We show memorized steering results on our website.

4.5 Impact of Training Steps and Knowledge Insulation

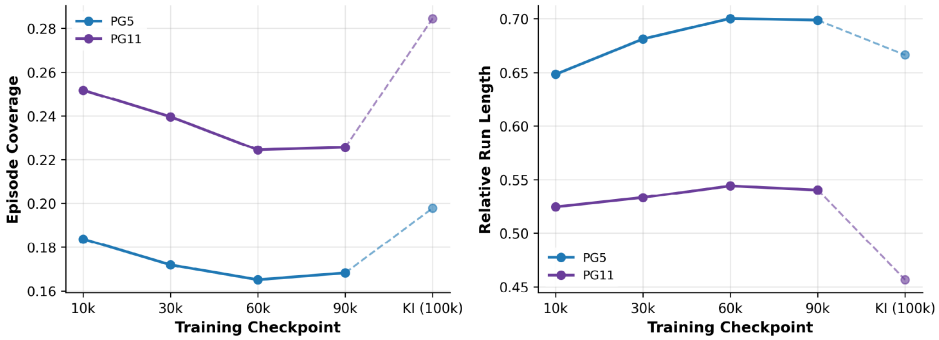


Figure 6: Two feature metrics across several training checkpoints on Pi05 DROID with full-parameter AE and Knowledge Insulation (KI) finetuning. Dotted lines indicate separate model initializations. Episode converge decreases with additional steps, while relative run length increases.

Here, we ask whether the feature metrics we propose can serve as a proxy for model generality during training. Our results, shown in Figure 6, weakly suggests this claim. As expected, generality decreases as fine-tuning proceeds. Although the Pi05 base is trained on over 10,000 hours of cross-embodiment robot data, catastrophic forgetting appears to begin once fine-tuning begins. In contrast, Knowledge Insulation (KI), which is designed to preserve general capabilities, appears to reverse this decline.

In Figure 6, we report episode coverage and relative run length across training checkpoints. As discussed in Section 3.3, higher episode coverage typically correlates with more general features, and we also find that lower relative run length correlates with greater generality. Episode coverage decreases while relative run length increases with training steps, yet the trends are reversed under KI, consistent with prior evidence that KI improves generalization [26].

This conclusion is further supported by the the distribution of $P(\text{general})$ from our regression-based generality classifier across all features for each model on DROID PG5. The KI model shifts the distribution towards generality with higher median than AE-finetuned models at comparable training steps. The median classifier outputs are 0.190 (10k), 0.181 (30k), 0.179 (60k), 0.181 (90k) and 0.206 (KI). Although these shifts are subtle, they suggest that feature-level metrics could provide a signal

of generalization during training without requiring real-world rollouts. However, a more rigorous evaluation is needed to confirm this.

5 Conclusion

In this work, we have introduced an SAE-based mechanistic interpretability pipeline for Vision Language Action (VLA) models. We train sparse autoencoders on residual stream activations to obtain sparse, human-interpretable features. Across two VLA architectures and two robotics datasets, we find that many SAE features exhibit clear interpretations as motion primitives, task-progress signals, semantic representations, and episode-specific memorization. Further sampling-based analysis shows that a large fraction of learned features are interpretable across both models and layers.

Beyond our interpretability results, we provide evidence that individual SAE directions causally influence closed-loop behavior. When we steer general features, we often observe that they modify the motion consistently with their hypothesized interpretation. For example, steering a pre-grasp alignment feature causes the policy to hover above objects instead of continue grasping. These interventions are complemented by our feature-search and activation-heatmap analyses. Together, this provides strong evidence that SAE features capture behaviorally meaningful computations rather than purely correlational artifacts. We introduce feature metrics (episode coverage, onset count, activation magnitude, and run length) to empirically quantify generality and support scalable feature categorization. Using these metrics, we observe that feature generality varies with different training choices. As we increase the number of fine-tuning steps, generality decreases, whereas knowledge insulation appears to mitigate this degradation. Likewise, increasingly large and more general datasets, like DROID, also seem to improve feature generality.

