RoboTerp: Interpreting Vision Language Models for Robotic Control

Ishan Dogra, Safaa Mouline, Henry Tsai, Dhruv Agarwal
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Abstract

Transformer-based models enable robots to interpret a wide range of environments both visually and textually. We examine how fine-tuning vision language models on robot trajectories changes their latent representations of the real world and how it affects their sensitivity to certain actions and objects. We evaluate this by training sparse autoencoders on the PaliGemma VLM model before and after fine-tuning it on 100 episodes from the DROID robot manipulation dataset and comparing the feature distributions. To compare feature quality, we conducted a randomized pairwise trial by human voting and find an absolute advantage of 12.1% for the fine-tuned model scene descriptions over the base model. We also discover that after fine-tuning the model it has an increased sensitivity towards targeted objects and learns new latent features to represent trajectory targets and task types. Our autoencoder achieves a mean feature L1 loss of 0.56 after fine-tuning demonstrating high sparsity. Our code can be found at `https://github.com/smouline/robointerp`.

1 Introduction

Vision-language models (VLMs) are a class of multimodal neural networks that process both visual and text inputs. They can perform many different tasks, such as caption generation and answering questions on images. VLMs have also been extended to the domain of robotics through vision-language action (VLA) models, which map images and instructions directly to actions. One such generalist robot policy model is OpenVLA (1). In this paper we focus on the latent visual representations of the real world that are stored in VLMs and examine how they shift in response to fine-tuning on recorded robot tasks, specifically PaliGemma2 3B-PT (2).

PaliGemma is a family of open-weight vision-language models (VLMs) designed for transfer learning, combining a visual encoder with an autoregressive language model. The vision component is a SigLIP-So400m encoder with a 14px patch size, producing 256, 1024, or 4096 image tokens depending on input resolution (224^2, 448^2, or 896^2). These visual tokens are linearly projected and concatenated with tokenized text inputs before being processed by the Gemma 2 language model in an autoregressive fashion.

In our work, we briefly study this architecture—highlighting how early fusion of visual and textual modalities occurs immediately after the linear projection layer, and how deeper decoder layers progressively refine semantic representations. By attaching activation hooks at multiple decoder layers, we extract and compare the hidden states of both the pretrained PaliGemma2 and a version fine-tuned on our niche robotics image–prompt dataset.

This layer-wise probing allows us to quantify how fine-tuning shifts the model's internal feature distributions. Through sparse autoencoder (SAE) analyses of these activations, we illuminate which stages of the network are most plastic under domain-specific tuning and how semantic information flows through the PaliGemma2 pipeline.

CS 182 2025.

PaliGemma is trained in three stages: (1) joint training of SigLIP and Gemma 2 at 224px^2 on 1B multimodal examples, (2) high-resolution training at 448px^2 and 896px^2, and (3) downstream task-specific fine-tuning. Importantly, no component is frozen during pretraining, enabling deep fusion between modalities. The modular design allows controlled experiments on the effects of model size and image resolution.

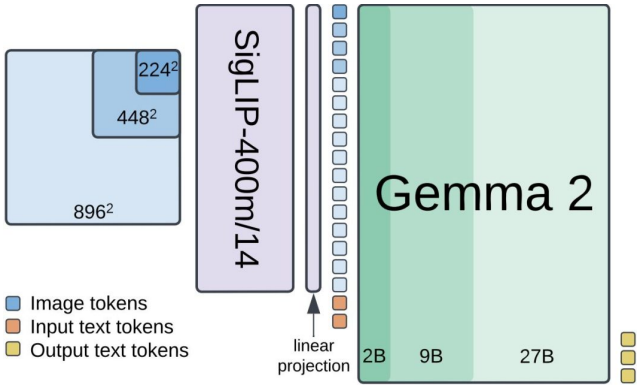

Figure 1: PaliGemma model architecture. The model fuses image and text tokens before passing them into the Gemma decoder.

2 Related Work

Mechanistic interpretability has made substantial progress in recent years, particularly through the use of sparse autoencoders (SAEs) to analyze large language models (LLMs). SAEs are neural networks trained to reconstruct their inputs while encouraging sparsity in their hidden representations. This constraint leads the model to compress information into a smaller number of interpretable, distinct features. In the context of LLMs, this allows researchers to disentangle complex internal activations into a set of latent features that can often be directly interpreted by humans.

