MULTI-DIMENSIONAL FEATURIZERS YIELD FINE-GRAINED STEERING AND DATA SHAPING IN ROBOT FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

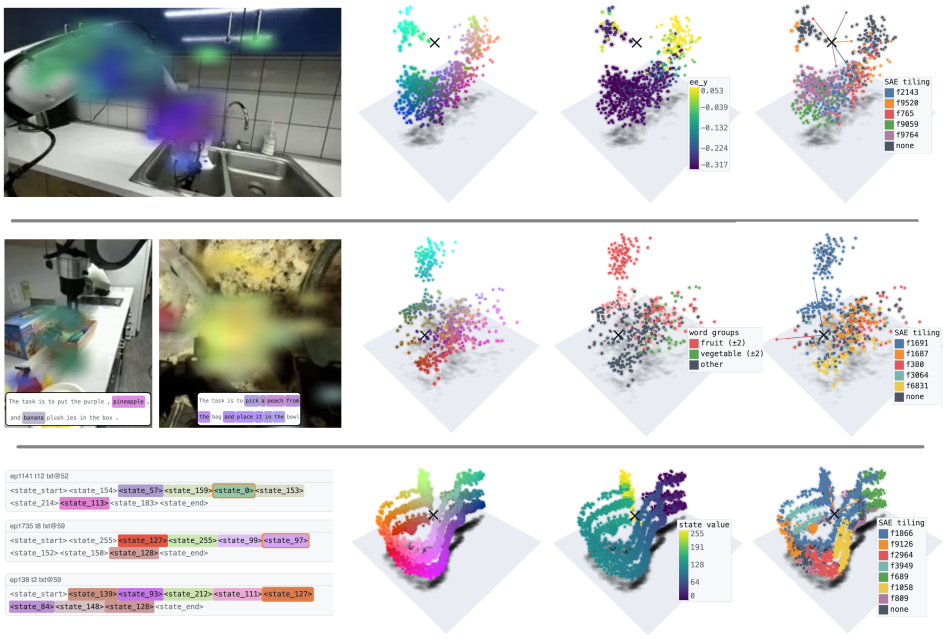


Figure 1: **Block-sparse featurizers (BSFs) recover multi-dimensional robotics concepts.** Examples from vision-language-action models encode end-effector position (top), fruit-and-vegetable semantics (middle), and proprioceptive state (bottom). Each point is a token's 3D feature code, colored by PCA coordinates, task or semantic labels, and sparse-autoencoder (SAE) assignments across columns. Left overlays share the PCA colors. SAE features tile distinct regions within each block, showing how BSFs group related variation into a single feature.

ABSTRACT

Can better feature discovery improve control over robot behavior through internal interventions and training data? Recent work in vision and language models reveals multidimensional features, motivating block-sparse featurizers (BSFs), which represent each feature in a low-dimensional subspace. We investigate whether recovering this structure provides an interpretable interface for steering vision-language-action models and shaping the data from which they learn. We show that BSFs recover several multi-dimensional task-relevant features underlying the model's computation (e.g., perception, language, proprioceptive state, episode phase, and control variables). We causally validate these features through fine-grained steering: targeted changes in visual perception, semantic representation, and spatial localization produce predictable changes in downstream action prediction. Finally, we use feature-based data attribution to isolate where a specific behavior (failure recovery) arises in the training dataset, identifying fewer than 1.4% of training tokens whose masking is sufficient to remove the targeted behavior. Overall, our results show how matching featurizers to representation geometry can provide more effective tools for understanding and changing robot behavior.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

A VLA maps visual observations, language instructions, and robot state to actions in the physical world. Although today's VLAs rarely perform consequential real-world tasks, fully learned robotics models may soon see widespread deployment. Yet little is known about the computations underlying their actions. Understanding these internal representations will be essential for safe deployment.

One way to understand the internal representation of a learned model is through perspective of *features*, which are recurring internal variables that encode concepts, states, relationships and ultimately underlie the process by which the model converts input to output. A promising line of work, e.g., sparse autoencoders (SAEs), leverage the *linear representation hypothesis* (LRH) (Park et al., 2023a; Huben et al., 2024; Gao et al., 2025) and assume that activations are composed of a sparse set of vectors. These methods have been well studied in language and vision modalities (Huben et al., 2024; Templeton et al., 2024; Gao et al., 2025). Examining models in the basis of the features that they express is useful for a plethora of problems, such as to study generalization (Gallifant et al., 2025; Olson et al., 2025), steer model behavior (Templeton et al., 2024; Prasad et al., 2026), attribute behaviors to training data (Kowal et al., 2026; Bergen et al., 2026), and identify safety-relevant computation (Templeton et al., 2024; Marks et al., 2025).

However, a featurizer is ultimately a hypothesis about the data-generating process of the activations (Fel et al., 2026a). The LRH is particularly questionable for visual domains involving perception, since continuous factors such as viewpoint, illumination, position, and motion naturally generate low-dimensional manifolds that are reflected in learned representations (Cohen et al., 2020; Kvinge et al., 2024; Fel et al., 2026a). Despite this, the majority of work in VLA interpretability assumes the LRH, with varying degrees of success. Early studies showed that specific MLP directions can offer limited control of variables like speed and height (Häon et al., 2025). Complementary studies use linear probes and activation interventions to characterize the contributions of visual, language, and action pathways and identify representations of environmental dynamics and task progress (Molinari et al., 2025; Grant et al., 2026; Bhardwaj et al., 2026; Buurmeijer et al., 2026). More recently, SAEs have been applied to recover semantic concepts, motion primitives, task structure, and episode-specific memorization, demonstrating that interventions on individual features can alter policy behavior (Swann et al., 2026; Khan et al., 2025; Jin et al., 2026; Grant et al., 2026).Prior work has improved our understanding of VLA policies, but precise behavioral steering remains challenging, with interventions often producing inconsistent, off-target, or weak effects. Much of this work relies on pooled visual activations or small activation corpora, leaving open whether training SAEs on individual token activations at greater scale improves feature discovery and steering.

In this work, we apply recent methods for multi-dimensional feature discovery—block-sparse featurizers (BSFs) (Fel et al., 2026a)—to frontier open-source VLA models, training BSFs and baseline SAEs on the largest corpora of unpooled VLA token activations to date. BSFs represent features as sparse low-dimensional blocks (Yuan & Lin, 2006; Eldar & Bolcskei, 2009; Jenatton et al., 2010; Bach et al., 2012) rather than single directions, making them a natural fit for robotic representations. Across models and layers, we recover a broad vocabulary of task-relevant features spanning visual perception, language, proprioceptive state, task variables, episode progress, and object affordances. BSFs outperform matched SAEs on recovering and intervening on task-relevant structure, suggesting that the additional geometry is useful rather than simply excess representational capacity. We then causally validate them through fine-grained steering experiments across multiple layers and models, inducing targeted changes at several stages of the policy: what is perceived in the observation, which object is selected, where that object is localized. Finally, we perform feature-based attribution to connect interpretable behaviors directly to the episodes in the fine-tuning dataset that shaped them. Using a 'failure-recovery' feature as as case-study, we isolate fewer than 1.4% of fine-tuning tokens in DROID whose influence is disproportionately associated with this behavior. Our contributions are:

1. **Comparing block-sparse and linear featurizers for VLAs.** We systematically compare BSFs and SAEs in VLAs. BSFs recover features spanning perception, language, proprioception, task state, episode progress, and affordances, better capturing computational geometry qualitatively and quantitatively. We also strengthen VLA SAE baselines with substantially larger, unpooled activation corpora to better evaluate multidimensional features.
2. **Causal validation through fine-grained steering.** Across multiple models, layers, and computational regimes, we causally validate recovered features by steering perception, semantics, target-object selection, geometric localization, and downstream robot behavior.