While in this paper, these results are not applied toward model improvement, our results suggest several immediate directions to make these features practically useful. One potential application of these results is during model training. Because SAEs do not require any rollouts, simulated or otherwise, they can be applied in a lightweight fashion during training. For example, the presence of episode-specific features can be used to diagnose the brittleness of common VLA fine-tuning procedures. Similarly, our feature metrics may serve as a training-time proxy for generalization without requiring real-world evaluation, which can be time-consuming and expensive. Using this information, future work can be done to explore techniques such as varying dataset mixing and fine-tuning schedules to preserve transferable behavior. At test time, these features can be used within a runtime monitor, which alerts users when a policy is leveraging a memorized trajectory rather than utilizing general features. To enhance the policy, users can explicitly encourage or ablate specific SAE features during fine-tuning or in a reinforcement learning setting to adjust which representations the model relies on.

6 Limitations

We find that meaningful top activations of a feature does not imply reliable steerability. Specifically, we find that many *clean* features exhibit limited or unpredictable causal impact when used to steer. We hypothesize that nonlinear downstream interactions (since the VLA produces actions via flow matching) or the fact that being predictive does not imply causality may inhibit steerability. Furthermore, we find that some features do not fit into the proposed general-versus-memorized feature classification. For example some features appear highly general, activating across many different episode types, yet do so only at the end or beginning of an episode. Other features appear to be general yet activate in such a small portion of the dataset that they are classified as memorized. More details can be found in Section A.5.1. Additionally, compared to VLM research, we use orders of magnitude less data due to data constraints specific to robotics. In our experiments, due to storage and computational constraints, we primarily used mean-pooled tokens, evaluating our SAEs on a per-timestep basis. We include some initial per-token experiments (Section A.7), which are promising but generally less interpretable and warrant further study. Finally, our evaluation scope remains

somewhat limited. While we demonstrate steering results in closed-loop simulation rollouts, we do not evaluate on hardware. The lack of high-success-rate general VLA policies that can be reliably deployed on real hardware may further inhibit steerability.

Acknowledgments

We would like to thank Timothy Chen for reading preliminary versions of this paper and providing feedback. We utilized the Stanford Marlowe cluster for several experiments [30]. Aiden Swann is supported by NSF GRFP Fellowship No. DGE-2146755.

References

[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

[3] 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. URL https://arxiv.org/abs/2410.24164.

[4] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[5] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[6] L. Floridi and M. Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and machines*, 30(4):681–694, 2020.

[7] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[8] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[9] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

[10] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

[11] M. Lan, P. Torr, A. Meek, D. Krueger, and F. Barez. Sparse autoencoders reveal universal feature spaces across large language models. 2024.

[12] L. Gao, T. D. la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, and J. Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

[13] J. Kramár, J. Engels, Z. Wang, B. Chughtai, R. Shah, N. Nanda, and A. Conmy. Building production-ready probes for gemini. *arXiv preprint arXiv:2601.11516*, 2026.

[14] H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL https://arxiv.org/abs/2309.08600.

[15] H. Zhang, Z. Zhang, M. Wang, Z. Su, Y. Wang, Q. Wang, S. Yuan, E. Nie, X. Duan, Q. Xue, Z. Yu, C. Shang, X. Liang, J. Xiong, H. Shen, C. Tao, Z. Liu, S. Jin, Z. Xi, D. Zhang, S. Ananiadou, T. Gui, R. Xie, H. K.-H. So, H. Schütze, X. Huang, Q. Zhang, and N. Wong. Locate, steer, and improve: A practical survey of actionable mechanistic interpretability in large language models, 2026. URL https://arxiv.org/abs/2601.14004.

[16] T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features#feature-analysis. Accessed 2026-01-27.

[17] M. Pach, S. Karthik, Q. Bouniot, S. Belongie, and Z. Akata. Sparse autoencoders learn monosemantic features in vision-language models, 2025. URL https://arxiv.org/abs/2504.02821.

[18] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, and C. Olah. Toy models of superposition, 2022. URL https://arxiv.org/abs/2209.10652.

[19] A. Templeton, T. Conerly, J. Marcus, et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/.

[20] H. Lu, H. Li, P. S. Shahani, S. Herbers, and M. Scheutz. Probing a vision-language-action model for symbolic states and integration into a cognitive architecture. *arXiv preprint arXiv:2502.04558*, 2025.

