

---

# Rank-1 LoRAs Encode Interpretable Reasoning Signals

---

**Jake Ward**  
MATS  
jakenicholasward@gmail.com

**Paul Riechers<sup>†</sup>**  
Simplex

**Adam Shai<sup>†</sup>**  
Simplex

## Abstract

Reasoning models leverage inference-time compute to significantly enhance the performance of language models on difficult logical tasks, and have become a dominating paradigm in frontier LLMs. Despite their wide adoption, the mechanisms underpinning the enhanced performance of these reasoning models are not well understood. In this work, we show that the majority of new capabilities in reasoning models can be elicited by small, single-rank changes to base model parameters, with many of these changes being interpretable. Specifically, we use a rank-1 LoRA to create a minimal parameter adapter for Qwen-2.5-32B-Instruct which recovers 73-90% of reasoning-benchmark performance compared to a full-parameter finetune. We find that the activations of this LoRA are as interpretable as MLP neurons, and fire for reasoning-specific behaviors. Finally, we train a sparse autoencoder on the entire activation state of this LoRA and identify fine-grained and monosemantic features. Our findings highlight that reasoning performance can arise largely from minimal changes to base model parameters, and explore what these changes affect. More broadly, our work shows that parameter-efficient training methods can be used as a targeted lens for uncovering fundamental insights about language model behavior and dynamics.

## 1 Introduction

Current frontier LLMs increasingly rely on chain-of-thought (CoT) reasoning to achieve strong performance on logical tasks [1, 7, 14]. Despite their ubiquity, we still lack a crisp, white-box understanding of the mechanisms inside the network which enable these gains. Some attempts have been made to mechanistically interpret fully finetuned reasoning models [2, 12, 13]. However, reasoning model interpretation presents a fundamental challenge: the parameters responsible for new reasoning behaviors in a finetuned model are many and differences are diffuse [11]. In this paper, we introduce an alternative approach: we use parameter-efficient methods to explicitly enforce that differences between base and finetuned models in parameter space are minimal, allowing us to perform focused and targeted interpretability experiments.

We show that a rank-1 LoRA [6] trained to adapt all layers of Qwen-2.5-32B-Instruct on a dataset of DeepSeek R1 rollouts is enough to recover 73-90% of the performance gap on reasoning benchmarks, compared to a full-parameter finetune (Table 1).

We directly interpret the directions defined by this LoRA. Because each adapted matrix is rank-1, activations of each adapter component can be represented by a single scalar. In total, our LoRA encodes 192 MLP and 256 attention adapter components across model layers and weight matrices,

---

<sup>†</sup>Equal senior author contribution.

Code for this paper is available at <https://github.com/jnward/reasoning-lora-interp>

Benchmark	Base Qwen2.5-32B-Instruct	Rank-1 LoRA	Full Finetune	% Recovery
AIME'24 (no-figures)	0.2333	0.5000	0.6000	72.73%
MATH500	0.8340	0.9100	0.9220	86.36%
GPQA-Diamond	0.4899	0.5808	0.5909	89.90%

Table 1: Performance of base model, rank-1 LoRA, and full-parameter finetune on three reasoning benchmarks. % Recovery is the fraction of the difference in performance between the base model and full finetune which is recovered by the rank-1 LoRA.

with each adapter component providing a single activation at each token position. This allows us to treat the LoRA itself as a measurement device: we find that individual adapter directions have interpretable properties comparable to those of MLP neurons, and identify monosemantic concepts encoded by these directions. Additionally, we take the entire 448-dimensional LoRA activation state, representing the entire adapter state across MLP and attention components, and train a cross-layer SAE [8]. This SAE uncovers sparse and monosemantic features which organize into categories such as *Mathematical Operators*, *Procedural Markers*, and *Discourse and Reasoning Markers* (Figure 2). Finally, perform a component-wise ablation study, and find that MLP adapters drive the majority of this change (Figure 3).

Taken together, these results show that minimal adapters both *elicit* and *expose* reasoning signals: a rank-1 LoRA is sufficient to recover the majority of reasoning performance, and yields interpretable adapter directions.

Our contributions are as follows:

- We train and open-source a rank-1 LoRA which adapts all projection matrices in Qwen-2.5-32B-Instruct. We demonstrate that this LoRA recovers 73-90% of reasoning benchmark performance compared to a full-parameter finetune.
- We show that individual adapter activations are as monosemantic as MLP neurons, and interpret these.
- We decompose the complete LoRA activation state into interpretable and monosemantic features using a cross-layer SAE.

## 2 Preliminaries

**LoRA training** We finetune Qwen-2.5-32B-Instruct using `slk-1.1`, a sample-efficient dataset of 1000 DeepSeek R1 chain-of-thought trajectories and answer attempts on diverse reasoning problems [10]. For all experiments, we train on this dataset for 5 epochs using cross-entropy loss on a single 8xH200 node. We train our LoRA to adapt all three MLP matrices at every layer (`up_proj`, `down_proj`, and `gate_proj`), as well as all four Q, K, V, and O attention matrices at every layer. In total, our LoRA contains less than 0.03% as many trainable parameters as the base model.