3. **Feature-based dataset attribution and shaping.** We trace behaviors captured by BSF features to the data that caused them. For failure recovery, masking $\sim 1\%$ of training tokens qualitatively transforms robot behavior, demonstrating precise control over robot datasets and learned policies.

2 From Sparse Directions to Sparse Subspaces

The goal of a featurizer is to factorize model activations into a set of interpretable concepts (Huben et al., 2024; Bricken et al., 2023; Gao et al., 2025; Bussmann et al., 2024; Rajamanoharan et al., 2024; Fel et al., 2025; Hindupur et al., 2026; Bussmann et al., 2025). Its feature geometry also implicitely encodes a hypothesis about how activations are generated. Choosing a featurizer for a VLA therefore requires matching this geometry to the variables the policy represents and computes. In this section, we first describe the feature unit assumed by sparse autoencoders, generalize it to block-sparse subspaces, explain their selection, and define the reconstruction metric shared by both featurizers.

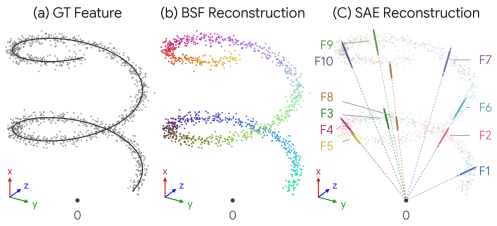


Figure 2: **BSFs better capture low dimensional manifolds.** (a). One three-coordinate BSF block reconstructs the helix. (b) Many individual SAE features tile the helix.

Notation. For a positive integer n, we write $[n] = \{1, \dots, n\}$. Vectors are bold lowercase and treated as row vectors, and matrices are bold uppercase. For $\boldsymbol{v} \in \mathbb{R}^n$, $\|\boldsymbol{v}\|_2$ is the Euclidean norm and $\|\boldsymbol{v}\|_0 = |\{i \in [n] : v_i \neq 0\}|$ counts nonzero coordinates. For a matrix $\boldsymbol{A}$, $\|\boldsymbol{A}\|_{\mathrm{F}}$ is the Frobenius norm. We write $\boldsymbol{x} \in \mathbb{R}^d$ for an activation and $\boldsymbol{z} \in \mathbb{R}^m$ for its feature code. The coordinates of $\boldsymbol{z}$ are partitioned into G disjoint groups $\boldsymbol{\mathcal{I}} = \{I_1, \dots, I_G\}$ of size b, so that $m = Gb$. The subvector $\boldsymbol{z}_g := \boldsymbol{z}_{I_g} \in \mathbb{R}^b$ contains the latent coordinates associated with feature g. For a decoder $\boldsymbol{D} \in \mathbb{R}^{m \times d}$, the corresponding rows form the decoder block $\boldsymbol{D}_g := \boldsymbol{D}_{I_g,:} \in \mathbb{R}^{b \times d}$. We omit decoder biases. The two featurizers below differ only in the value of b, and we start with the smallest one.

Sparse autoencoders. Sparse autoencoders use $b = 1$ and $m = G$. Each feature has one coordinate z_g and one decoder direction $\boldsymbol{d}_g \in \mathbb{R}^d$, and the reconstruction is

$$\hat{\boldsymbol{x}} = \sum_{g=1}^{G} z_g(\boldsymbol{x})\, \boldsymbol{d}_g = \boldsymbol{z}(\boldsymbol{x})\boldsymbol{D}, \qquad \|\boldsymbol{z}(\boldsymbol{x})\|_0 \leq k. \tag{1}$$

The decoder range is a union of spans $\mathrm{span}\{\boldsymbol{d}_g : g \in S\}$ over supports $S \subseteq [G]$ of size k. Each feature therefore contributes along a fixed direction, with its coefficient determining magnitude and, for signed codes, its orientation. Several scalar features together can represent multidimensional variation, but the SAE architecture cannot group them as single feature. The helix in Figure 2, which is a one-dimensional curve with a three-dimensional linear span, illustrates this: an SAE tiles it with scalar features covering individual sub-arcs (Bhalla et al., 2026), splitting one variable across features whose relationships must be recovered after training. This leads to a natural question: can the unit of a feature be enlarged to hold such a curve, while keeping a linear decoder?

Block-sparse featurizers. The simplest answer is to allow $b > 1$. A block-sparse featurizer (BSF) (Fel et al., 2026a) replaces the scalar z_g with a vector $\boldsymbol{z}_g \in \mathbb{R}^b$ and the direction $\boldsymbol{d}_g$ with a matrix $\boldsymbol{D}_g$. The reconstruction stays linear:

$$\hat{\boldsymbol{x}} = \sum_{g=1}^{G} \boldsymbol{z}_g(\boldsymbol{x})\boldsymbol{D}_g = \boldsymbol{z}(\boldsymbol{x})\boldsymbol{D} \tag{2}$$

and feature g now contributes within a learned subspace

$$\mathcal{S}_g := \left\{\boldsymbol{u}\boldsymbol{D}_g : \boldsymbol{u} \in \mathbb{R}^b\right\}, \qquad \dim(\mathcal{S}_g) = \mathrm{rank}(\boldsymbol{D}_g) \leq b. \tag{3}$$

An activation with k active features lies in the Minkowski sum$\mathcal{S}_{g_1} + \cdots + \mathcal{S}_{g_k}$ (Fel et al., 2026b). The decoder range becomes a union of sums of subspaces instead of a union of sums of rays. The

observed contributions need not fill $\mathcal{S}_g$. They can occupy a curve, a surface, or another structured subset $\mathcal{M}_g \subseteq \mathcal{S}_g$, such that one block to hold the helix, or a multi-dimensional variable such as gripper aperture or end-effector position. Importantly, the block dimension does not fix the intrinsic dimension of $\mathcal{M}_g$ (analyzed in Section 3) and only *bounds* the dimension above.

A larger unit also changes what can be read from an active feature. Unlike an SAE feature which provides a single scalar value, a block conveys both how strongly the feature is present via the block norm $|\boldsymbol{z}_g(\boldsymbol{x})|_2$, and where the activation lies on $\mathcal{M}_g$, i.e., the position of $\boldsymbol{z}_g(\boldsymbol{x})$ within the block, without privileging any particular basis for the block coordinates. Specifically, for any orthogonal $\boldsymbol{Q} \in O(b)$, replacing $\boldsymbol{z}_g$ with $\boldsymbol{z}_g\boldsymbol{Q}$ and $\boldsymbol{D}_g$ with $\boldsymbol{Q}^\top\boldsymbol{D}_g$, combined with a matching rotation of the block encoder, preserves the feature subspace, the reconstruction, and all block norms. Unless otherwise specified, we use this freedom to display block codes in a principal-component basis in Section 3.3.