A notable example of this approach was demonstrated by Anthropic in their work on the Claude model. In their well-known "Golden Gate Claude" blog, researchers trained SAEs on intermediate activations from Claude and discovered a specific feature that consistently activated when the Golden Gate Bridge was referenced (3). By manipulating this single feature, they were able to amplify or suppress mentions of the Golden Gate Bridge in Claude's outputs—demonstrating that high-level semantic concepts can be isolated and causally intervened upon within the model. These kinds of findings highlight the promise of SAEs for shedding light on how neural networks represent abstract concepts, enabling more transparent and controllable AI systems.

So far, most applications of sparse autoencoders have focused on purely textual models. Recently, there have been efforts to extend interpretability to vision transformers, where features can correspond to object parts, edges, or spatial structures. However, sparse autoencoders have rarely been applied to vision-language models (VLMs), especially those used in robotic control.

Our work brings this interpretability technique to a new setting: we apply sparse autoencoders to PaliGemma, a vision-language model fine-tuned on robot manipulation tasks. This allows us to explore how internal representations shift when a model is adapted from general vision-language understanding to embodied reasoning grounded in real-world robotic tasks. We believe this opens up a promising new direction for combining interpretability with robotics, going beyond what has been done in LLMs or static vision tasks.

3 Preliminaries and discussions

Sparse Autoencoders. A sparse autoencoder (SAE) is an unsupervised neural network that learns compressed representations of input data while enforcing sparsity in the hidden activations (4). The goal is to extract meaningful, disentangled features that can reconstruct the input with minimal information (5).

An SAE consists of two main components: an encoder and a decoder. Given an input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$, the encoder maps it to a hidden representation $\mathbf{z} \in \mathbb{R}^{d_{\text{hidden}}}$ via:

$$\mathbf{z} = f_{\text{enc}}(\mathbf{x}) = \phi(\mathbf{W}_e\mathbf{x} + \mathbf{b}_e),$$

where $\mathbf{W}_e$ and $\mathbf{b}_e$ are the encoder weights and biases, and ϕ is a nonlinearity. The decoder reconstructs the input as:

$$\hat{\mathbf{x}} = f_{\text{dec}}(\mathbf{z}) = \mathbf{W}_d\mathbf{z} + \mathbf{b}_d,$$

where $\mathbf{W}_d$ and $\mathbf{b}_d$ are decoder parameters.

The overall objective minimizes a reconstruction loss with an added sparsity regularizer:

$$\mathcal{L}_{\text{SAE}} = \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 + \lambda \cdot \mathcal{L}_{\text{sparse}}(\mathbf{z}),$$

where λ is a weighting factor and $\mathcal{L}_{\text{sparse}}$ encourages sparse activations.

A common choice for the sparsity penalty, which we use, is the L_1 norm:

$$\mathcal{L}_{\text{sparse}}(\mathbf{z}) = \|\mathbf{z}\|_1 = \sum_i |z_i|,$$

By enforcing sparsity, SAEs learn localized and interpretable features that are useful for downstream analysis such as mechanistic interpretability of neural networks.

4 Method

We apply mechanistic interpretability to vision-language models (VLMs) to understand how specific internal activations change as a result of fine-tuning on language task data. Our goal is to identify how representations within the model evolve when trained on tasks.

As an initial test, we trained first trained SAEs on natural images (ImageNet-1K) to verify that interpretable features emerged in frozen PaliGemma activations. Once this was working well, we proceeded to use a Robotic interaction dataset.

This approach proceeds in three stages:

1. **Baseline analysis**: We first train a sparse autoencoder (SAE) on the frozen activations of the original PaliGemma 2 3B model to identify interpretable features.
2. **Fine-tuning**: We then fine-tune PaliGemma 2 3B on paired vision-language data from the DROID dataset. We focus on Google's **PaliGemma-2-3B**, a multimodal model with approximately 3 billion parameters that accepts both image and text inputs and generates text outputs. We fine-tuned PaliGemma, with 896 by 896 images on language tasks by unfreezing 27 layers in the language decoder, all the language layers that were for attention, as well as the image head bias and kernel values which are used in the multimodal projector. We used 50 episodes from the DROID dataset, which includes episodes of robotic tasks for fine-tuning, of which 20 images were chosen from each episode, all equally spaced with each other. One other variation which we tried was a collage of 6 or 8 images equally-spaced out from a single episode of the DROID dataset, all combined into one image, with the prompt corresponding to the language action described in the dataset for that episode. The input into the PaliGemma model was "describe the task the robot is taking:", and the objective was to fine-tune to the language tasks that are paired with the episodes.
3. **Post-fine-tuning analysis**: Finally, we train a second SAE on the activations of the fine-tuned model.