**Extracting LoRA activations** For every  $N \times M$  adapted matrix in the base model, a rank- $r$  LoRA encodes an  $N \times r$  `lora_A` matrix and an  $r \times M$  `lora_B` matrix. Because  $r = 1$  for our LoRA, `lora_A` and `lora_B` are  $N$ - and  $M$ -dimensional vectors respectively. We can extract scalar activations for every LoRA component by taking the activation value between `lora_A` and `lora_B` during the forward pass, which is equivalent to the projection of `lora_A` onto the input activation vector to the adapted matrix.

## 3 LoRA Analysis

### 3.1 Interpreting LoRA Components

We interpret LoRA activations using two different methods: by treating individual adapter activations as probes, and separately by training an SAE on the entire 448-dimensional adapter activation vector. In both cases, we generate activations over the entire training set and extract max-activating examples with associated contexts. We then use LLM autointerpretation to generate interpretations from the

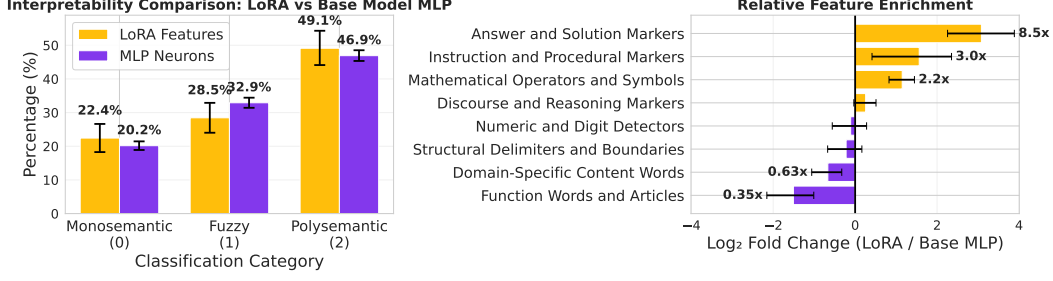


Figure 1: **(left)**: Comparison of interpretability scores of individual LoRA adapter activations to arbitrarily sampled MLP neurons, and find that LoRA activations tend to be monosemantic roughly as often. **(right)**: Comparison of autointerpretation categories between MLP neurons and LoRA adapter activations. LoRA adapters tend to activate more often for reasoning-specific feature categories.

top 64 contexts with token + activation pairs [3], use the same LLM to classify the monosemanticity of features, and finally categorize each feature such that we can compare and examine aggregate feature distributions. Categories were generated by prompting an LLM with feature interpretations and examples from an equal number of MLP neurons and LoRA directions. The exact prompts used are in Appendix B.

**Direction-level interpretation** We run our interpretation and feature classification pipeline both on individual LoRA activations and on the first 60 neurons of each MLP in the unadapted base model. This gives us a baseline to compare feature distributions to, allowing us to examine which feature categories the LoRA activates on compared to the baseline feature distribution in the dataset (as measured by MLP activations). We find that LoRA activations have roughly the same likelihood as MLP neurons to be monosemantic, but tend to encode different feature categories. We interpret this level of monosemanticity as a positive result for LoRA interpretation, given significant recent work utilizing MLP neurons for pragmatic interpretability tasks [5]. Relative to MLP neurons, LoRA activations are more likely to fire for text corresponding to answers or solutions, problem instructions, mathematical symbols, and reasoning discourse (Figure 1).

**Cross-layer SAE over entire adapter state** To gain a more fine-grained and comprehensive view of features encoded by our LoRA, we train a cross-layer sparse autoencoder (SAE) on the entire 448-dimensional LoRA activation state. We use a batch-top-k SAE with  $k=16$ , and an expansion