Selecting features jointly. BSFs assume that a block is active or inactive as a whole. If its coordinates could be selected independently, the block norm would no longer measure the presence of one feature, and the block would decompose back into scalar features. Sparsity must therefore act on groups rather than on coordinates, so that the coordinates of a selected block vary together and unselected blocks are exactly zero. We count active features with the group pseudo-norm

$$\|\boldsymbol{z}\|_{0,\boldsymbol{\mathcal{I}}} := \sum_{g=1}^{G} \mathbf{1}(\|\boldsymbol{z}_g\|_2 > 0)\,, \tag{4}$$

which reduces to $\|\boldsymbol{z}\|_0$ when $b = 1$. We use the Vanilla BSF of Fel et al. (2026a), with independently learned encoder and decoder. Its per-activation encoder is

$$\boldsymbol{z}(\boldsymbol{x}) = \Pi_k(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}_{\text{enc}})\,, \qquad \boldsymbol{W} \in \mathbb{R}^{d\times m}, \quad \boldsymbol{b}_{\text{enc}} \in \mathbb{R}^m. \tag{5}$$

Here Π_k is the Euclidean projection onto $\{\boldsymbol{z} : \|\boldsymbol{z}\|_{0,\boldsymbol{\mathcal{I}}} \le k\}$. It keeps the k blocks with largest ℓ_2 norm and sets the others to zero. Retained coordinates are signed, and no coordinate-wise nonlinearity is applied. For $b = 1$, the block norm becomes an absolute value and the featurizer becomes a TopK SAE (Gao et al., 2025; Zhu et al., 2026). All SAE baselines in this paper are this special case, trained on the same data with the same pipeline, so the block dimension is the only difference between the two featurizers. In practice, we train with the BatchTopK variant of Π_k (Bussmann et al., 2024), which keeps the Bk highest-scoring blocks across a batch of B activations, with each block scored by $\|\boldsymbol{a}_{ng}\|_2\|\boldsymbol{D}_g\|_{\text{F}}$ to account for decoder scale. Because a block uses b coordinates where a scalar feature uses one, we always report which budget is matched, either the number of active features k or the number of active coordinates kb; details are given in Appendix D. With the unit of a feature and its selection rule fixed, the remaining choice is the objective used to train it.

Reconstruction and functional reconstruction. We first recall briefly the usual featurizer reconstruction loss and then introduce a functional invariant that we investigate in parallel. The standard objective is the expected squared reconstruction error,

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_{\boldsymbol{x}}\big(\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2\big)\,. \tag{6}$$

This loss treats all directions of activation space equally: an error of a given norm costs the same wherever it points. However, some directions of the activation strongly affect the predicted action, while others have almost no effect on it. We therefore also measure reconstruction error by its effect on the policy's *output*. Let $\boldsymbol{g} \in \mathbb{R}^d$ be the gradient, with respect to the activation, of the teacher-forced log-likelihood of the recorded action tokens. For an error $\boldsymbol{e} = \boldsymbol{x} - \hat{\boldsymbol{x}}$, the change in log-likelihood is, to first order, $\boldsymbol{e}\boldsymbol{g}^\top$. Averaging the square over a corpus of N gradients gives

$$\frac{1}{N}\sum_{n=1}^{N}\big(\boldsymbol{e}\boldsymbol{g}_n^\top\big)^2 = \boldsymbol{e}\,\hat{\boldsymbol{F}}\,\boldsymbol{e}^\top, \qquad \hat{\boldsymbol{F}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{g}_n^\top\boldsymbol{g}_n, \tag{7}$$

where $\hat{\boldsymbol{F}} \in \mathbb{R}^{d\times d}$ is the empirical Fisher matrix. An error along a direction of large curvature changes the predicted action and is thus heavily penalized. Say it otherwise, if there exist a null space in the activations (e.g. a neuron with no effect) then the reconstruction error in the null space of $\hat{\boldsymbol{F}}$ satisfies $\boldsymbol{e}\boldsymbol{g}_n^\top = 0$ for every n: it has no first-order effect on the action and does not need to be reconstructed.

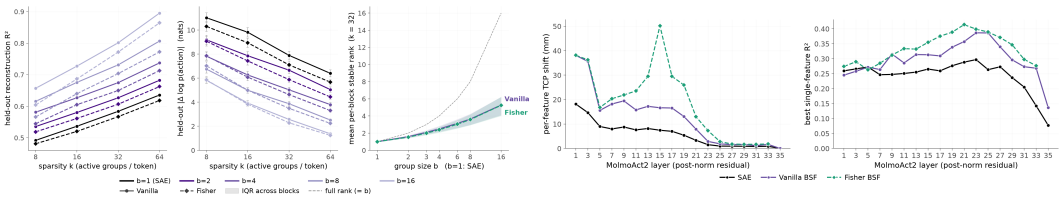


Figure 3: **Reconstruction, dimensionality, and behavioral relevance of SAE and BSF features in MolmoAct2.** *Left three panels (layer 13):* Held-out reconstruction R^2 and absolute recorded-action log-likelihood change versus active blocks k (lower change is better; mean $\pm$ SEM, 48 episodes), and within-block stable rank versus block size at $k = 32$ (bands: interquartile range; gray: full dimension). Colors denote block size ($b = 1$: SAE); solid/dashed lines denote Vanilla/Fisher training. *Right two panels (across layers):* Three-dimensional Vanilla and Fisher BSFs versus SAEs at approximately matched latent widths and active-coordinate budgets (21 blocks versus 63 latents). Ablation effects measure mean predicted end-effector displacement over the top seven BSF blocks or 21 SAE latents by activation mass. Decodability is the best single-feature ridge-probe R^2 on held-out episodes, averaged over ten robot-state and task-progress variables.

Activation corpus and SAE baselines. Since our claims about block structure's benefits are relative to an SAE, we aim for the strongest possible baseline. Prior SAEs trained on VLA activations have two limitations: the main experiments of Swann et al. (2026) mean-pool activations, combining all modalities into one vector; Grant et al. (2026) and Jin et al. (2026) avoid pooling but use small corpora—500k action-token activations and a few hundred simulated rollouts per SAE, respectively. Instead, we retain each image, text, and state token as a separate activation, yielding approximately 52M training activations collected uniformly across 45,344 episodes, roughly two orders of magnitude more than the pooled activations in the next largest dataset from Swann et al. (2026). All our featurizers train on this corpus, so differences in results can be attributed purely to featurizer structure.

3 Results

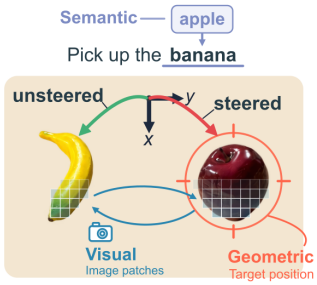
Given the three featurizers covered in (Section 2), we now turn to the features they discover and the understanding and control they admit. We first perform a quantitative comparison of BSFs to SAEs (Section 3.1) and causally validate the discovered features through open- and closed-loop steering experiments (Section 3.2). We then review a broad range of feature phenomenology, spanning visual, language, control, and proprioceptive features (Section 3.3). Finally, we demonstrate an application of BSF features for the goal of behavior modification through data attribution (Section 4).

3.1 Quantitative Comparison to SAEs

We hypothesize that BSFs—due to their inductive prior of low-dimensional features that are smooth over time—are a more natural fit to VLAs than traditional SAEs. We perform a quantitative comparison between SAEs, Vanilla BSFs, and Fisher-weighted BSFs on MolmoAct2, and use the following three metrics: (i) *activation reconstruction*, (ii) *preservation of output behavior* (i.e., action likelihood), and (iii) robotics-grounded measures of *causal influence* and *task-variable decodability*.

Figure 3 (column 1 and 2) shows held-out activation reconstruction and action fidelity (i.e., how much the teacher-forced log-likelihood of recorded actions changes when prompt activations are replaced by their reconstructions) for various methods and block sizes (note, SAEs are just a special case of BSFs where block size is one). As expected, larger blocks improve both metrics at fixed block sparsity while Fisher weighting trades lower R^2 for better preserves action likelihoods. We further estimate the effective dimensionality for the features recovered using the *stable rank* of each (Figure 3 column 3). Vanilla and Fisher BSFs both grow slower than the block dimension, reaching a mean stable rank of approximately 5.3 for 16-dimensional blocks.