We use **DROID** (Distributed Robot Interaction Dataset), a robot manipulation dataset containing over 76,000 demonstration trajectories. The dataset spans 564 unique scenes across 52 buildings and includes 86 distinct manipulation tasks such as picking, placing, opening, closing, wiping, and pouring. It was collected on a hardware setup featuring a 7-DoF Franka Emika Panda robot arm and multiple RGB cameras. Each trajectory includes synchronized RGB video, depth images, camera calibration parameters, and natural language instructions.

In our study, we use a curated subset of 100 episodes in the RLDS format from diverse environments. This subset enables us to investigate how fine-tuning vision-language models on action-language pairs influences their internal representations, using sparse autoencoders for mechanistic analysis.

To evaluate our autoencoders, we use **sparsity loss**, specifically the L_1 norm. Sparsity loss is a regularization term used to encourage sparse activations in neural networks, particularly in autoencoders. It penalizes the number or magnitude of non-zero activations in the hidden layer, promoting representations where only a small subset of units are active at a time. This helps disentangle features and improves interpretability by isolating distinct concepts across neurons. In the context of mechanistic interpretability, sparsity loss serves as a useful metric for evaluating how well the learned features align with meaningful, localized activations. We also evaluate it using a randomized pairwise trial by humans for determining which set of images had the highest activation for those features.

By comparing the SAE features before and after fine-tuning, we aim to isolate how the internal representations of the model adapt to embodied reasoning tasks. This can reveal which features are reused, modified, or newly learned as a result of language-task training.

5 Experiments

We study how fine-tuning a vision-language model (VLM) on robot demonstration data affects its internal representations by training sparse autoencoders (SAEs) on hidden activations extracted from both the pretrained and fine-tuned models. Specifically, we examine whether new features emerge after fine-tuning, and whether the model becomes more sensitive to task-relevant elements in the visual input.

5.1 Overview

First, we will test SAEs on the Vision Transformer in PaliGemma with the natural images in ImageNet. Once, we prove that we can see that the top activating images from the top features in the SAE have same coherent similarity, we will then perform this on a robotics dataset.

Recall our robotic interpretibility experimental pipeline follows three main steps:

1. **Baseline Analysis:** Train an SAE on the frozen activations of the pretrained PaliGemma 2 3B model to establish a baseline representation.
2. **Fine-tuning:** Fine-tune PaliGemma on paired robot images and action-language captions from the DROID dataset, with several configurations explored.
3. **Post fine-tuning Analysis:** Train a second SAE on the activations from the fine-tuned model and compare its learned features to those of the baseline.

5.2 Setup

Compute We use an A2 virtual machine instance from Google Cloud Platform containing eight NVIDIA A100 GPUs with 40GB memory and CUDA 12.8 on Ubuntu 22.04. We originally used two smaller instances containing one and two A100 GPUs respectively. However, we found that we ran out of memory often and that slow training jobs required more compute to reasonably complete.

Data preparation We write several helper functions to help organize the DROID dataset. Specifically, we sample individual still frames from the recordings to pass as images into PaliGemma. In addition, we combine these images in ordered sequences and format into the correct input dimensions.

5.3 SAE Testing on ImageNet

To test the base PaliGemma model, we collected activations from running inference on two curated datasets: ImageNet-1k (6) and DROID. We wrote our own sparse autoencoder class and tested it on activations from a 5,000 image sample of ImageNet. We conducted 114 training runs, of which 37 crashed (32.4%) due to GPU memory exhaustion, kernel failures, and other compute difficulties.

In Table 1 below we report on a subset of our ImageNet training runs. We present runs that had a diversity of hyperparameter values, ceteris paribus, rather than just the runs that minimized loss to explain our conjectures on how these hyperparameters affect learning.