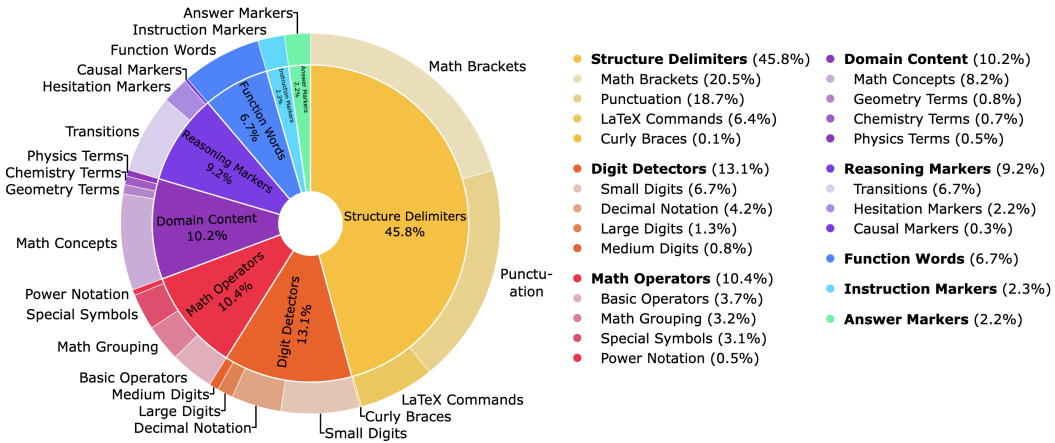


Figure 2: Overview of feature categories learned by an SAE trained on LoRA activation states. Percentages indicate relative feature activation densities. Categories and subcategories were generated by prompting an LLM with feature descriptions.

factor of 8 [4, 9]. After filtering for dead latents, the trained SAE contains roughly 2000 features. We run the same automated interpretation and classification pipeline on max-activating examples for each feature in this SAE, and then further categorize each feature with LLM-generated subcategories. We then visualize relative feature prevalence by computing category-wise activation densities, which is the normalized sum of feature activations within each category over a subset of training data, and plot these in figure 2. We note that this categorization methodology is subjective and sensitive to prompt verbiage, and provide this visualization only as a tool to develop high-level intuitions about the adapter’s representational budget across layers and components. Broadly, we observe that features tend to concentrate on mathematical operators and syntax, numbers, formatting tokens, and reasoning control-flow. Our LLM-based interpretability classification identifies 62% of SAE features as “cleanly monosemantic”, up from 22% of LoRA features, with an additional 22% of SAE features classified as “broad but consistent”. For a more direct look into our SAE, a selection of individual SAE feature activations and interpretations are provided in appendix A.3.

### 3.2 Ablation Study

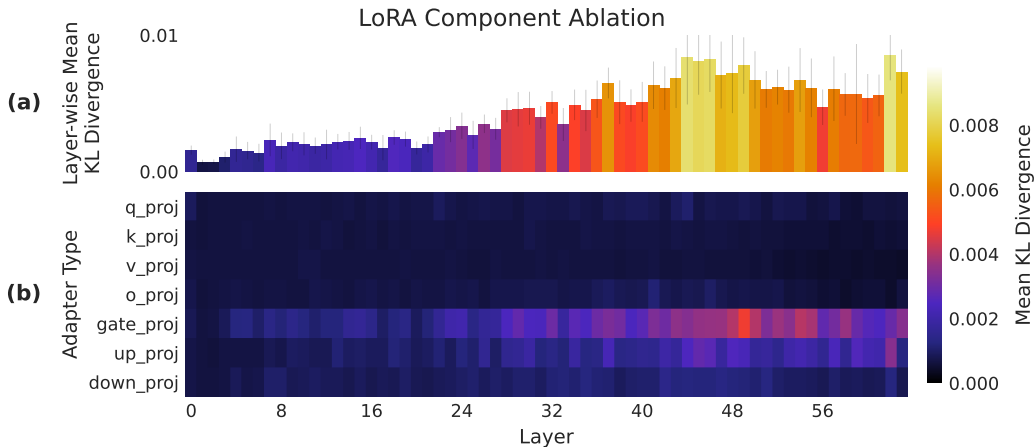


Figure 3: Effect of ablating individual LoRA components from full adapter. **(a)** Effect of ablating all adapter components at a given layer on KL divergence. **(b)** Effect of ablating each adapter component individually. Ablation at later layers tends to have a significantly greater effect on the model’s output distribution compared to earlier layers. MLP adapters, especially those trained on `gate_proj` matrices tend to have the strongest effect.

To identify which LoRA components most affect downstream performance, we conduct two ablation experiments. First, we zero out individual components and layers to measure their impact on output KL divergence relative to the unmodified LoRA. Second, we ablate all MLP components simultaneously, and separately all attention components, to assess benchmark performance impact.

Individual component ablation (Figure 3) reveals that mid-to-late layers (particularly 44, 45, 46, and 62) have the greatest effect on downstream KL. MLP components show significantly larger impact than attention components, with `gate_proj` having the strongest average effect.

Simultaneous ablation results (Table 2) confirm MLP adapters’ greater contribution: removing all attention adapters decreases performance but still outperforms the base model, while removing all MLP adapters causes severe degradation, underperforming the base model on one of three tasks and showing poor performance on the others.