Figure 3 ((column 4 and 5) shows the comparison for the two robotics-specific metrics. First, *task-variable decodability* measures how well a linear probe predicts robot state and task progress from individual features, i.e., the best held-out single-feature R^2 for each target, averaged across targets. Specifically, the targets are comprised of: end-effector position and displacement, episode progress,



Intervention	Featurizer	# Features↓	Held-out recovery ↑
Semantic L3	BSF	2.15	83.8%
Noun token	Fisher BSF*	2.56	78.9%
	SAE	3.50	81.9%
Visual L7	BSF	20.40	84.6%
63 exo-image tokens	Fisher BSF	16.33	88.3%
	SAE	36.25	97.3%
Geometric L23	BSF	1.00	99.7%
Action-readout token	Fisher BSF	1.08	83.9%
	SAE	1.00	98.2%

Figure 4: **Fine-grained steering through three representations.** Semantic, visual, and geometric interventions redirect predicted actions from the prompted object to the other object while leaving input images and instructions unchanged. Results cover 52 held-out scene–instruction comparisons with six noise seeds each. *# Features*: selected-set size, counting three-dimensional BSF blocks or scalar SAE features. *Held-out recovery*: fraction of the activation-patching effect reproduced by the selected features, averaged over comparisons for which feature discovery finds a steering set. *A steering set is found for only 50/52 comparisons; all other rows cover all 52. Details in Appendix C.1.

gripper state, distance to the first gripper-closure position, and whether that closure has occurred. Second, *per-feature causal effect* measures the change in predicted end-effector position after ablating an individual feature, where more causal features are assumed induce a larger change. Indeed, we observe that BSFs produce both larger per-feature ablation effects than SAEs across layers as well as higher task-variable decodability, particularly through the middle and later layers. Moreover Fisher-weighted BSFs strictly outperform vanilla BSFs across both causal effect and task-variable decodability.

3.2 Causal Validation of Features via Steering

Open-Loop Steering Results. A useful featurizer should enable targeted changes to policy behavior through a small number of feature edits that generalize to new scenes. We evaluate this with a redirection task in which two objects sit at opposite ends of a table, making the first predicted action chunk indicative of the robot's intended target. We instruct the policy to pick up one object and test the minimum number of feature edits required to redirect its reach toward the other. To scale this evaluation across hundreds of scene–method combinations, we use an automated search to identify compact feature sets that achieve redirection (see Appendix C.1 for full details) on a 'train' set of episodes, and then hold the selected sets fixed and evaluate their effectiveness on held-out episodes.

We apply this procedure at three sites associated with distinct aspects of target grasping (Figure 4). Editing the early-layer noun token changes *what* the robot is looking for; editing mid-layer image tokens changes *where* it perceives each object; and editing the late-layer action-readout token changes the represented target *position*. At each steering location, replacing the activations with those from the condition targeting the other object flipped the robots target action towards the corresponding object. We use this activation-patching effect as a reference, measuring held-out recovery as the fraction of our steering relative to it. Held-out evaluation tests whether the selected feature sets generalize beyond the episodes we found them on, while task completion is evaluated separately in closed loop (Table 1). Full details are provided in Appendix C.1.

Figure 4 reports three metrics averaged over 52 scenes: steerable feature search success, the number of selected features, and held-out recovery. Each feature is one SAE latent or BSF block, and 100% recovery means matching the trajectory shift produced by activation patching (Appendix C.1). All configurations yield successful interventions on 52/52 scenes except semantic steering with Fisher-weighted BSFs, which succeeds on 50/52. Geometric steering is the most compact: a single BSF block achieves 99.7% recovery, comparable to a single SAE latent. For semantic steering, standard BSFs require fewer features while slightly improving held-out recovery. Perceptual steering reveals a trade-off: the BSF variants require roughly 16–20 blocks, compared with 36 SAE latents, but the SAE transfers more effectively to held-out scenes. Overall, BSFs thus provide more compact semantic and perceptual interventions, although fewer feature edits do not consistently yield better transfer.

	SAE				BSF			
Object pair	# Feat.	Select	Grasp	Task	# Feat.	Select	Grasp	Task
Apple / banana	2.0	90%	85%	75%	2.5	100%	95%	90%
Bowl / cable	2.5	100%	95%	85%	2.0	100%	100%	100%
Mouse / towel	3.0	75%	70%	65%	2.5	100%	95%	90%
Total	2.5	88.3%	83.3%	75.0%	2.3	100%	96.7%	93.3%

Table 1: **Closed-loop semantic target redirection.** The policy is instructed to pick up one object while a fixed semantic feature intervention redirects it to the other. Entries are success rates over 20 trials per object pair (10 per direction; 60 in total): Select, the policy goes for the redirected object; Grasp, it grasps that object; Task, it places that object in the basket. # Feat.: features per direction, averaged. Per-direction counts are in Table 3.

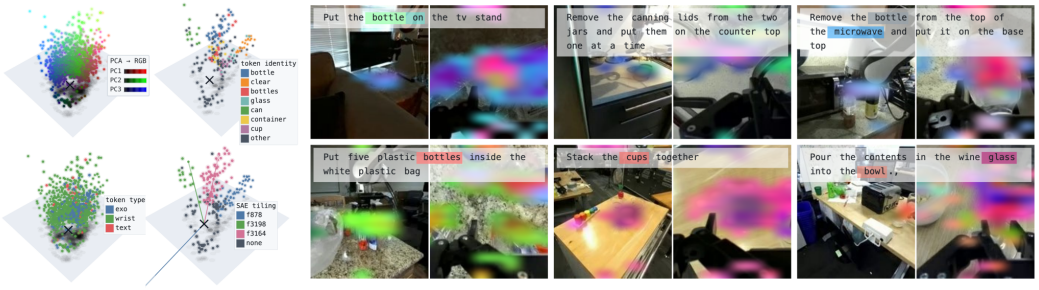


Figure 5: **A single BSF feature links related container concepts across vision and language.** The left column of point clouds shows activations colored by PCA coordinates (top) and token modality (bottom). The right column shows only activating text tokens, colored by token identity (top) and SAE assignment (bottom). The feature nicely represents different types of containers across spatially distinct positions. This feature is Vanilla BSF F1777 from $\pi_{0.5}$ layer 3.

Closed-Loop Target-Object Redirection. We next test whether semantic steering, whose intervention does not depend on the specific observation, persists through closed-loop execution. The robot is asked to "pick up the [noun] and put it in the basket" for one object, while a fixed noun-token feature intervention, selected as in Figure 4, redirects it to the other object for the whole episode. We use three object pairs, each steered in both directions for 10 trials, and randomize the object and basket positions in every trial. BSF features redirect the policy more reliably with fewer features (Table 1): they select the redirected object in every trial and complete the task in 93.3% of trials using 2.3 features on average, compared with 88.3% selection and 75.0% completion for 2.5 SAE features.

3.3 Feature Phenomenology

In this section, we highlight a representative sample of BSF features across models and layers, focusing on how variation within a block relates to interpretable properties of its inputs. Broadly, we find BSFs features associated with visual, visual-language, and robot state and control concepts. Each point in the activation visualization represents the three-dimensional code of a block at an individual token where it activates and we center and rotate these codes using PCA, retaining all three dimensions. Additional results are shown in Appendix A.

Visual Language Features. Many blocks activate across both pixels and instruction text and are common across the tested models and layers. For example, Figure 5 shows a container feature in $\pi_{0.5}$ that activates on external-camera, wrist-camera, and text tokens. Within the block, words referring to related containers occupy distinct regions. This organization is consistent with a shared vision–language representation that retains distinctions between members of a semantic category. The fruit-and-vegetable feature in Figure 1 provides another example of a block spanning related objects. Notably, three SAE features tile much of the displayed text-token geometry, while the BSF represents these related groups within a single block.