We would expect loss to increase the most drastically when sparsity weight increases, since sparsity weight is a penalty. While our lowest loss run did have the lowest sparsity weight, the second

Learning rate	Sparsity weight	Hidden multiplier	Epochs	Batch size	Loss
$5 \cdot 10^{-2}$	$2 \cdot 10^{-03}$	16	50	256	2.568
$1 \cdot 10^{-02}$	$5 \cdot 10^{-02}$	32	200	64	1.604
$1 \cdot 10^{-03}$	$5 \cdot 10^{-03}$	16	15	256	1.016
$1 \cdot 10^{-04}$	$1 \cdot 10^{-02}$	8	50	128	0.425
$1 \cdot 10^{-03}$	$5 \cdot 10^{-03}$	**16**	**100**	**256**	**0.079**

Table 1: Table displaying the mean z sparsity for SAEs trained with five different hyperparameter combinations on base PaLI-Gemma and 5k images from ImageNet-1K.

lowest loss run had the second highest sparsity weight. This points at the importance of tuning other hyperparameters, such as the number of epochs.

We train to convergence for all runs, and we note that the number of epochs required varies based on learning rate and sparsity weight. Increasing learning rate decreases required epochs while increasing sparsity weight increases required epochs, but this is not guaranteed. One interesting note is that the first and third runs in Table 1 have the same hidden multiplier and batch size, but the third has a lower learning rate and higher sparsity weight. We would expect this to increase the number of epochs needed, but in fact the third run achieves a 60.4% lower loss in less than a third the number of epochs as the first run (15 versus 50). This highlights the stochastic nature of training sparse autoencoders and complex interactions between hyperparameters. We show below the loss curve for training one of these ImageNet autoencoders in Figure 2.

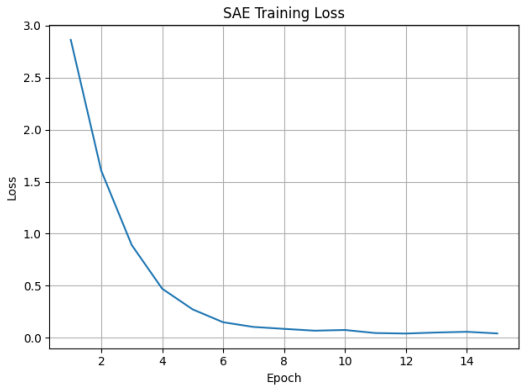

Figure 2: Training loss over 15 epochs for our SAE.

To visually and intuitively evaluate our autoencoders, we display the five images with the highest activations for each feature. We did this for the top five features after every successful training run and we display some examples below.

5.4 Dataset Variants

We curated multiple dataset variants from the DROID corpus to probe the impact of visual diversity and temporal granularity:

- **100 episodes, 20 frames each:** Frames evenly sampled across each episode.
- **100 episodes, 4 frames each:** A smaller subsample for faster experimentation.
- **10 episodes, all frames:** High-resolution frame coverage across a limited number of episodes (1,906 frames total).
- **100 episode collages:** 6 equally spaced frames per episode composited into a single 896×896 image.

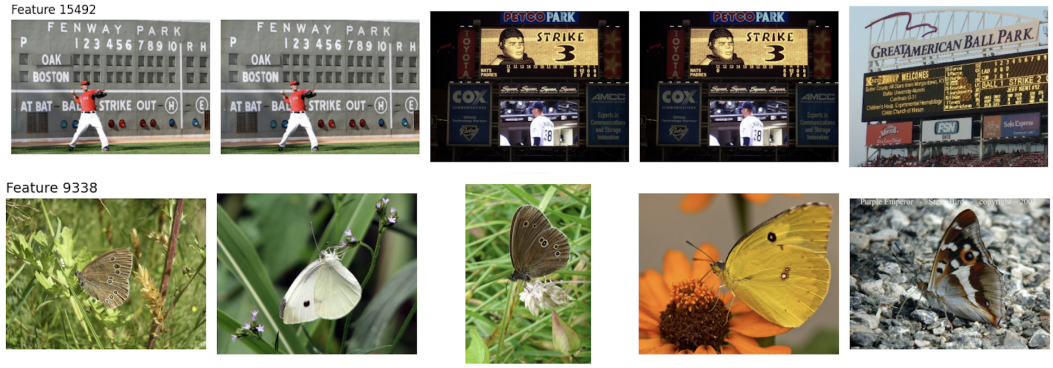

Figure 3: Examples of clearly human-identifiable features from different training runs of SAEs; the subjects are baseball and butterflies. We see repeated images due to the sampling with replacement from ImageNet. We do not specify or filter classes at all and sample at random from the 1,000 classes.

Figure 4: Example of unclear feature from an SAE training run.