## 4 Conclusion and Outlook

**Limitations** This work includes several limitations. First, we train our LoRA using a sample-efficient dataset with roughly 10 million tokens. While significant performance can be elicited using this dataset [10], it is likely that our trained models do not contain all of the circuits which are learned

by models trained on larger datasets. Additionally, we attempted to prove that extracted LoRA directions have a causal effect on model outputs when used for steering, but found inconclusive results. Anecdotally, steering experiments required very large steering magnitudes to have noticeable effects (in excess of 50x normal activation magnitudes), at which point the model would have a high propensity for backtracking (outputting the `Wait,` token, among others). More investigation of this phenomenon is required to make claims about what is going on. We also heavily use LLM-based autointerpretation and autocategorization methods in our analysis. These methods are useful for painting a general picture of model mechanisms, but we suspect they often fail to uncover the “true” role of extracted features. Finally, we only study one model, Qwen-2.5-32B-Instruct, and evaluate this model mostly on math-related reasoning benchmarks.

We demonstrate that reasoning capabilities in large language models can be substantially recovered through minimal parameter modifications, with a rank-1 LoRA recovering 73-90% of full finetuning performance while having only  $\sim 0.03\%$  as many trainable parameters. Our analysis reveals that these minimal parameter changes encode interpretable, reasoning-specific signals, with individual adapter directions exhibiting monosemantic properties. Further, we find that sparse autoencoders are useful for extracting additional monosemantic features from LoRA activation states. These findings open new avenues for understanding LLMs through parameter-efficient methods, and suggest that similar targeted approaches could be useful to study other emergent capabilities. Future work could involve focused analysis identifying specific circuits that LoRAs interface with, which we hope could illuminate the core computational mechanisms underlying reasoning behavior in language models.

## 5 Acknowledgments

This work was completed during the ML Alignment and Theory (MATS) 8.0 program. JW thanks Elizabeth Donoway for useful discussions at the beginning of this project, and Bryce Woodworth for support throughout.

## References

- [1] Openai o1 system card. Technical report, OpenAI, December 2024. URL <https://cdn.openai.com/o1-system-card-20241205.pdf>.
- [2] David D. Baek and Max Tegmark. Towards understanding distilled reasoning models: A representational approach. *arXiv preprint arXiv:2503.03730*, 2025. URL <https://arxiv.org/abs/2503.03730>. ICLR 2025 Building Trust Workshop (paper).
- [3] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. <https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html>, 2023.
- [4] Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024. doi: 10.48550/arXiv.2412.06410. URL <https://arxiv.org/abs/2412.06410>.
- [5] Dami Choi, Vincent Huang, Kevin Meng, Daniel D. Johnson, Jacob Steinhardt, and Sarah Schwettmann. Scaling automatic neuron description. <https://transluce.org/neuron-descriptions>, October 2024. Published October 23, 2024. Accessed August 22, 2025.
- [6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2022. URL <https://arxiv.org/abs/2106.09685>.
- [7] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022. URL <https://arxiv.org/abs/2205.11916>.
- [8] Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing. <https://transformer-circuits.pub/2024/crosscoders/>, 2024. Transformer Circuits Thread.

- [9] Julian Minder, Clément Dumas, Caden Juang, Bilal Chughtai, and Neel Nanda. Overcoming sparsity artifacts in crosscoders to interpret chat-tuning. *arXiv preprint arXiv:2504.02922*, 2025. doi: 10.48550/arXiv.2504.02922. URL <https://arxiv.org/abs/2504.02922>.
- [10] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. sl: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025. URL <https://arxiv.org/abs/2501.19393>.
- [11] Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. *arXiv preprint arXiv:2505.11711*, 2025. doi: 10.48550/arXiv.2505.11711. URL <https://arxiv.org/abs/2505.11711>.
- [12] Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Understanding reasoning in thinking language models via steering vectors. *arXiv preprint arXiv:2506.18167*, 2025. doi: 10.48550/arXiv.2506.18167. URL <https://arxiv.org/abs/2506.18167>. ICLR 2025 Workshop on Reasoning and Planning for LLMs (paper).
- [13] Jake Ward, Chuqiao Lin, Constantin Venhoff, and Neel Nanda. Reasoning-finetuning repurposes latent representations in base models. *arXiv preprint arXiv:2507.12638*, 2025. doi: 10.48550/arXiv.2507.12638. URL <https://arxiv.org/abs/2507.12638>. ICML 2025 Workshop on Actionable Interpretability.
- [14] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H. Chi, Quoc V. Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022. URL <https://arxiv.org/abs/2201.11903>.

## A Appendix

### A.1 Model Weights

Weights for the trained LoRA are available at <https://huggingface.co/jnward2/qwen-reasoning-lora>.

### A.2 Selected LoRA Directions

We present a cherry-picked selection of interesting and interpretable LoRA directions, visualized with a dashboard. The full dashboard is available at [https://jakeward.page/lora\\_sae\\_dashboard.html](https://jakeward.page/lora_sae_dashboard.html). On the left of the dashboard are max-activating examples of the direction over the training dataset. On the right, a full training sample is shown with activations highlighted. Highlight color represents whether the activation is positive or negative.



