Controlling Vision–Language–Action Policies through Sparse Latent Directions

Momin Ahmad Khan[1], **Novak Boškov**[2], **Fatima Muhammad Anwar**[1], **Manzoor Ahmad Khan**[2]

[1]University of Massachusetts Amherst
[2]Nokia Bell Labs
{makhan, fanwar}@umass.edu, {novak.boskov, manzoor.a.khan}@nokia-bell-labs.com

Abstract

Vision–language–action (VLA) agents combine perception, language, and control to perform general-purpose tasks, but their internal decision-making is poorly understood and hard to steer. This opacity limits trust and safe deployment in robotics (*i.e., embodied AI*). In this work, we show that discrete robot actions can be steered by identifying a small number of meaningful features inside the residual stream of a VLA policy. Using a Magma-style model with a ConvNeXt vision encoder and a LLaMA-3-8B-Instruct decoder in the *SimplerEnv* simulator, we learn *behavior directions* from contrastive pairs of inputs that differ only in the target action (e.g., *open* vs. *close* gripper). Specifically, we use a sparse autoencoder (SAE) fitted to the decoder's residual stream to construct steering vectors in latent space, which are then decoded back and applied at inference time. This intervention reliably shifts the model's action choice while preserving overall coherence. Our analysis shows that steering is effective but not perfectly disentangled due to inadvertent activations of related features during steering. These results provide the first evidence that latent-space techniques can steer embodied multimodal policies without retraining. More broadly, this work highlights that mechanistic interpretability techniques (*e.g.,* SAE) can provide handles to control action-level behavior of complex agents.

1 Introduction

Interpreting and controlling the behavior of large neural networks is a central challenge in modern AI [6, 16]. *Mechanistic interpretability* seeks to go beyond performance metrics by analyzing internal components, such as attention heads, MLP neurons, and residual streams, to explain how specific computations and concepts are represented and used [18]. This perspective not only improves our understanding of model internals, but also opens the door to targeted interventions at inference time.

Recent work has shown that models can often be *steered* by editing their internal representations, biasing them toward or away from specific behaviors without retraining [2, 21, 13, 14, 19, 5, 3]. Such edits are typically derived from *contrastive pairs* of inputs that differ only in the target property, yielding directions in representation space that can be added or subtracted from the residual stream during inference. While this approach has been explored in language models for attributes such as sentiment or honesty, its potential beyond the text domain (*i.e.,* in embodied AI agents) remains vastly untested.

In this work, we take a first step toward steering robotic action policies using mechanistic interventions. We focus on *vision–language–action* (VLA) models such as Magma [23], which combines a ConvNeXt [11] vision encoder with a LLaMA-3-8B-Instruct text decoder to map observations and instructions into action logits. We introduce a sparse autoencoder (SAE)-based steering method that learns behavior-specific directions (*e.g., open* vs. *close* gripper) from contrastive robot scenar-

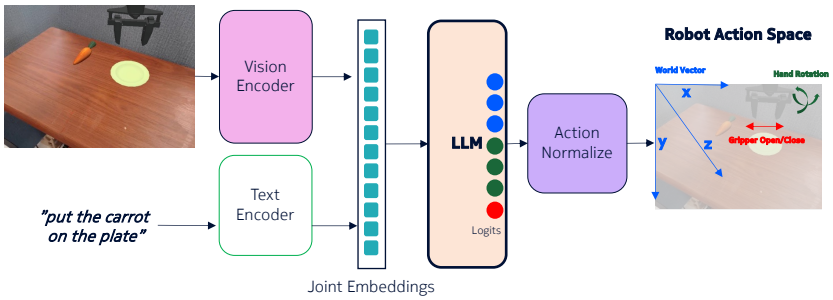

Figure 1: Anatomy of a vision–language–action (VLA) policy. Visual and textual inputs are encoded into joint embeddings, processed by a language model, and mapped to action logits. Action normalization produces the robotic control in the action space of a concrete robot. We are the first to explore steering in the *action space* of robotic policies.

ios. These directions are constructed in the SAE latent space and decoded back into the residual stream, where they are injected during inference. This minimal edit reliably shifts action selection while remaining interpretable, providing a practical bridge between mechanistic interpretability and embodied control.

1. We present the first *proof-of-concept* application of sparse autoencoder (SAE)–based steering within the *action space* of a vision–language–action (VLA) policy.
2. We propose a lightweight pipeline that learns behavior-specific directions (*e.g.,*, *open* vs. *close* gripper) from contrastive action pairs in the text decoder and applies them as inference-time residual edits.
3. Through an empirical case study on Magma [23] in SimplerEnv [9], we demonstrate that such edits can reliably bias a binary robotic action and reveal key challenges such as co-activation and disentanglement of latent features, highlighting open directions for future work on scalable multimodal steering.