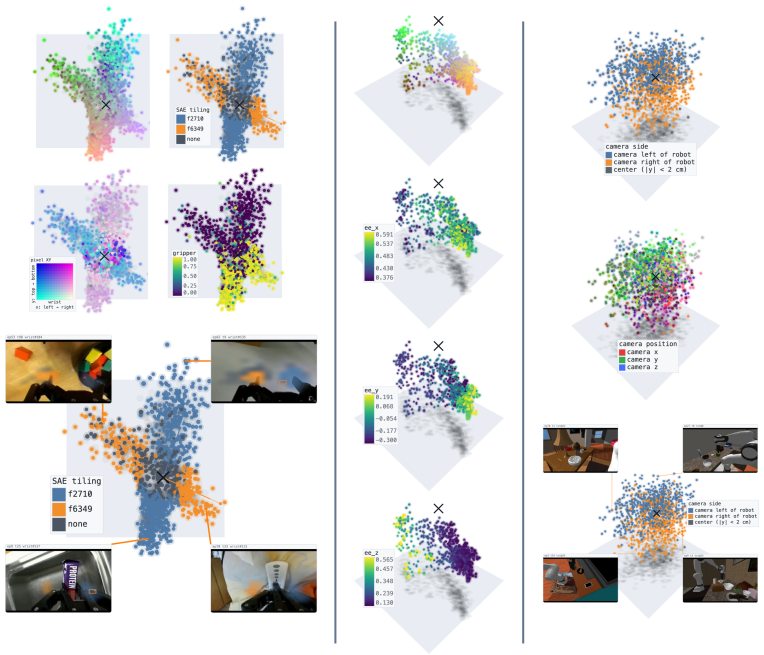


Figure 6: **A sample of BSF control-relevant features.** *Left:* MolmoACT2 layer 13, `F937`, sampled across the dataset. Colorings show PCA coordinates, SAE assignment, image-patch position, and gripper state; linked images illustrate the gripper configurations and locations associated with selected points. *Middle:* $\pi_{0.5}$ layer 7, `F2113`, sampled from a single episode and colored by PCA coordinates (top) and end-effector x, y, and z position (bottom). *Right:* MolmoBot layer 24, `F725`, colored by camera side and camera position in the robot frame, with four representative views.

Task and Control Variables. Beyond visual and semantic content, we find blocks associated with robot configuration and spatial variables (Figure 6). A gripper feature forms an `X`-shaped point cloud whose organization reflects both gripper opening and the side of the gripper on which the feature activates. Notably, each side is tiled by exactly two SAE features. The BSF block brings these complementary views of a changing two-side gripper into a single feature.

A second feature varies with the end-effector's Cartesian coordinates over a single episode. This association is notable because the policy receives joint angles as proprioceptive input, rather than explicit Cartesian end-effector coordinates. The feature also primary activates on unused image-token positions near the robot base. This use of image-tokens for information storage is a unique phenomenon to $\pi_{0.5}$ given its bidirectional attention mask.

A third example comes from MolmoBot, which is trained on synthetic data. Known camera poses allow us to relate the feature's geometry to camera position relative to the robot, including whether the camera lies to its left or right. This feature captures a property of the observation viewpoint alongside the robot-state variables illustrated above.

4 Application: Behavior Shaping via Data Attribution

Identifying failure recovery features. We next ask whether a discovered feature can identify training data responsible for a policy behavior, and whether filtering that data predictably changes the learned policy. We focus on `F1515` (Figure 7), a three-dimensional BSF feature in layer nine of $\pi_{0.5}$ that activates during *recovery from failed grasps*. We identified it using a small dataset of our own rollouts Figure 7 shows the feature's geometry and activations during repeated grasp failures and corrections in a DROID demonstration (top) and a real-robot rollout (bottom). Because it fires broadly across visual tokens, we show mean activation per frame for clarity. The feature activates during missed grasps in both settings and has a distinctly nonlinear, *conical* geometry: multiple SAE features tile this structure, while the BSF captures it in a single block.

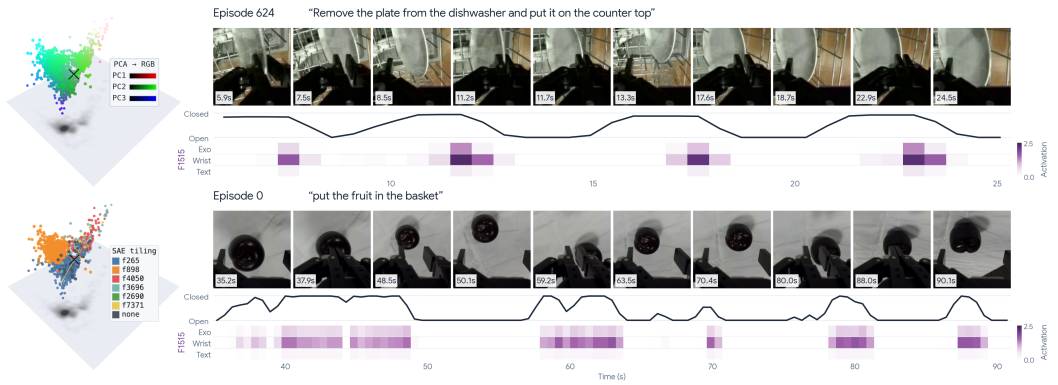


Figure 7: **Feature associated with failure recovery.** *Left:* F1515 block codes colored by PCA coordinates and SAE assignment. *Right:* Wrist observations, gripper state, and mean feature norm across exterior-image, wrist-image, and text tokens in a DROID demonstration (top) and a real-robot rollout (bottom). Feature activations are measured by replaying the recordings through $\pi_{0.5}$-DROID.

From features to training data. We identify recovery windows in DROID where F1515 activation exceeds a threshold for at least two consecutive evaluated frames (Appendix B), finding failures in approximately 6.5% of episodes. To test whether these segments teach recovery, we fine-tune from $\pi_{0.5}$-base and mask the loss for any 16-step action chunk overlapping a F1515 selected window. This removes supervision for 1.38% of action tokens while preserving observations and trajectory structure. We compare our BSF-based selection with TRAK attribution (Park et al., 2023b), a covariance-adjusted difference-of-means labeler, kNN, and random masking, matching selected-frame budgets and training recipes across methods. No feature interventions are used at inference time.

Targeted changes in recovery behavior. We evaluate recovery by manually closing the gripper before a grasp and measuring the time to reopen it (this remains in distribution because the models have no memory). Across five trials per method (Table 2), concept-based masking increases mean reopening time more than sevenfold, from 1.89 to 14.25 seconds, exceeding the mean delay of all comparison methods. In a separate real-world pick-and-place evaluation without induced failures, the concept-filtered policy completes the task with a perfect success rate and similar rollout durations (11.79 versus 12.13 seconds). These results show how interpretable feature-guided data attribution can target specific behaviors while preserving overall task performance in this evaluation.

Method	Reopen (s) ↑	Success (%) ↑	Complete (s) ↓
Unmodified	1.89	90	12.13
Random	5.00	100	10.15
Diff. means	8.52	65	17.49
kNN	3.85	100	12.82
Influence	4.90	100	11.12
Concept (ours)	14.25	100	11.79

Table 2: **Behavior shaping.** Mean reopening time (5 perturbed trials) and task performance (20 unperturbed trials; 500-step limit). Longer reopening indicates stronger recovery suppression; completion time averages successful trials only.

5 Conclusion

We have shown that BSFs recover interpretable, multidimensional features in vision-language-action models, providing tools for steering robot behavior and tracing it to training data. Targeted interventions establish these features' behavioral relevance, while feature-guided data masking selectively delays failure recovery and preserves task performance in our evaluation. In terms of limtations, our analysis does not fully explain how these features emerge or participate in policy computation; tracing their geometry to proprioceptive inputs is a promising next step. We demonstrate closed-loop semantic steering, but some interventions are evaluated only open loop. Robust closed-loop visual and geometric steering will require tracking moving image regions and adapting to changing observations and robot state. BSFs are also not uniformly superior: SAEs perform better on some quantitative metrics, and our stronger baselines underscore the importance of unpooled activations and training scale alongside feature geometry.