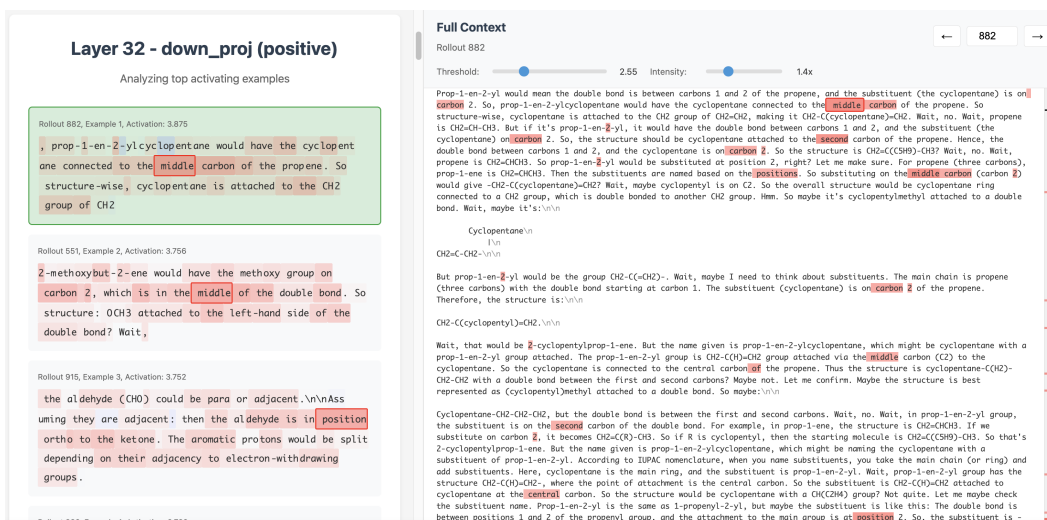


Figure 6: Autointerpretation: indicates positional numbering in molecules (position, numeral, middle)  
Note: a thresholding operation was applied to make visualizing top activations in the full context easier.

### A.3 Selected SAE Features

We additionally present a cherry-picked selection of SAE features. The dashboard screenshots highlight max-activating feature examples over the training dataset.