5.5 Models Under Study

1. **PaliGemma 2 3B (frozen):** The original pretrained vision-language model, used to extract baseline activations. We evaluated two variants based on input resolution: 224×224 and 896×896 images.
2. **RoboVLM (frame-based fine-tuning):** The same PaliGemma 2 3B architecture fine-tuned on individual frames from the DROID dataset paired with corresponding language captions. This setup preserves the temporal sequence by using spaced-out frames and aims to ground the model in task-specific robotic behavior.
3. **RoboVLM (collage-based fine-tuning):** A variant fine-tuned on composited images formed by collaging 6 equally spaced frames from each episode into a single 896×896 input. The model is trained to generate the episode's caption from this visual summary.

5.6 Finetuning Analysis

Although the collage-based fine-tuning (example given in Figure 4) provided an efficient way to condense temporal context into a single input, it often led to degenerate language outputs. This is likely due to over compression of scene information, which overwhelms the spatial resolution of PaliGemma's SigLIP-based image encoder. Because the encoder divides the image into non-overlapping patches, collaging frames causes some patches, and thus the tokens they produce, to span across the boundaries of separate images. These patch collisions introduce visual ambiguity, breaking spatial coherence and degrading the clarity of frame-level cues. The resulting visual tokens confuse the model's cross-modal attention layers, impairing its ability to generate grounded, coherent text. Thus, we mainly focus on the frame-based fine-tuned model.

The frame-based fine-tuning methodology focuses on being able to predict the action that the robot is going to take given a prompt about having to describe the action the robot is going to take and the image of the scene. After fine-tuning the language layers and the multimodal layer, the fine-tuned model showed improvements over the base model in terms of image caption generation, as shown

Figure 5: Example of collated image

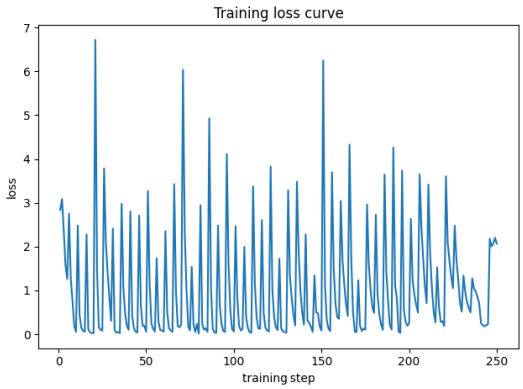

Figure 6: Training Loss for Fine-Tuned Model

by the results of our randomized pairwise trials, where Table 3 shows that the finetuned model was chosen more often than the baseline model. We provide some samples of the baseline and fine-tuned good and bad examples in Figure 6. Additionally, the training curve loss can be found in Figure 5, which shows that the loss is not improving after the first couple of epochs. However, in those epochs, the model is able to improve relative to the generic PaliGemma model, as shown in the results.

5.7 Layer Probing and Hooks

To analyze the internal representations learned by PaliGemma, we inserted forward hooks at several key locations within the model architecture. These hooks allow us to extract hidden activations for training sparse autoencoders (SAEs). Specifically, we experimented with:

- **Vision encoder MLP layers (e.g., layer 20 and 24):** These layers capture high-level visual abstractions, such as object shapes and spatial layouts. We chose deeper layers to probe richer visual features after substantial processing through the SigLIP backbone.
- `multi_modal_projector.linear` (fusion layer): This layer projects image embeddings into the language model's input space, effectively bridging the vision and language modalities. Since our fine-tuning begins at this layer, it serves as a natural comparison point for observing how visual-language fusion representations shift after training.