[21] M. Molinari, L. Nevali, S. Navani, and O. G. Younis. Emergent world representations in openvla. *arXiv preprint arXiv:2509.24559*, 2025.

[22] B. Häon, K. Stocking, I. Chuang, and C. Tomlin. Mechanistic interpretability for steering vision-language-action models, 2025. URL https://arxiv.org/abs/2509.00328.

[23] M. A. Khan, N. Boskov, F. M. Anwar, and M. A. Khan. Controlling vision–language–action policies through sparse latent directions. In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=wtf3ww1EOL.

[24] H. Buurmeijer, C. A. Alonso, A. Swann, and M. Pavone. Observing and controlling features in vision-language-action models, 2026. URL https://arxiv.org/abs/2603.05487.

[25] N. Kachaev, M. Kolosov, D. Zelezetsky, A. K. Kovalev, and A. I. Panov. Don't blind your vla: Aligning visual representations for ood generalization. *arXiv preprint arXiv:2510.25616*, 2025.

[26] 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. URL https://arxiv.org/abs/2505.23705.

[27] S. Grover, A. Gopalkrishnan, B. Ai, H. I. Christensen, H. Su, and X. Li. Enhancing generalization in vision-language-action models by preserving pretrained representations. *arXiv preprint arXiv:2509.11417*, 2025.

[28] A. Makhzani and B. Frey. Winner-take-all autoencoders, 2015. URL https://arxiv.org/abs/1409.2752.

[29] P. 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. URL https://arxiv.org/abs/2504.16054.

[30] C. Kapfer, K. Stine, B. Narasimhan, C. Mentzel, and E. Candes. Marlowe: Stanford's gpu-based computational instrument, Jan. 2025. URL https://doi.org/10.5281/zenodo.14751899.

[31] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[32] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

A Appendix

A.1 Model and Datasets

A.1.1 Models

We perform our analysis on three VLA models spanning two architectures and robot embodiments. Our primary model is Pi0.5 [29], a state-of-the-art VLA that couples a PaliGemma [31] vision-language backbone (Gemma 2B [32], 18 transformer layers, $d = 2048$, 400M SigLIP image encoder) with a dedicated action expert (300M, 18 transformer layers, $d = 1024$). The PaliGemma (PG) backbone processes three camera images (base, left wrist, right wrist); however, for our cases, we use only two input images, along with a natural-language instruction. The action expert (AE) decodes these representations into continuous robot actions via iterative denoising, attending to the PaliGemma key-value cache. We use two fine-tuned variants open-sourced by Pi: Pi0.5-LIBERO and Pi0.5-DROID, both fine-tuned on the respective datasets with a technique known as knowledge insulation [26]. We choose both because LIBERO provides a controlled simulation environment for closed-loop evaluations, while DROID offers some real-world diversity, enabling us to identify general semantic features.

As a secondary model, we study OpenVLA [2], an open-source VLA built on a Llama 2 7B backbone (32 transformer layers, $d = 4096$). We use the publicly available checkpoint fine-tuned on the LIBERO Spatial suite (`openvla/openvla-7b-finetuned-libero-spatial`). OpenVLA uses a single language model for both perception and action prediction, predicting actions autoregressively. We include OpenVLA primarily to test whether the features and phenomena we observe in Pi0.5 generalize across VLA architectures with fundamentally different action decoding mechanisms.

A.1.2 Datasets

LIBERO [8] is a simulated benchmark covering a limited set of tabletop tasks built on the Robosuite simulator. We collect activations from the full training set provided via the HuggingFace repository (`physical-intelligence/libero`) used by the Pi0.5 training pipeline. This comprises **1,693 episodes** spanning **40 unique tasks** across four suites: LIBERO-Spatial (10 tasks, 432 episodes), LIBERO-Object (10 tasks, 454 episodes), LIBERO-Goal (10 tasks, 428 episodes), and LIBERO-10 (10 tasks, 379 episodes). The dataset contains **273,465 total timesteps** with 7-dimensional actions (6-DoF end-effector + gripper). Notably, the Spatial and Object suites each share a single base scene across all 10 tasks therefore the dataset contains fewer visual environments ($\sim$20) than unique task descriptions, and considerable similarities between episodes as noted in [10].