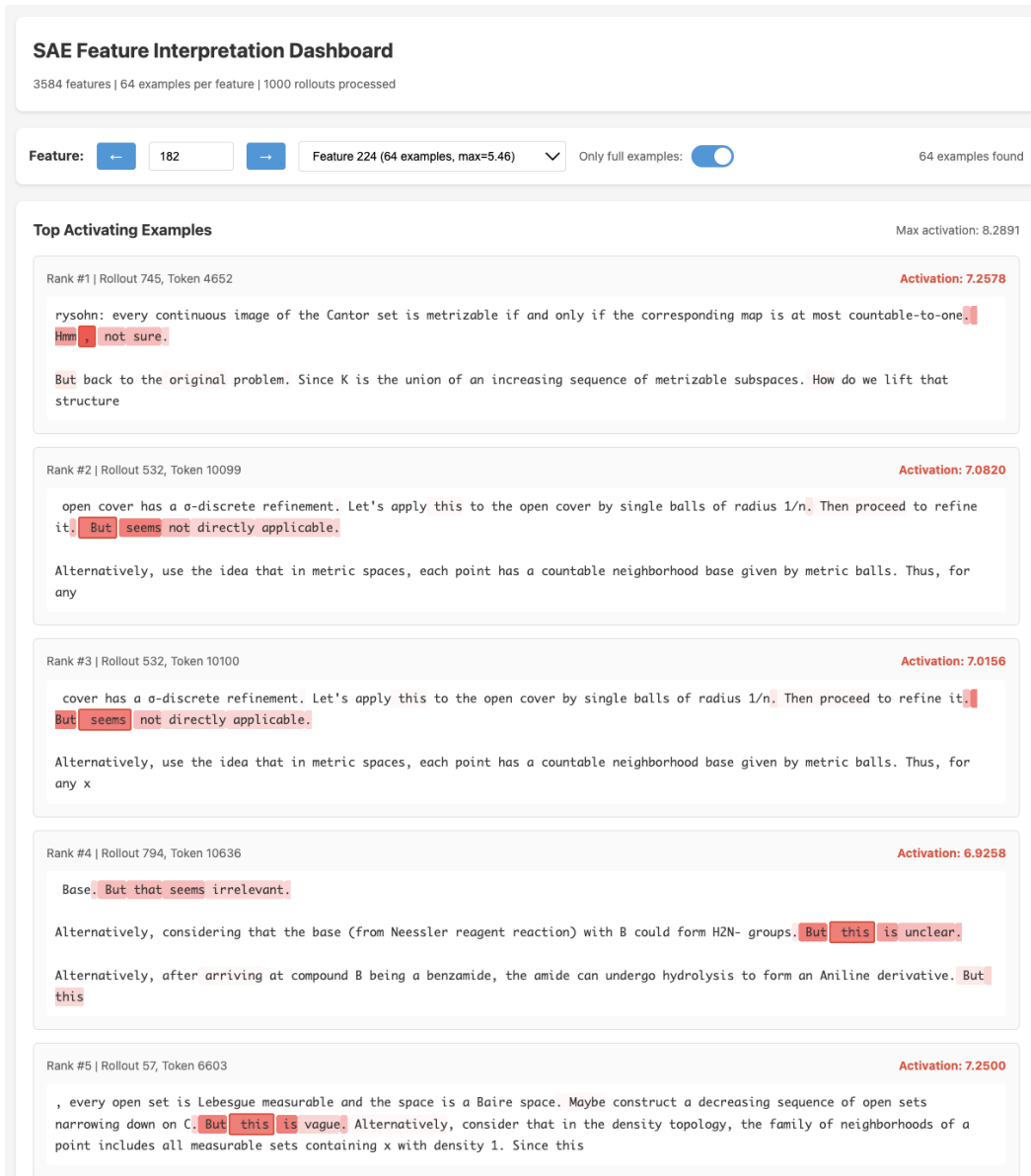


Figure 7: Autointerpretation: *Hesitation markers, notably “Hmm”, in reflective internal thought.*

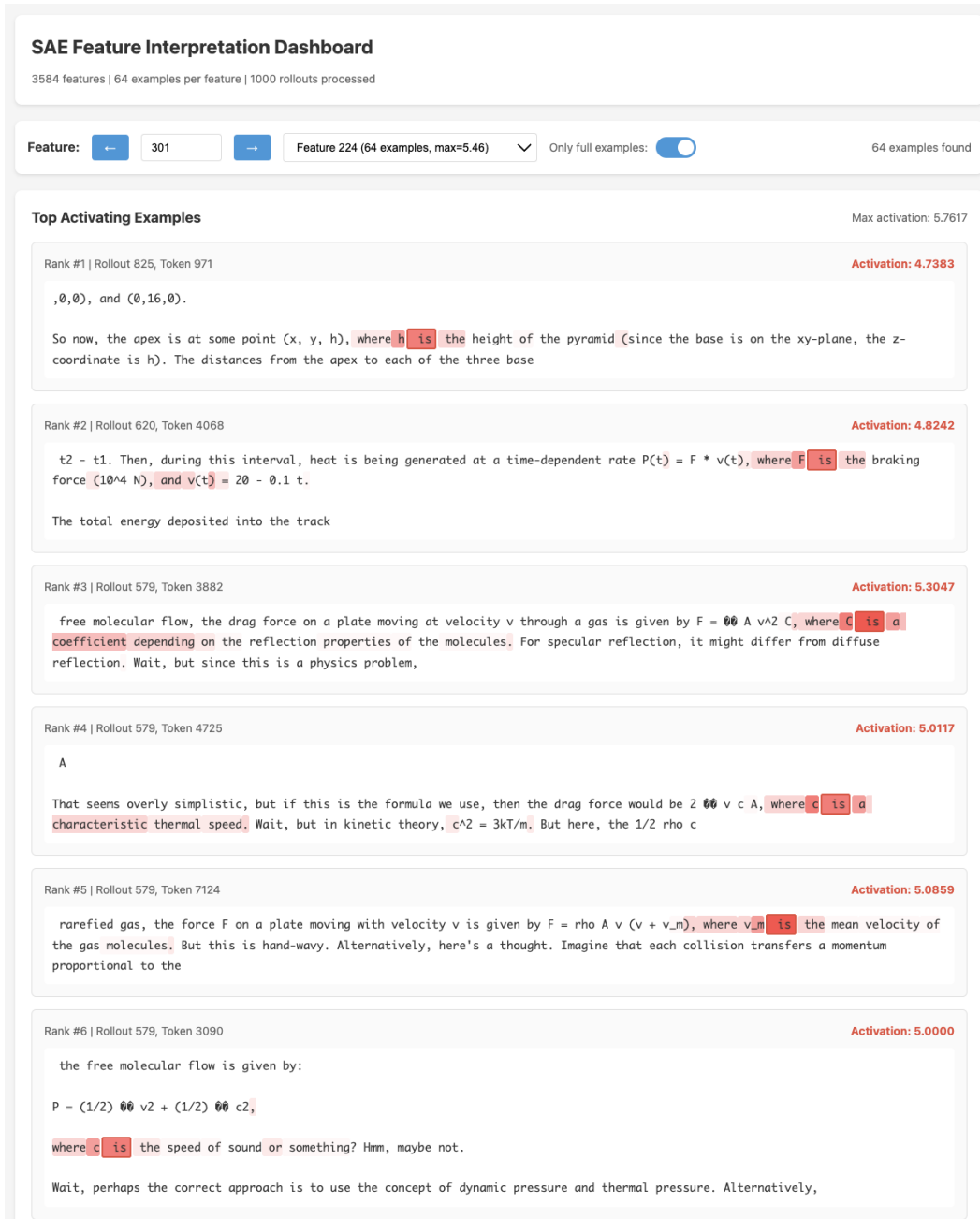


Figure 8: Autointerpretation: *Firing on “is” as an equality indicator in mathematical definitions.*