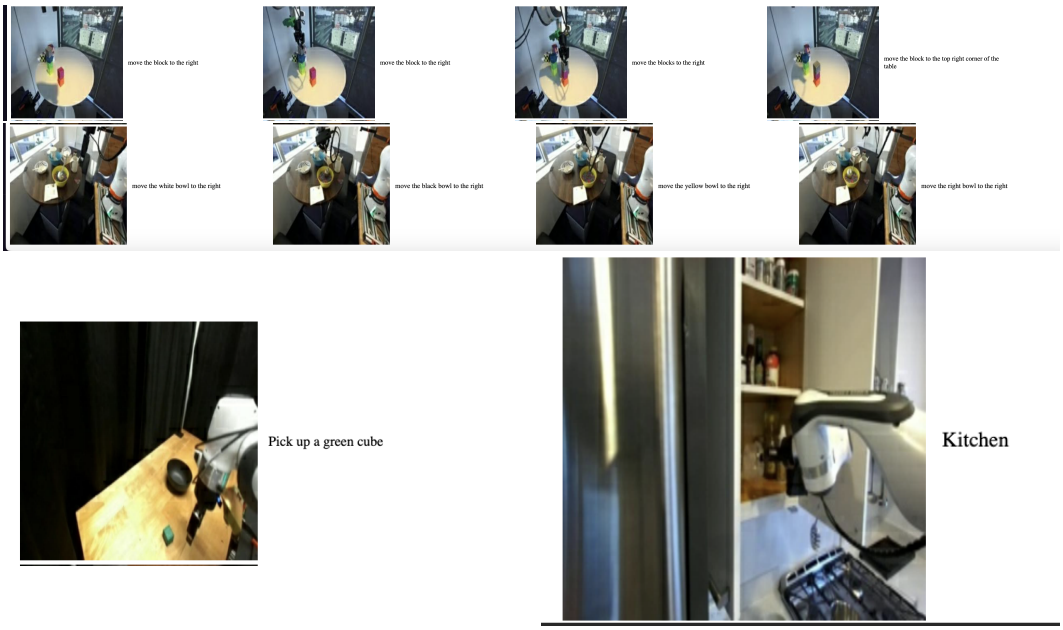

Figure 7: Examples of good and bad baseline and finetuned images and episodes. Top 4 images demonstrates a good example of actions for one episode of the finetuned model, while second row of 4 images show bad examples. The bottom left image demonstrates a good example from the baseline model, and the bottom right image shows a bad example from the baseline model.

- **Language decoder MLP layers (e.g., layer 12):** These deeper language layers reflect semantic reasoning and decision-making processes downstream of the vision-language fusion. However, they are influenced more heavily by text decoding, making them less directly comparable to purely visual features.

Among these, the `multi_modal_projector` layer provides an most interpretable comparison between the frozen and fine-tuned models, since it marks the first trainable point in our fine-tuning setup. Probing downstream language layers also reveal how the model reorganizes information post-fusion.

5.8 SAE on DROID

To examine whether sparse autoencoders (SAEs) could identify semantically coherent visual features in a robotics context, we trained an SAE on a curated DROID subset containing all frames from 10 episodes (1,906 images total). This subset provided diverse viewpoints and temporal slices of robotic activity while remaining compact enough for rapid experimentation. We used the same vision layer we hooked in the ImageNet SAE. Note that this layer was frozen in finetuning, so it would not change representations either way.

The autoencoder was trained using the hyperparameters in Table2. This configuration resulted in robust sparse features. One particularly interpretable feature consistently fired in the presence of markers on the robot arm or workspace.

Hyperparameter	Value
hidden_multiplier	8
learning_rate	1×10^{-4}
sparsity_weight	1×10^{-2}
n_epochs	50
batch_size	512

Table 2: Sparse Autoencoder Training Hyperparameters

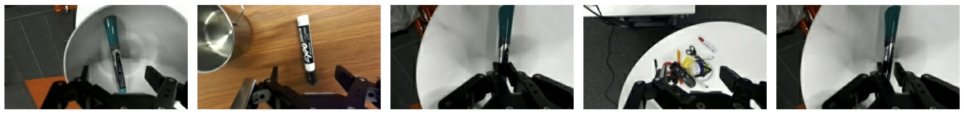

Figure 8: Images that maximally activate a specific SAE feature. Each image contains a marker that is about to be grasped.

Additionally, we found that multiple top-activating images for certain features came from the same episode, despite appearing at slightly different timesteps. This suggests that perhaps SAEs can learn temporally consistent concepts even when trained without explicit temporal structure. However, more realistically, it just means that these images are so similar, often differing only by slight camera motion or small changes in the robot's position, that they activate the same feature due to low-level visual redundancy, rather than the model learning a truly temporal abstraction. Ideally, we would want more features that have semantic, task, or gripper similarity (such as the previous example)–not just images that look alike.

Figure 9: Images from the same DROID episode that activate a single feature.

These results validate that sparse autoencoders can extract meaningful, reusable concepts from robot video data.

5.9 Human feature evaluation

We conduct a randomized pairwise trial to evaluate the performance of our fine-tuned model at describing still frames of robot action recordings versus the base model. We run inference on both models prompted to generate captions for a collection of still frames. We create a simple user interface for humans to judge captions. The user is presented a random image with two captions underneath, one from each model, and asked to vote on which caption most accurately describes the still. The user cannot see which model generated each caption and we randomize the order of captions presented. We collected 100 votes total from two human subjects and present the results in Table 3 below.