DROID [5] is a large-scale, in-the-wild robot manipulation dataset collected across diverse real-world environments on Franka Panda robots. We collect activations on a randomly sampled subset of **2,000 episodes** (1,750 successful, 250 failed) drawn from the RLDS-formatted DROID corpus. We intentionally consider failed episodes, as they represent a traditional underexamined form of robotic data. This subset spans **1,545 unique task instructions** and **567,088 total timesteps** with 8-dimensional actions (7 joint positions + gripper). Note that Pi0.5-DROID was trained by Pi on the full DROID dataset ($\sim$75k episodes); our 2,000-episode subset is used solely for activation collection and analysis.

For OpenVLA analysis, we collect activations only from the LIBERO-Goal suite, matching the model's fine-tuning distribution. This subset contains **428 episodes** across **10 goal-conditioned tasks** with **52,042 total timesteps**.

A.1.3 Activation Collection

For each model, we collect residual-stream activations from the output of the full transformer block (after both the self-attention and MLP sublayers and their respective residual connections), which is the standard residual stream location used in prior work on mechanistic interpretability [12].

Dataset	Episodes	Tasks	Timesteps
LIBERO	1,693	40	273,465
DROID (2k subset)	2,000	1,545	567,088
LIBERO-Goal (OpenVLA)	428	10	52,042

Table 3: Dataset statistics for the models we analyze in this work.

Activations are captured via PyTorch `register_forward_hook` callbacks on the target decoder layer modules.

For Pi0.5, we record eight layers: PaliGemma layers 0, 5, 11, 17 ($d = 2048$) and action expert layers 0, 5, 11, 17 ($d = 1024$). PaliGemma hooks fire during the single prefill pass that processes all image and instruction tokens, while action expert hooks fire during the final denoising step of the action diffusion process. For OpenVLA, we record five layers of the Llama 2 backbone: layers 0, 8, 16, 24, and 31 ($d = 4096$), which capture the prefill activation over the full input sequence.

At each timestep, the Pi0.5 PaliGemma backbone processes a variable-length sequence of tokens: three 256-token image embeddings (one per camera, yielding 768 image tokens) plus additional instruct and state tokens. By default, we mean-pool all tokens within the hook into a single d-dimensional vector per timestep. This yields a compact representation suitable for SAE training: each episode produces a matrix of shape (T, d), where T is the number of timesteps. All SAE results in the main text use these pooled activations unless otherwise noted. We acknowledge that this differs from the per-token approach commonly taken in activation collection for LLMs. We choose pooled activations for two key reasons. Firstly, each timestep represents a single observation-action pair, making it the natural unit of analysis for robot behavior. This is exacerbated by the fact that the token sequence ($\sim$768 tokens) is dominated by image patch tokens, which individually carry limited semantic content; mean-pooling compresses these into a single representation that captures the aggregate state at each step while reducing the dataset size by two orders of magnitude. Secondly, since each timestep would involve collecting activations for over 768 tokens, which, even for a single layer, would require $\sim$3.5TB of storage across our 2000-episode DROID dataset.

Despite these challenges, we also collect per-token activations to examine finer-grained structure. To address storage issues, we sum the patches of each image into a single vector. At each timestep, this yields 2 image vectors (one per camera, each obtained by mean-pooling that camera's 256 patch tokens) and $\sim$21 individual text-token vectors, all of dimension 2048. We train per-token SAEs on PaliGemma layer 5 and on the embedding layer (layer 0) for both LIBERO and DROID, at expansion ratios ranging from $1\times$ to $8\times$. Per-token results are presented in Section A.7.

Hyperparameter	**Value**
Expansion ratio	1
Active features	100
Auxiliary k_{aux}	512
Auxiliary loss coefficient λ_{aux}	$1/32$
Learning rate η	1×10^{-4}
Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Batch size	4096
Training epochs	100
Geometric median samples	10,000

Table 4: Default SAE training hyperparameters. An expansion ratio of $ER = 1$ means the SAE hidden dimension equals the input dimension d (e.g., 2048 features for PaliGemma layers). Ablations over expansion ratio are presented in Figure 7.