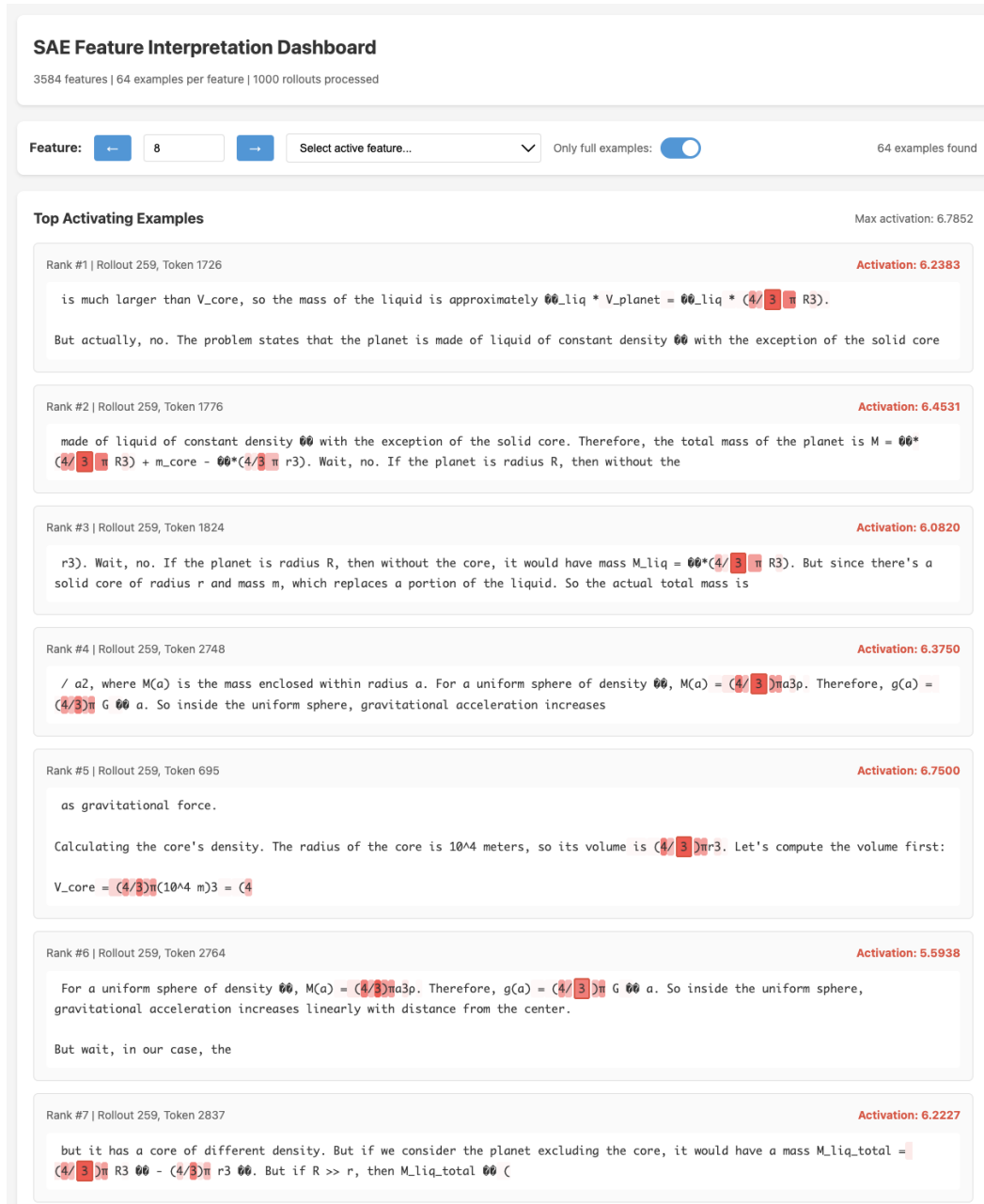


Figure 9: Autointerpretation: *Sphere volume formula tokens, specifically  $\frac{4}{3}\pi R^3$  components.*