AI USE STATEMENT

Generative AI tools were used to support language editing and LaTeX preparation, propose or refine hypotheses, assist in designing or providing feedback on research methodologies or experiments, and implement methods. The authors reviewed all AI-assisted work and take full responsibility for the final content, claims, and artifacts presented in this paper.

REPRODUCIBILITY STATEMENT

Code for training and visualizing BSF and SAE features will be released at the time of publication.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Rika Antonova, Marco Pavone, Haruki Nishimura, Masha Itkina, and Jeannette Bohg. Cupid: Curating data your robot loves with influence functions. *arXiv preprint arXiv:2506.19121*, 2025.

Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Structured sparsity through convex optimization. *Statistical Science*, 2012.

Leon Bergen, Usha Bhalla, Sidharth Baskaran, Max Loeffler, Raphael Sarfati, Dhruvil Gala, Ryan Panwar, Santiago Aranguri, Thomas Fel, Atticus Geiger, et al. Anatomy of post-training: Using interpretability to characterize data and shape the learning signal. *arXiv preprint arXiv:2606.12360*, 2026.

Usha Bhalla, Thomas Fel, Can Rager, Sheridan Feucht, Tal Haklay, Daniel Wurgaft, Siddharth Boppana, Matthew Kowal, Vasudev Shyam, Jack Merullo, et al. Do sparse autoencoders capture concept manifolds? *arXiv preprint arXiv:2604.28119*, 2026.

Atiksh Bhardwaj, Edward Weiyi Duan, Prithwish Dan, Wei-Chiu Ma, and Preston Culbertson. Decoding task progress from vla representations. *arXiv preprint arXiv:2608.13474*, 2026.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.

Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv:2503.17547*, 2025.

Hugo Buurmeijer, Carmen Amo Alonso, Aiden Swann, and Marco Pavone. Observing and controlling features in vision-language-action models. *arXiv preprint arXiv:2603.05487*, 2026.

Uri Cohen, SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. Separability and geometry of object manifolds in deep neural networks. *Nature communications*, 11(1):746, 2020.

Abhay Deshpande, Maya Guru, Rose Hendrix, Snehal Jauhri, Ainaz Eftekhar, Rohun Tripathi, Max Argus, Jordi Salvador, Haoquan Fang, Matthew Wallingford, et al. Molmob0t: Large-scale simulation enables zero-shot manipulation. *arXiv preprint arXiv:2603.16861*, 2026.

Yonina C Eldar and Helmut Bolcskei. Block-sparsity: Coherence and efficient recovery. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2885–2888. IEEE, 2009.

Thomas Fel, Ekdeep Singh Lubana, Jacob S Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba Ba, and Talia Konkle. Archetypal sae: Adaptive and stable dictionary learning for concept extraction in large vision models. *arXiv preprint arXiv:2502.12892*, 2025.

Thomas Fel, Matthew Kowal, Mozes Jacobs, Dron Hazra, Usha Bhalla, Lee Sharkey, Lucius Bushnaq, Satchel Grant, Tal Haklay, Thomas Icard, et al. Structuring sparsity: Block-sparse featurizers capture visual concept manifolds. *arXiv preprint arXiv:2606.25234*, 2026a.

Thomas Fel, Binxu Wang, Michael Lepori, Matthew Kowal, Andrew Lee, Randall Balestriero, Sonia Joseph, Ekdeep Singh Lubana, Talia Konkle, Demba Ba, et al. Into the rabbit hull: From task-relevant concepts in dino to minkowski geometry. In *International Conference on Learning Representations*, volume 2026, pp. 154538–154584, 2026b.

Jack Gallifant, Shan Chen, Kuleen Sasse, Hugo Aerts, Thomas Hartvigsen, and Danielle Bitterman. Sparse autoencoder features for classifications and transferability. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 29927–29951, 2025.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations*, volume 2025, pp. 26721–26754, 2025.

Bryce Grant, Xijia Zhao, and Peng Wang. Not all features are created equal: A mechanistic study of vision-language-action models. *arXiv preprint arXiv:2603.19233*, 2026.

Bear Häon, Kaylene Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. *arXiv preprint arXiv:2509.00328*, 2025.

Sai Sumedh R Hindupur, Ekdeep S Lubana, Thomas Fel, and Demba Ba. Projecting assumptions: The duality between sparse autoencoders and concept geometry. *Advances in Neural Information Processing Systems*, 38:11649–11699, 2026.

Robert Huben, Hoagy Cunningham, Logan Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations*, volume 2024, pp. 7827–7845, 2024.

Rodolphe Jenatton, Guillaume Obozinski, and Francis Bach. Structured sparse principal component analysis. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 366–373. JMLR Workshop and Conference Proceedings, 2010.

Xinchen Jin, Aditya Chatterjee, Pranav Kumar, and Rohan Paleja. Event-grounded sparse autoencoders for vision-language-action policies. *arXiv preprint arXiv:2605.17204*, 2026.

Momin Ahmad Khan, Novak Boskov, Fatima M Anwar, and Manzoor A Khan. Controlling vision–language–action policies through sparse latent directions. In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Matthew Kowal, Goncalo Paulo, Louis Jaburi, Tom Tseng, Lev E McKinney, Stefan Heimersheim, Aaron David Tucker, Adam Gleave, and Kellin Pelrine. Concept influence: Leveraging interpretability to improve performance and efficiency in training data attribution. *arXiv preprint arXiv:2602.14869*, 2026.

Henry Kvinge, Grayson Jorgenson, Davis Brown, Charles Godfrey, and Tegan Emerson. Internal representations of vision models through the lens of frames on data manifolds. In Sophia Sanborn, Christian Shewmake, Simone Azeglio, and Nina Miolane (eds.), *Proceedings of the 2nd NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, volume 228 of *Proceedings of Machine Learning Research*, pp. 75–115. PMLR, 16 Dec 2024. URL `https://proceedings.mlr.press/v228/kvinge24a.html`.

Samuel Marks, Johannes Treutlein, Trenton Bricken, Jack Lindsey, Jonathan Marcus, Siddharth Mishra-Sharma, Daniel Ziegler, Emmanuel Ameisen, Joshua Batson, Tim Belonax, et al. Auditing language models for hidden objectives. *arXiv preprint arXiv:2503.10965*, 2025.

Marco Molinari, Leonardo Nevali, Saharsha Navani, and Omar G Younis. Emergent world representations in openvla. *arXiv preprint arXiv:2509.24559*, 2025.

Matthew Lyle Olson, Musashi Hinck, Neale Ratzlaff, Changbai Li, Phillip Howard, Vasudev Lal, and Shao-Yen Tseng. Probing the representational power of sparse autoencoders in vision models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6226–6236, 2025.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023a.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023b.

Aaditya Vikram Prasad, Connor Watts, Jack Merullo, Dhruvil Gala, Owen Lewis, Thomas McGrath, and Ekdeep Singh Lubana. Features as rewards: Scalable supervision for open-ended tasks via interpretability. *arXiv preprint arXiv:2602.10067*, 2026.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Aiden Swann, Lachlain McGranahan, Hugo Buurmeijer, Monroe Kennedy III, and Mac Schwager. Sparse autoencoders reveal interpretable and steerable features in vla models. *arXiv preprint arXiv:2603.19183*, 2026.

Adly Templeton, Tom Conerly, Jonathan Marcus, et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/`.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(1):49–67, 2006.

Xudong Zhu, Mohammad Mahdi Khalili, and Zhihui Zhu. Abstopk: Rethinking sparse autoencoders for bidirectional features. In *International Conference on Learning Representations*, volume 2026, pp. 50325–50352, 2026.