2 Background

Residual streams in transformers: In decoder-only transformers, each layer L takes the residual stream $R^{(L)} \in \mathbb{R}^{d_{\text{model}}}$ from the previous layer, applies attention and MLP sublayers, and adds the result back into $R^{(L)}$ via residual connections [1]. This residual stream can be interpreted as the model's current *state of computation*: a linear space in which many high-level features are represented as directions. Probing or editing $R^{(L)}$ at specific tokens allows us to inspect or influence the model's intermediate computations.

Feature directions: A *feature direction* [2] $v^{(L)} \in \mathbb{R}^{d_{\text{model}}}$ is a unit vector in the residual space of layer L that corresponds to a feature or concept. Projecting $R^{(L)}$ onto $v^{(L)}$ yields a scalar activation

$$s^{(L)} = \langle R^{(L)}, v^{(L)} \rangle,$$

which measures the strength of that feature at that layer and token. High positive (or negative) activations often correlate with the presence (or absence) of the feature in the model's current computation.

Contrastive steering: One practical way to isolate feature directions is through *contrastive pairs* of inputs that differ only in a target property (*e.g.,* honest vs. dishonest answers, open vs. close gripper) [21, 13, 2]. Subtracting their activations cancels out nuisance variation and highlights the subspace tied to that property. From many such differences, one can extract a low-dimensional axis (via averaging, PCA, or other methods) that serves as a steering direction $v^{(L)}$. At inference time, adding or subtracting a scaled version $\alpha\, v^{(L)}$ to the residual stream biases the model toward or away from the target property:

$$R'^{(L)} = R^{(L)} + \alpha\, v^{(L)}.$$

Sparse autoencoders (SAEs): A sparse autoencoder learns an overcomplete, sparsely activated basis for residual stream activations [24, 12, 7]. Given $R^{(L)}$, an SAE encoder maps it to a high-dimensional

latent vector $z \in \mathbb{R}^{d_{\text{latent}}}$, where most components are zero for any given input. Each latent dimension is intended to correspond to a more interpretable "feature neuron." The decoder reconstructs the residual stream from these latents. Steering can then be performed in latent space: contrastive pairs define a vector in z-space, which is added before decoding back to $R^{(L)}$. This enables finer-grained interventions, since edits can target individual features rather than arbitrary residual directions.

Mechanistic interpretability connection: Both residual-space and SAE-latent steering operate directly on a model's internal representations, shedding light on how features are encoded while enabling intervention [8, 10]. By localizing edits to specific layers, tokens, and features, they bridge interpretability and control, offering a way to probe and modulate model behavior without retraining.

3 Method

Setup. We use a Magma-style vision–language–action (VLA) policy: a vision encoder produces embeddings that condition a LLaMA-3B-Instruct backbone. We insert a sparse autoencoder (SAE) trained on the residual stream at layer 25 from SAE Lens[4]. The SAE encodes each residual vector $R^{(L)} \in \mathbb{R}^{d_{\text{model}}}$ into a sparse latent $z^{(L)} \in \mathbb{R}^{d_{\text{latent}}}$ via $z = \text{ReLU}(W_{\text{enc}}R + b_{\text{enc}})$, and reconstructs back to the residual space via $R \approx W_{\text{dec}}z + b_{\text{dec}}$.

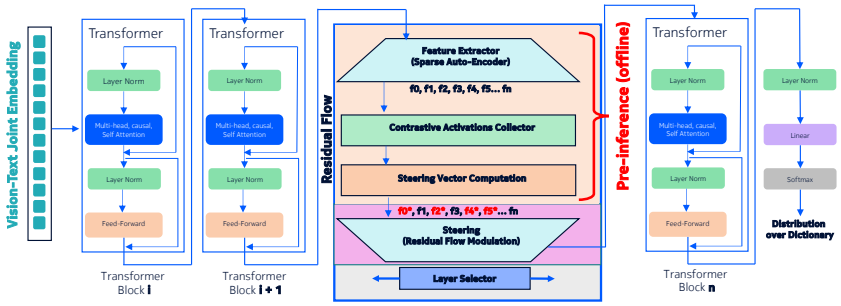

Figure 2: Our Inference-time intervention framework. The residual stream is passed through a sparse autoencoder (SAE) to extract latent features and collect the feature activations of contrastive pairs. The top three modules in the orange box constitute the *pre-inference* phase, run offline to prepare steering vectors. During inference (pink box), the steering vector is injected back into the residual flow, modulating downstream computation. This layer-agnostic design enables fine-grained and interpretable control over model outputs.