Vote	Count	Percentage (%)
Fine-tuned	46	46.0
Baseline	41	41.0
Can't Tell	13	13.0

Table 3: Summary of user votes comparing captions from the fine-tuned and baseline models.

The fine-tuned model received 5% more votes and a 12.1% absolute advantage compared to the base model. We use the same randomized pairwise trial methodology to evaluate the contextual interpretability of vision features from the autoencoder trained on both the fine-tuned and base models on DROID dataset frames. We collected 6 votes total from one human subject due to setup constraints and present the results in Table 4 below. The unanimous decision is that finetuned features are more cohesive and interpretable, but a larger sample size would better support that conclusion.

We also evaluate the sparsity of our autoencoders using L1 loss (mean absolute error). Sparse codes reduce overfitting and encourage efficient representations analogous to neurons, thereby yielding compressed features that generalize across downstream tasks and lower error. Specifically, we compare the distribution of L1 loss across features for multiple autoencoders, where an inverse power

Vote	Count	Percentage (%)
Fine-tuned	2	25.0
Baseline	0	0.0
Can't Tell	6	75.0

Table 4: Summary of user votes comparing captions from the fine-tuned and baseline models.

law and very low median represents higher sparsity. We show below two histograms of sparsity distribution. The first is a training run on ImageNet that yielded few interpretable features. Note that although there is a right skew the distribution is still bell shaped and therefore the features are not very sparse. The second is a final training run on DROID. It exhibits the inverse power law distribution we are seeking for sparse features.