A Extended Phenomenology Results



Figure 8: **Visual BSF features capture variation across task stages and robot parts.** (left) Marker feature `F2973`, with points colored by task completion. Linked images illustrate changes in appearance across a single episode; the larger overlays show the four highest-activating timesteps in the dataset. (right) Robot feature `F2571`, with PCA colors and linked images showing how different regions of the block correspond to different robot parts. Both features are Fisher-weighted BSFs from MolmoACT2 layer 7.

Visual Features. We find visual features similar to those reported by Fel et al. (2026a) in DINOv3 Siméoni et al. (2025), now in VLA activations on robotics data (Figure 8). These include features selective for objects in the environment and for the robot itself. In contrast to the features found in DINOv3, latent reconstruction is not part of the VLA training objective.

We show two example features in Figure 8, one which activates on 'markers' and another which activates on the robot. Both features illustrates how a block can capture concept variation within a single object category: the marker's activation geometry varies with the appearance of markers at different stages of manipulation, while the robot feature instead organizes patches from different parts of the arm. In both cases, the block captures recognizable visual content together with structure within that content. Across our models and layers we find plentiful features for a wide range of objects.

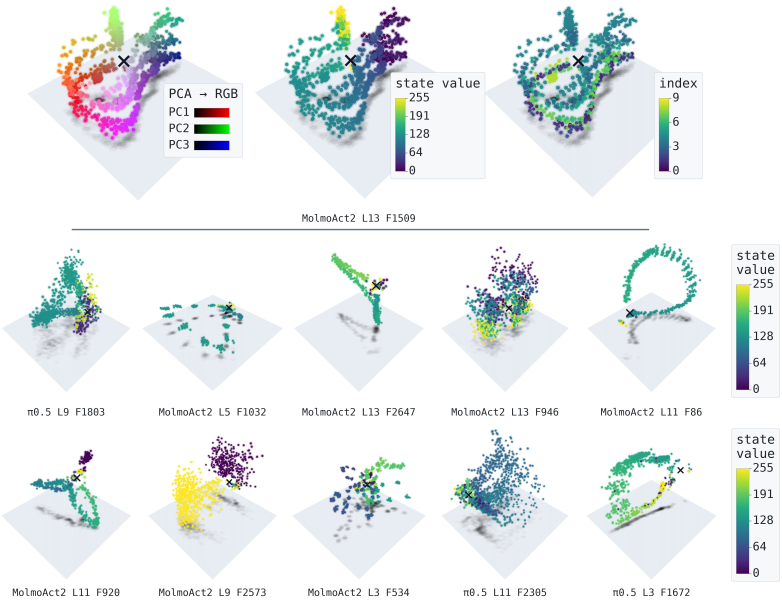


Figure 9: **Sampling of vanilla BSFs propriocetion features.** *Top:* A MolmoACT2 layer-13 feature shown with PCA coloring (left), quantized state value from 0 to 255 (middle), and state-component index (right).

Proprioceptive Features. Proprioception, the robots state information, enters the VLM backbone as part of the prompt. The two models encode these inputs differently: $\pi_{0.5}$ expresses discretized

state values as numeric text, while MolmoACT2 uses dedicated continuous state tokens. Despite this difference, both models exhibit features with geometry related to proprioceptive values as shown in Figure 9.

The highlighted MolmoACT2 feature notably forms a complex manifold of three looping branches that converge into a common extension. Coloring by quantized state value reveals an ordering along the curves, while coloring by state-component index reveals additional structure associated with which component is represented. The block thus organizes both numeric value and component identity. Other examples exhibit discrete clusters, curved trajectories, and more diffuse clouds, demonstrating the range of geometries associated with proprioceptive inputs across layers and models.

B Behavior Shaping: Implementation Details

Concept Based Attribution. We labeled 81 timesteps of failed-grasp *recovery* and 25 timesteps of successful grasps, drawn from real-robot rollouts of $\pi_{0.5}$-DROID on a pick-and-place task. `F1515` fires on 56 of the 81 recovery timesteps and on none of the 25 successful grasps. We use this feature to scan the full DROID training set for other failure-recovery behavior. For each timestep we record the `F1515` block norm summed over all image and text tokens, E, and the number of tokens where the block is active, C. A timestep is *active* if $E \geq 150$ and $C \geq 50$. Runs of at least two consecutive active frames, with single-frame gaps bridged, form a recovery event. Across the entire dataset this flags 4,866 of 74,604 episodes (6.5%). The median duration is 1.1s.

Baselines. Every baseline scores the same evaluated frames. Its threshold is chosen so that its windows, built with the same event and padding rules, cover exactly as many raw frames as the `F1515` windows (150,656).

- **Difference of means.** A direction in the mean-pooled layer-9 residual, fit to separate the 81 recovery timesteps from all other frames of the same seven episodes, with the within-class covariance shrunk toward the identity ($\alpha = 0.95$, chosen by nested leave-one-episode-out cross-validation). Frames are scored by projection onto this direction.
- **kNN.** Each frame's mean-pooled layer-9 residual, centered on the annotated-rollout mean, is scored by its mean cosine similarity to the five nearest recovery timesteps minus its mean cosine similarity to the five nearest successful-grasp timesteps.
- **Influence Functions.** Following CUPID (Agia et al., 2025), we estimate *action influence*, the effect of a training state–action pair on the policy's log-likelihood of a query action, with TRAK (Park et al., 2023b). We compute per-example gradients of the flow-matching loss with respect to the action expert (430M parameters) of the baseline policy, randomly project them to 4,096 dimensions, and apply the Gauss–Newton kernel. CUPID weights action influence by the binary return of closed-loop rollouts, estimating each demonstration's effect on expected return, and curates whole demonstrations. We target a specific behavior rather than task success. The query is therefore the annotated recovery timesteps, contrasted against the successful-grasp timesteps, and each training frame is scored as its mean influence on the recovery queries minus its mean influence on the successful-grasp queries. Scores stay at the frame level, so the same budget and windowing apply as for every other labeler.
- **Random.** Windows with the same count and length distribution as the `F1515` windows, placed uniformly at random in episodes `F1515` did not flag.

Selections overlap little with the concept windows: Jaccard similarity over raw frames is 0.20 for difference of means, 0.09 for kNN, 0.02 for TRAK and 0.00 for random. Figure 10 shows the highest-scoring frames selected by each method.

Training. All policies are fine-tuned from $\pi_{0.5}$-base on 98% of DROID (73,112 episodes); the remaining 2% is held out. Training runs for 25k steps at batch size 1024 with a constant learning rate of 10^{-4} after a 1k-step warmup, on 32 H100s. All runs share the same seed and data order. For the masked policies, the flow-matching loss is zeroed for any 16-step action chunk containing a selected frame; observations are unchanged. Because window lengths differ, the masked fraction of action tokens varies slightly: 1.38% for concept, 1.28% for difference of means, 1.27% for kNN, 1.44% for TRAK and 1.27% for random.