Contrastive pairs for actions: We collect *contrastive pairs* (x_i^+, x_i^-) that differ only in the target action (e.g., "open" vs. "close" gripper) while holding all other inputs constant (similar scene and prompt). To capture the relevant activations, we run an instrumented version of the model with a feature extractor that records the residual stream at the action decision step, building contrastive representations for each pair.

Token selection: We compute the steering direction using the hidden representation at the last token of the input sequence. This choice follows standard practice in mechanistic steering and activation-editing literature, where the final token embedding captures the model's full contextualized representation of the input and influences the subsequent output generation [19, 20, 17]. We therefore did not perform a token-sensitivity analysis, as the last-token representation is commonly used to define steering vectors that summarize the model's predicted behavior over the entire sequence.

Steering vector computation: We further process the qualified contrastive pairs by eliminating the rarely activated features and those that often activate in both x_i^- and x_i^+. For each pair (x_i^+, x_i^-), we collect the SAE latents $z^{(L)}(x_i^+)$ and $z^{(L)}(x_i^-)$, compute their difference $\Delta z_i^{(L)} = z^{(L)}(x_i^+) - z^{(L)}(x_i^-)$, and average over all pairs:

$$v_{\text{latent}}^{(L)} = \frac{1}{N}\sum_{i=1}^{N} \Delta z_i^{(L)}. \tag{1}$$

This yields a single *steering vector* in the SAE latent space that points in the direction of the target action concept.

Control edit in latent space. At inference time, we hook into the SAE at layer $L^\star$, and then:

1. Encode the current residual vector $R^{(L^\star)}$ into latents z.
2. Apply the edit $z' = z + \alpha\, v_{\text{latent}}^{(L^\star)}$ (or with a sign flip to bias toward the opposite action).
3. Decode back to residual space via $R' = W_{\text{dec}} z' + b_{\text{dec}}$.
4. Replace the original residual $R^{(L^\star)}$ with R' before passing to the next transformer block.

The scalar α controls edit strength; we sweep it to examine steering magnitude and side effects.

4 Results and Discussion

Experimental Setup

We evaluate on *SimplerEnv[9]*, a simulated environment with object manipulation tasks. In this section, we use the "put carrot on plate" task from BridgeDataV2 [22]. The policy backbone is a Magma-style architecture composed of a ConvNeXt vision encoder and a LLaMA-3-8B-Instruct text decoder. As Magma backbone consisists of LLaMA, we adopt an existing SAE from the SAE Lens library[4] that was trained on LLaMA-3-8B-Instruct. This SAE is designed for the residual stream at the 25th layer of LLaMA, which we align with the corresponding layer in the Magma text decoder. Thus, we insert the SAE directly into the text portion of Magma and apply our latent-space interventions there. We focus on the residual stream at the final action token.

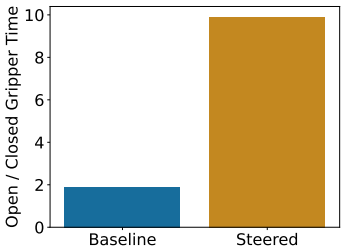

Figure 3: Effect of steering robot arm gripper using open gripper as the positive behavior (*i.e.,* x_i^+). Y-axis is the ratio of open to closed gripper simulation steps.

Steering effectiveness: Figure 3 shows that editing in SAE latent space can reliably bias the gripper action. By constructing steering vectors from contrastive pairs (*e.g.,* scenes differing only in gripper state) and applying them through SAE features, we were able to bias the policy toward more frequent gripper open. Through this pairing technique we found that feature 4909 was one of the dominant feature for gripper action. Amplifying its direction in the residual stream has a significant effect on the state of the gripper in the robot action space. This demonstrates that sparse autoencoder (SAE) latents provide a workable substrate for controlling vision–language–action (VLA) agents at inference time, where small, interpretable edits in latent space can reliably influence discrete robot behaviors without finetuning.

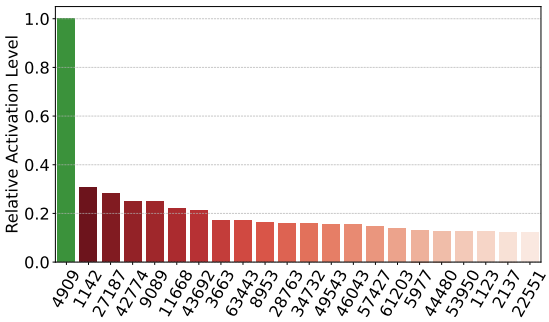

Figure 4: Top SAE features sorted by mean activation difference. Feature 4909 is effective, but additional features also shift, showing unintended co-activations. We color the feature 4909 green to emphasize we inteded to steer it. The features in red are the features that are steered unintendedly due to feature entanglement. The gradient shows a decreasing impact of steering on features.