A.2 SAE Training Hyperparameters

We train all SAEs using the TopK architecture with AuxK auxiliary loss described in Section 3, with default hyperparameters summarized in Table 4. A notable departure from typical VLM interpretability work is our low expansion ratio (ER = 1). As shown in Figure 7, higher expansion ratios lead to substantially more dead features, and we find that interpretable features emerge reliably even at this compact dictionary size, likely due to the much smaller-scale datasets we work with in robotics. Note that for OpenVLA, we train our SAEs with $ER = 0.5$ to match the dictionary size of 2048 features for our Pi05 models. All other parameters are left unchanged.

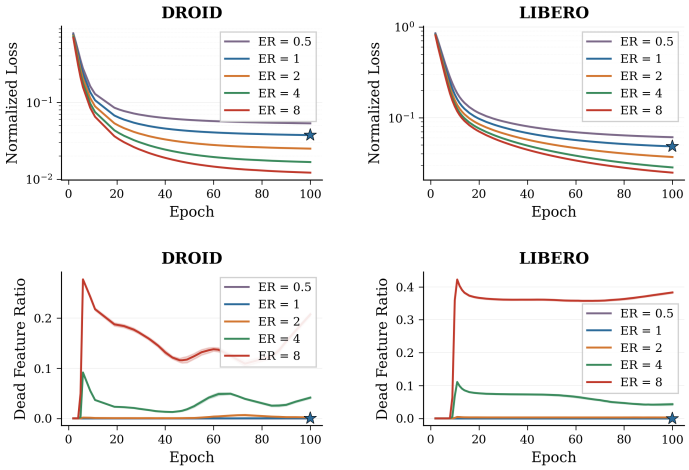


Figure 7: Ablations over SAE expansion ratio and learning rate on Pi05 PG5 activations. Top row: reconstruction loss versus epoch for DROID (left) and LIBERO (right) at each expansion ratio. Bottom row: dead feature ratio versus epoch for the same configurations. The star marks the setting used throughout this paper (ER = 1), chosen as a trade-off between low reconstruction loss and low dead feature ratio.

A.3 Multi-Seed Ablation

To verify that the features we identify reflect structure in the model's representations rather than imaginations of the SAE, we train seven SAEs from independent random seeds on the same activation data (Pi0.5 LIBERO PG5). Figure 8 shows that the top features for a given episode are consistently recovered across seeds, with similar temporal activation patterns.

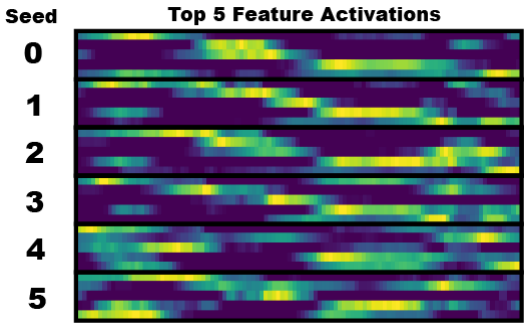


Figure 8: Activations from SAEs trained on Pi0.5 LIBERO from 6 randomly initialized seeds. We plot the top five features for episode 689; activations are normalized for visualization and ordered by consistency relative to seed 0. The shared patterns across random initializations provide evidence that the learned SAE features arise from the underlying model.

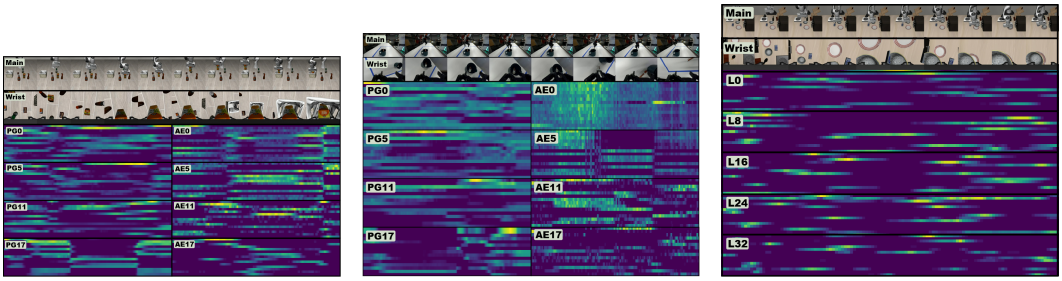
A.4 SAE Activations

A.4.1 Cross-Layer Feature Visualization

Figure 9 displays the top 15 SAE features per layer for a single episode from Pi05 LIBERO, Pi05 DROID and OpenVLA LIBERO-Object, providing a qualitative overview of how activation structure varies across layers, architectures, and datasets.