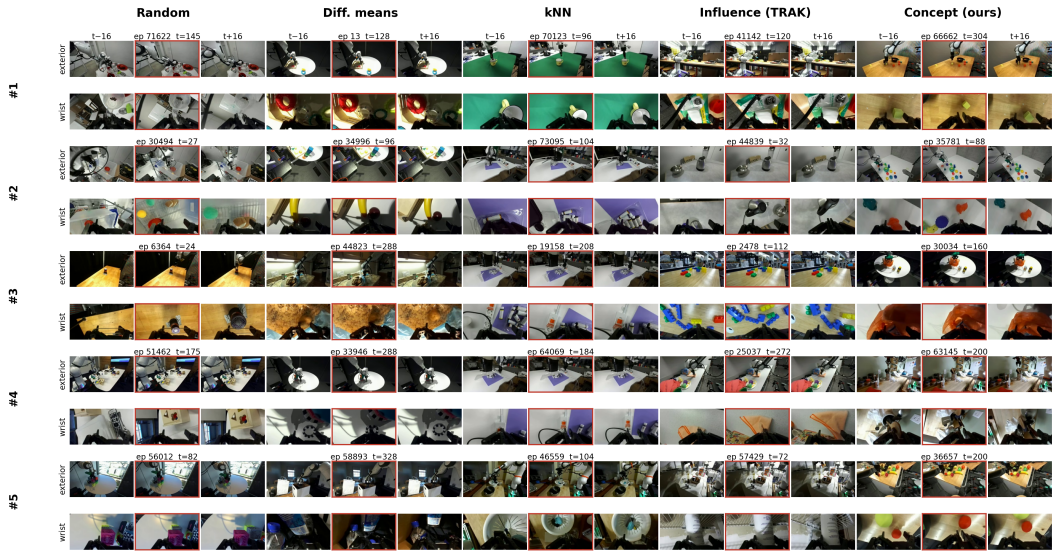


Figure 10: **Highest-scoring frames for each labeler.** Each column shows five DROID episodes for one method. For the four scored methods, these are the five highest-scoring episodes, and t is the highest-scoring evaluated frame within that method's selected windows. Random has no score, so its column shows five randomly drawn windows with t at each window's midpoint; its row order carries no meaning.

C STEERING

C.1 OPEN-LOOP STEERING PROTOCOL

Our goal is to find a set of features which steers the robot toward a target object on a set discovery inputs and then evaluated, on held-out inputs with fresh noise seeds. Consider a scene with a banana on the left and an apple on the right. We task the robot with "pick up the banana", our goal is to steer it to instead pick up the apple using three different means: steering semantic representations (changing what the model thinks the target object it), steering visual tokens (changing where them model sees the target object) or geometry representations (changing the models internal representation of the location of the target object). For each of these we have a different held out set to evaluate our steering on. Semantic steering is evaluated on the other mirrored layout. So for example if we have a set of features which redirects the policy from the banana towards the apple we may evaluate this on a different scene where the position of the banana and apple are flipped. For visual token steering we evaluate with the other object's instruction and for geometry we evaluate on an entirely new scene (leave-one-pair-out over the 13 pairs). Steering results are conducted on MolmoACT2.

Finding features. We want to find a compact set of features that steers the policy as well as patching the full activation does, since the number of features needed measures how directly a featurizer's units capture the representation that drives the behaviour. Searching all feature subsets is intractable, so we use a greedy procedure. Candidates are the BSF blocks or SAE latents are ranked by how much their activation differs between the prompted and redirected conditions. We add features to a group until its steering effect recovers at least 75% of the patching effect. Then we remove features one at a time while the set still recovers over 75% of patching effect, so the reported sets are small but not guaranteed to be the smallest possible.

Applying features. The policy represents the task at different layers and tokens in different forms. Activation patching localizes three sites where the target object is determined: the early noun token encodes *which* object is requested, mid-layer image tokens encode *where* each object appears, and the late action-readout token encodes *where* the arm should move. At the L3 noun token, the selected features' contribution is replaced with that of the redirected instruction. At the 63 L7 image tokens,

the recipient's activation within the selected features' subspace is replaced, token by token, with that of the mirrored scene. Geometric edits add one fixed vector at the L23 action-readout token.

Measuring recovery. We compare the endpoint of the predicted action chunk under steering and under activation patching, each relative to the unmodified policy and averaged over six held-out noise seeds:

$$R = 100\,\frac{\langle \Delta x_{\text{steer}}, \Delta x_{\text{patch}} \rangle}{\|\Delta x_{\text{patch}}\|^2}.$$

A value of 100% means steering moves the endpoint as far as patching, in the same direction. Our patching replaces the full activation at the noun token, all 196 exo-image tokens, or all 79 text tokens at the intervention layer.

C.2 Closed-loop Steering

Feature sets for the closed-loop evaluation are selected with the semantic procedure of Appendix C.1, adapted to a new environment. Because no benchmark scene needs to be held out, each object pair and direction is calibrated on both mirrored layouts at once: a feature set qualifies only if it reaches at least 75% recovery with cosine alignment at least 0.90 on each layout, and the frozen donor codes are the average of the two layouts' donor codes. The search runs directly on the dashboard's steering operations (project out each selected feature, then add its donor code), so the deployed intervention is exactly the calibrated one. Feature sets are fixed before evaluation and applied without gain tuning at the noun token of the prompt "pick up the [noun] and put it in the basket" throughout each episode. Each object pair is evaluated in both directions with 10 trials per direction.

		A → B				B → A			
Objects (A / B)	Featurizer	Feat.	Select	Grasp	Successful	Feat.	Select	Grasp	Successful
Apple / banana	SAE	2	10/10	9/10	7/10	2	8/10	8/10	8/10
	BSF	2	10/10	10/10	9/10	3	10/10	9/10	9/10
Cable / bowl	SAE	3	10/10	9/10	8/10	2	10/10	10/10	9/10
	BSF	2	10/10	10/10	10/10	2	10/10	10/10	10/10
Towel / mouse	SAE	4	6/10	5/10	4/10	2	9/10	9/10	9/10
	BSF	4	10/10	9/10	8/10	1	10/10	10/10	10/10

Table 3: **Closed-loop redirection by direction.** Trials out of 10. A → B: instructed to pick up A while steered toward B. Feat.: number of features in the intervention; Grasp counts trials that grasp the redirected object; Successful counts trials that place it in the basket.

D Featurizer Training and Evaluation

Activation Collection We record the residual stream at the output of language backbone block L (zero-indexed). Featurizers are trained at every odd block of MolmoAct2 ($L \in \{1, 3, \ldots, 35\}$) and of $\pi_{0.5}$ ($L \in \{1, 3, \ldots, 17\}$), and at blocks $\{1, 3, 13, 15, 24\}$ of MolmoBot. Every image, text, and state token of the is kept as a separate activation. We collect activations from MolmoAct2 and $\pi_{0.5}$ on the DROID dataset (Khazatsky et al., 2024).For MolmoBot, which is trained entirely in simulation, we collect activations on its own training distribution Deshpande et al. (2026). We collect and train on approximately 50M activations for each model and layer combination. To elminate issues with outlier tokens we filter tokens by activation magnitude $[p_{0.5},\ \min(p_{99.5},\ 8 \cdot \text{median})]$, removes between 1–5% of tokens depending on the layer. We then subtract a per-position mean computed over training tokens and rescale activations to a mean squared norm of d.

Training Decoder blocks are initialized as Gaussian matrices with unit Frobenius norm, the encoder is initialized to the decoder transpose, and $\boldsymbol{b}_{\text{enc}} = \mathbf{0}$. Once training begins the encoder and decoder are untied. The loss is the squared error of reconstructing the clean input from a lightly corrupted one: before encoding we add isotropic Gaussian noise with per-coordinate standard deviation $0.1\|\boldsymbol{x}\|_2/\sqrt{d}$ (1% of the token's energy). To revive unused blocks we add the AuxK loss of Gao et al. (2025) The featurizers used throughout the paper (phenomenology, steering, data shaping, and the per-layer

comparison) are trained with a $b = 3$ BSF with $k = 21$ and the SAEs are a $b = 1$ BSF with $k = 63$. Both have the same number of coordinates in total. They train for 10 epochs, one 1 H200.

Fisher-weighted featurizers We train Fisher-weighted featurizers for MolmoAct2 at every odd block up to 33. For 2,000 DROID episodes, we compute the gradient of the teacher-forced log-likelihood of the recorded action tokens with respect to the residual stream, yielding roughly 28M activation–gradient pairs per layer, and clip gradient norms to their $[p_{0.5},\ p_{99.5}]$ range.