Inadvertent co-activations. While steering feature 4909 effectively changes the gripper state in the action space, Figure 4 shows that steering feature 4909 alone results in many inadvertent co-activations. These co-activations are weaker but persistent, indicating that SAE-based directions are not perfectly disentangled, making it difficult to steer one feature in isolation. Dependeing on the context, the inadvertent activations may influence policy behavior in unintended ways. Therefore, the central challenge is not only identifying useful features, but also developing methods to mitigate inadvertent co-activations during intervention.

5 Limitations and Conclusion

Limitations and future work: Our study is restricted to simulations, a single binary behavior (open/close gripper), and SAE models trained on the text-decoder portion of Magma rather than the full multimodal pipeline. The SAE basis was pre-trained for LLaMA-3 and not optimized

for embodied control, making results sensitive to fit quality, contrastive pair construction, and intervention layers. Also, we do not provide safety or robustness guarantees. Future work should focus on improving disentanglement (*e.g.,* orthogonalization or feature-subset selection), improving SAE reconstruction [15], extending this technique beyond a simulator to an actual robot, and enabling multi-skill steering.

Conclusion: We introduced a SAE-based steering method to steer robot actions in a multimodal policy that is generated by the Magma model. By leveraging contrastive pairs to isolate action-specific latents and decoding these edits back into the residual stream, we shifted gripper open/close decisions in a controlled way. While unintended co-activations remain a challenge, this work demonstrates that mechanistic interpretability techniques like SAEs can be repurposed for practical policy steering, providing a foundation for more interpretable VLA control.

References

[1] An Intuitive Explanation of Sparse Autoencoders for LLM Interpretability — adamkarvonen.github.io. `https://adamkarvonen.github.io/machine_learning/2024/06/11/sae-intuitions.html`. [Accessed 22-08-2025].

[2] Reza Bayat, Ali Rahimi-Kalahroudi, Mohammad Pezeshki, Sarath Chandar, and Pascal Vincent. Steering large language model activations in sparse spaces. *arXiv preprint arXiv:2503.00177*, 2025.

[3] Amrita Bhattacharjee, Shaona Ghosh, Traian Rebedea, and Christopher Parisien. Towards inference-time category-wise safety steering for large language models. *arXiv preprint arXiv:2410.01174*, 2024.

[4] Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. Saelens. `https://github.com/jbloomAus/SAELens`, 2024.

[5] Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization. *Advances in Neural Information Processing Systems*, 37:49519–49551, 2024.

[6] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

[7] Eoin Farrell, Yeu-Tong Lau, and Arthur Conmy. Applying sparse autoencoders to unlearn knowledge in language models. *arXiv preprint arXiv:2410.19278*, 2024.

[8] Connor Kissane, Robert Krzyzanowski, Joseph Isaac Bloom, Arthur Conmy, and Neel Nanda. Interpreting attention layer outputs with sparse autoencoders. *arXiv preprint arXiv:2406.17759*, 2024.

[9] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[10] Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv preprint arXiv:2408.05147*, 2024.

[11] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11966–11976, 2022.

[12] Kyle O'Brien, David Majercak, Xavier Fernandes, Richard Edgar, Blake Bullwinkel, Jingya Chen, Harsha Nori, Dean Carignan, Eric Horvitz, and Forough Poursabzi-Sangdeh. Steering language model refusal with sparse autoencoders. *arXiv preprint arXiv:2411.11296*, 2024.

[13] Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

[14] Nate Rahn, Pierluca D'Oro, and Marc G Bellemare. Controlling large language model agents with entropic activation steering. *arXiv preprint arXiv:2406.00244*, 2024.

[15] 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.

[16] Tilman Räuker, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. In *2023 ieee conference on secure and trustworthy machine learning (satml)*, pages 464–483. IEEE, 2023.

[17] Daniel Scalena, Gabriele Sarti, and Malvina Nissim. Multi-property steering of large language models with dynamic activation composition. *arXiv preprint arXiv:2406.17563*, 2024.

[18] Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, et al. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*, 2025.

[19] Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. Improving instruction-following in language models through activation steering. *arXiv preprint arXiv:2410.12877*, 2024.

[20] Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

[21] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *arXiv e-prints*, pages arXiv–2308, 2023.

[22] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

[23] Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, et al. Magma: A foundation model for multimodal ai agents. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14203–14214, 2025.

[24] Yu Zhao, Alessio Devoto, Giwon Hong, Xiaotang Du, Aryo Pradipta Gema, Hongru Wang, Xuanli He, Kam-Fai Wong, and Pasquale Minervini. Steering knowledge selection behaviours in llms via sae-based representation engineering. *arXiv preprint arXiv:2410.15999*, 2024.