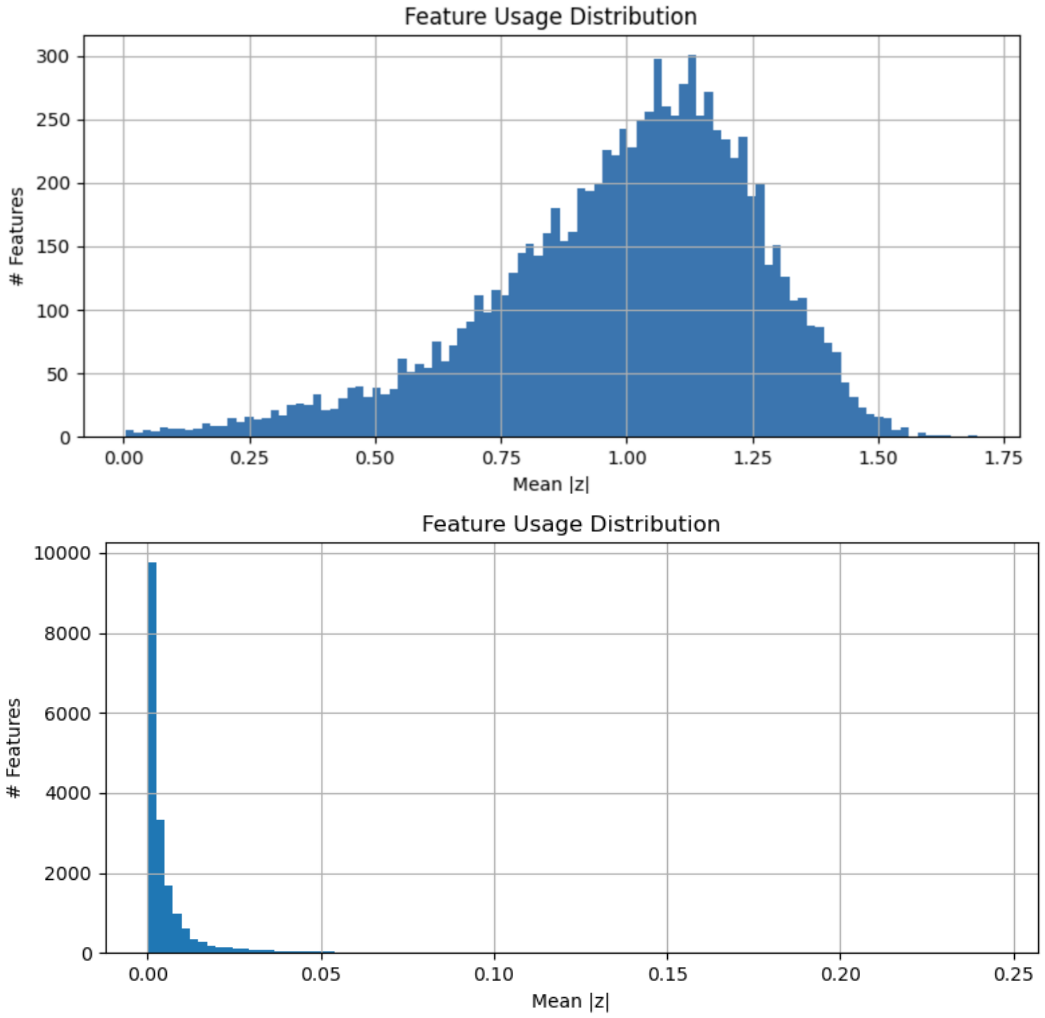

Figure 10: Comparison of feature L1 loss (MAE) distributions for ImageNet (top) and the DROID dataset (bottom)

6 Conclusion

We present RoboVLM, a fine-tune of PaliGemma 2 3B-PT on robot action recordings from the DROID dataset. RoboVLM exhibits new latent features that better capture robot intentions and

environment action targets. We also present RoboTerp, a collection of sparse autoencoders trained on RoboVLM and the base PaliGemma model using subsets of ImageNet and DROID. Finally, we provide a human-blinded comparison tool and the results gathered from using it to evaluate image captions and sparse features.

RoboVLM exhibits a 12.1% absolute advantage over base PaliGemma in human blind comparisons for image captions and also demonstrates strength in its feature interpretability. We also show that our autoencoders are able to compress features into a significantly smaller latent space by showing the feature L1 loss distribution with an inverse power law shape after training to convergence on the DROID subset.

Future directions of this work include testing the larger variations of PaliGemma, such as the 27 billion parameter version, as well as other VLMs, since these may provide a better baselines for building on. Additionally, we would like to pass in multiple images into the PaliGemma model for fine-tuning. Although our approach of running inference on collated combinations of scene stills did not perform well, other variations of this could include passing in different time steps of the episode into the NLVR2 variant of PaliGemma, which is able to handle multiple images through temporal locality. Additionally, DROID contains three different angles of the image scene, of which we are currently only using one. These could also be passed into the NLVR2 variant to provide more context to the model regarding the scene.

Acknowledgments and Disclosure of Funding

We would like to thank our advisors Colin Li and Naman Jain for their advice during office hours on mechanistic interpretability. Thank you to David Chanin for advice on using the SAELens open-source library, even though we did not end up using it. Thank you to Berkeley Wireless Research Center for compute resources. We would also like to thank the undergraduate upper division lounge in Soda Hall for hosting our project work sessions.

References

[1] M. J. K. et al., "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024. [Online]. Available: https://arxiv.org/pdf/2406.09246

[2] A. Steiner, A. S. Pinto, M. Tschannen, D. Keysers, X. Wang, Y. Bitton, A. Gritsenko, M. Minderer, A. Sherbondy, S. Long, S. Qin, R. Ingle, E. Bugliarello, S. Kazemzadeh, T. Mesnard, I. Alabdulmohsin, L. Beyer, and X. Zhai, "Paligemma 2: A family of versatile vlms for transfer," 2024. [Online]. Available: https://arxiv.org/abs/2412.03555

[3] Anthropic, "Golden gate claude," https://www.anthropic.com/news/golden-gate-claude, accessed: April 22, 2025.

[4] A. Karvonen, "Sparse autoencoder intuitions," https://adamkarvonen.github.io/machine_learning/2024/06/11/sae-intuitions.html, accessed: April 22, 2025.

[5] EleutherAI, "Delphi," https://github.com/EleutherAI/delphi, accessed: April 22, 2025.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255.

[7] OpenAI, "Chatgpt," https://chat.openai.com, large language model. Accessed: April 22, 2025.

[8] J. Bloom, "Saelens," https://jbloomaus.github.io/SAELens/, accessed: April 22, 2025.

[9] Google, "Gemma 3: Open models for responsible ai development," https://blog.google/technology/developers/gemma-3/, accessed: April 22, 2025.

[10] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu,

J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, V. Guizilini, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn, "Droid: A large-scale in-the-wild robot manipulation dataset," 2025. [Online]. Available: https://arxiv.org/abs/2403.12945

Appendix

- GitHub Repo Link:
  https://github.com/smouline/robointerp
- Fine-Tuned RoboVLM Checkpoint:
  https://tinyurl.com/PaliGemma-checkpoint