For Pi0.5, layers PG5 and PG11 contain the most salient and interpretable features. PG0, as the first transformer layer, produces activations that closely resemble the input embeddings; comparing Figure 9a PG0 to the SigLIP embedding baseline in Figure 10 confirms this similarity. In contrast to the action expert layers, these patterns more closely mirror the action expert layers than the earlier PaliGemma layers. This is consistent with Pi0.5's knowledge-insulation training, in which the VLM backbone produces action tokens in addition to the action expert's continuous actions, suggesting that the later PaliGemma layers have a similar action-encoding structure to the action expert. The action expert layers display a qualitatively distinct pattern. SAE features are activated in distinct phases of manipulation, punctuated by grasp events. We interpret these as motion primitives encoding characteristic directions of motion. This motion-primitive-like structure is consistent across both the LIBERO and DROID action expert layers (Fig. 9a and Fig. 9b), despite the two models relying on entirely different action representations (end-effector pose for LIBERO versus joint positions for DROID).

Comparing the two Pi0.5 variants, DROID activations are noticeably denser than their LIBERO counterparts. We attribute this to the substantially greater visual diversity of DROID's real-world scenes relative to LIBERO's simulated environments, which requires the SAE to recruit more features simultaneously to represent each timestep. The OpenVLA features in Fig. 9c show a qualitatively different texture from Pi0.5. As OpenVLA is just a VLM containing no action expert, its activation patterns more closely resemble the PaliGemma backbone layers than the action expert layers. Although individual features appear sparser in the heatmap, the fraction of features classified as general is in fact substantially lower than in Pi0.5 (Table 2). However, this lack of generality is to be expected given the relatively small training dataset as discussed in Section A.1.2.



(a) LIBERO episode 870: *pick up the orange juice and place it in the basket*.

(b) DROID episode 855: *Put the glass lid on the black pot*.

(c) OpenVLA episode 0: *put the bowl on the plate*.

Figure 9: Top 15 most active SAE features across layers for single episodes from LIBERO (left), DROID (middle), and OpenVLA (right).

A.4.2 Embedding Baseline

To ensure that the features we identify arise from robotic fine-tuning rather than from existing features in the pretrained vision encoders, we train SAEs on two baselines: the Pi0.5 embedding layer and the frozen pretrained SigLIP encoder that has never seen any robotics data. Figure 10 shows the top 15 features for a single episode from each. The Pi0.5 embedding activations bear some resemblance to those found in PG0, which is unsurprising given that the two are separated by only a single attention layer and MLP. In contrast, the pretrained SigLIP model produces a qualitatively different and much denser activation pattern, lacking the event-locked structure characteristic of the general

features we identify in later layers. This suggests that the interpretable features reported in our main results are not inherited from the pretrained visual representations but instead emerge during robotic fine-tuning.

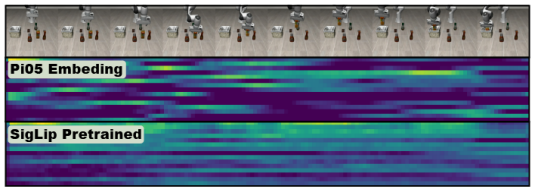


Figure 10: Top 15 SAE features for the Pi0.5 embedding layer (top) and the pretrained SigLIP encoder (bottom) for a single episode, *Pick up the orange juice and place it in the basket.*. The SigLIP encoder has not been exposed to any robotics data.

A.5 Feature Classification

A.5.1 Under-classification of Generality

These classification results likely underestimate the number of general features within a given layer, especially in high-diversity datasets such as DROID. Two examples illustrate a class of features that our metrics fail to capture: F1939 from LIBERO PG5 and F1381 from DROID PG5.

F1939 (LIBERO PG5): $\bar{o} = 1.00$, $c = 0.732$, $\bar{a} = 0.037$, $\bar{\ell}_r = 0.127$. Despite an episode coverage of 73%, the onset count of exactly 1.0 causes the classifier to label this feature as memorized. All activations occur within the first 20 timesteps of each episode regardless of scene or task, consistent with encoding the robot "home" position—a pose shared across all LIBERO environments. The feature is general by visual inspection (scene- and task-invariant, present in nearly three-quarters of episodes), but its single-onset-per-episode pattern yields an $\bar{o}$ indistinguishable from a memorized feature that fires once in a narrow set of episodes.

F1381 (DROID PG5): $\bar{o} = 1.00$, $c = 0.226$, $\bar{a} = 0.065$, $\bar{\ell}_r = 0.990$. This feature is activated during the grasp of a lid, regardless of lid type (metal pot lid, glass lid, paper cup lid, green plastic lid) and the scene. It fires in 116 of the 135 episodes whose task instructions contain the word "lid" (86% recall within that subset), and in 336 additional episodes with instructions such as "put the pot on the right side of the table and open it," which are accomplished via lid grasps. However, because lid-related episodes constitute only 6.7% of the DROID dataset, the episode coverage of 0.226 falls below the classifier's decision boundary. The near-unity relative run length ($\bar{\ell}_r = 0.990$) further penalizes it, as the feature remains active throughout the lid-grasp episodes rather than activating in brief bursts.

Both cases expose the same structural limitation: features that activate once per episode ($\bar{o} \approx 1$) across a semantically coherent but proportionally small subset of the dataset will be classified as memorized. The classifier conflates low burstiness with memorization because, in the labeled training set, general features are predominantly multi-onset. Addressing this would require either a dataset-diversity-aware normalization of episode coverage or an additional metric that captures cross-scene consistency independently of activation frequency.

A.6 Steering

We evaluate whether individual SAE features causally influence model behavior through steering. Following LLM interpretability research [19], we add our SAE feature vectors directly into the model residual stream. This provides evidence of whether a feature's learned direction carries meaning beyond simple activation correlations.

A.6.1 Closed-Loop Evaluation

We evaluate steering effects closed loop in LIBERO [8]. For Pi0.5, we predict an action trajectory over 50 future timesteps in a single forward pass via 10 iterative denoising steps. The steering hook is active throughout all 10 denoising iterations, so the perturbation influences the entire refinement process, from noise to the action trajectory. Of the 50 predicted timesteps, only the first 5 are executed before the model is re-queried. In contrast, OpenVLA predicts a single action per forward pass in an autoregressive manner. The model is re-queried at every environment step with the current observation.

A.6.2 Additional Steering Experiments

We examine how the steered effect varies with layer depth and type in the Pi05 model. We steer using SAE features from each of the four Paligemma and action expert layers and present the results in Figure 11. Here we find that steering produces much larger end-effector displacements in early action expert layers than in later layers or PaliGemma layers. In both LIBERO and DROID, AE0 is the most steerable, but the effect curtails sharply with depth. Paligemma layers are consistently weaker at the same depth, and PG L17 produces exactly zero displacement in both models.

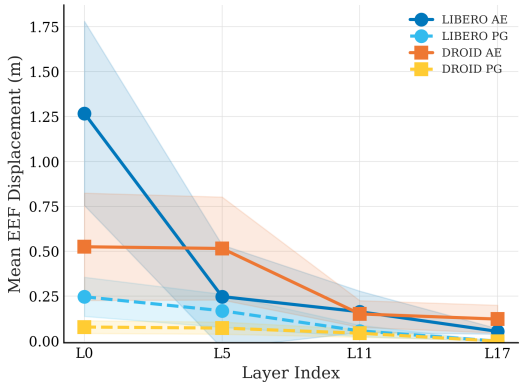


Figure 11: Mean end-effector (EEF) displacement induced by steering with random SAE decoder directions across Action Expert (AE, solid) and PaliGemma (PG, dashed) layers for LIBERO and DROID Pi0.5 models. For each of the 8 layers, 10 random features were sampled and steered at α=300 across 5 randomly selected episodes, yielding 50 measurements per layer. Shaded regions denote one standard deviation. PG L17 produces zero effect in both models.

A.7 Per-Token SAEs

As discussed in Section A.1.3, we also collect per-token activations. While our summed-token activations reveal several exciting, broadly general features, probing visual–textual semantic alignment requires token-level resolution. To make the problem more tractable, we aggregate all image inputs into a single "image" token, while we collect individual activations for each "text" token. We train a single SAE to reconstruct all of these activations. We find that while per-token SAEs provide more insight into the model, more investigation and likely more training data are needed to fully evaluate them. Overall, per-token SAEs are promising but currently less interpretable than their summed-token counterparts. Figure 12 shows our results. We discuss two promising features below.

F1881 is prominent across all episodes but lacks human interpretability. In this episode, it is strongly and nearly uniformly activated on the main camera token and more sparsely, at a lower magnitude, on the wrist camera. All highlighted text tokens exhibit non-zero activation, consistent with a general visual-semantic alignment role. The noun "pot" and its modifier "black" display nearly identical activation patterns, suggesting this feature links co-referent tokens within a scene. However, the temporal structure of these activations is not clearly interpretable.

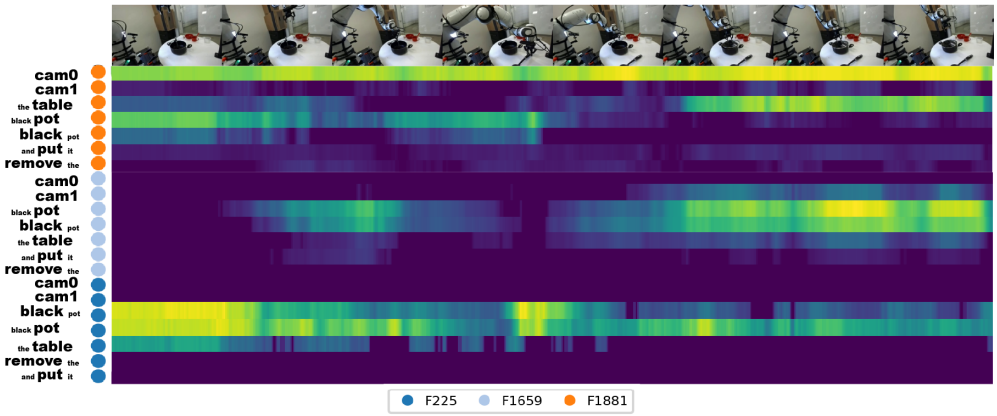


Figure 12: Per token activations for three features: F225 F1659 and F1881 for Pi05 DROID PG5. These activations are collected for episode 90 *Remove the black ladle from the black pot and put it on the table, put the lid on the black pot*. We show the SAE activations for 7 tokens: cam0, cam1, table, pot, black, put, and remove. Additional tokens are shown only for context.

F225 is similarly prominent across episodes but appears to capture purely textual semantic structure. It has zero activation on both camera tokens but strong activation on the nouns in the task instruction. It does not activate on verbs, a pattern consistent across episodes. The tokens "black" and "pot" again co-activate, peaking when the wrist camera is positioned over the pot.

F1659 is an interpretable, memorized feature: its top camera-token activations come exclusively from episodes containing lids and pots, peaking when the lid is placed on the pot. Its top text-token activations correspond to "pot" or the adjacent space when "pot" is the final text-token. Within this episode, "pot" is the highest activating token, again co-activated with "black" and the wrist camera. The remaining tokens show qualitatively similar but substantially weaker activation. Analogous features in other episodes exhibit strong visual-textual alignment while remaining task-specific.

Compared with the summed-token SAEs, these per-token SAEs are substantially less interpretable in the general case and require further investigation to fully characterize their visual-semantic alignment properties. Evidence of generality exists but is concentrated in fewer than ten features per layer, and the top-activating text-tokens of these general features are often function words ("the," "and") rather than semantically informative content words. Memorized features retain greater interpretability and exhibit stronger visual-textual alignment, though they also exhibit inconsistencies.