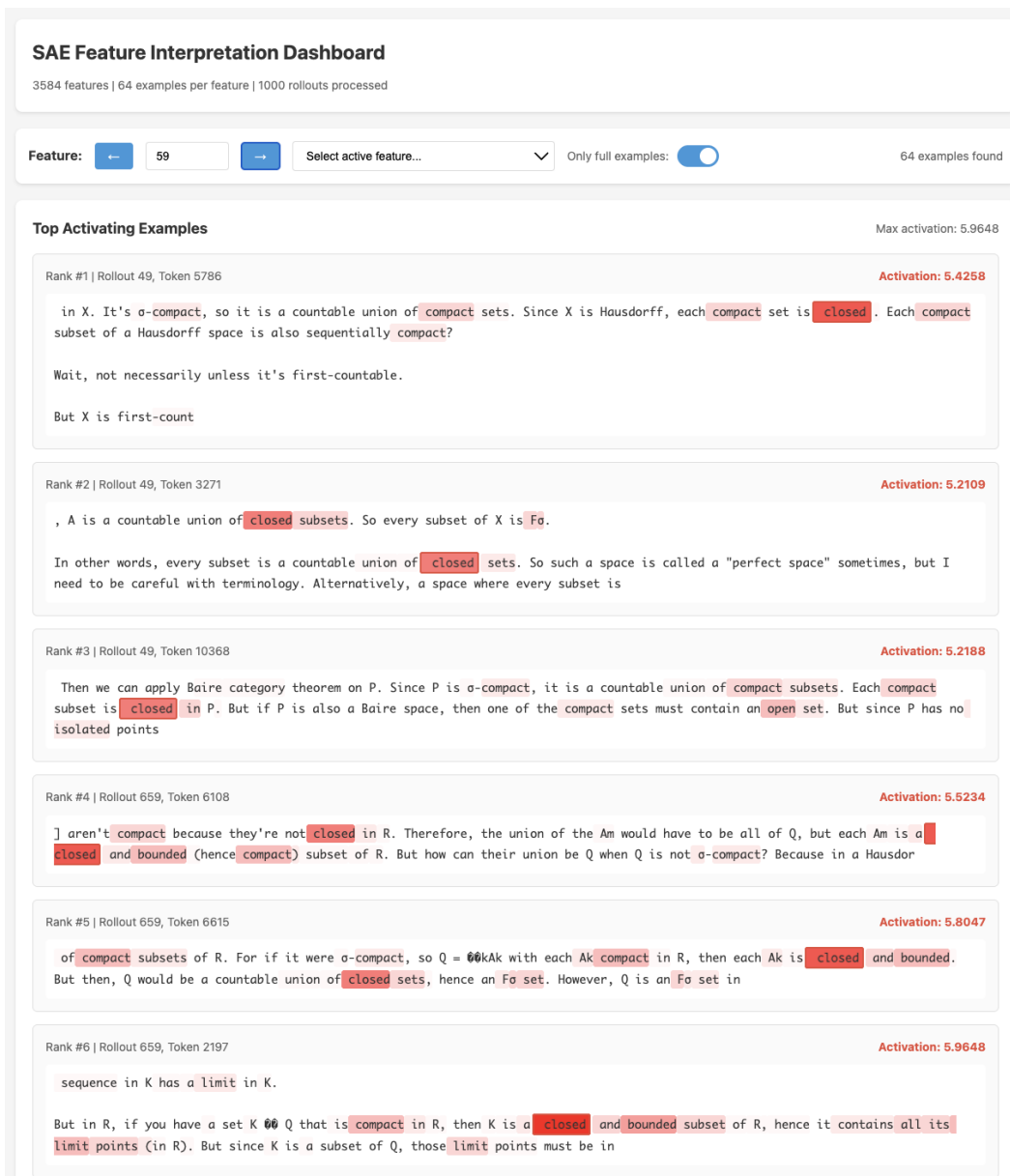


Figure 10: Autointerpretation: *Fires on “closed” in mathematical topology contexts.*

## B LLM Prompts

### B.1 Autointerpretation

We use gpt-5-mini to automatically generate interpretations for LoRA directions, MLP neurons, and SAE features:

We’re studying neurons in a neural network. Each neuron looks for some particular \ thing in a short document. Look at the parts of the document where the neuron \ activates and describe what it’s firing for.

Some activations will be noisy, in these cases you’ll have to look for common \ phrases or concepts in the examples.

If a feature always activates for the same token, you should note this in your \ explanation, and also state whether that feature represents that token in some \ specific context. You may need to look at words surrounding activating tokens \ in order to understand why a feature is firing. Features should have a clear, singular explanation and should be monosemantic. \ If there isn't a clear monosemantic explanation, note this.

Your explanation should not exceed ten words. Don't write complete sentences. \ The neuron might be responding to:

- Individual tokens or specific words
- Phrases or expressions
- Abstract concepts or behaviors
- Broader context or topics

The activation format shows the full text first, then lists tokens where the neuron \ fired along with their activation strengths (0-10 scale). Higher values mean \ stronger activation.

For example:  
cat and mouse ran around the tree. They quickly  
tree 7.81  
ran 2.30  
around 1.01

Look at every example, and then generate an explanation. After generating an \ explanation, assess how monosemantic the feature is.

```
<neuron_activations>
{activations_str}
</neuron_activations>
```

Classify the feature's interpretability:  
0: The feature is specific, clear, and monosemantic. All given examples clearly \ adhere to the explanation. The explanation is not broad, and the examples are not \ noisy. This feature has a clear and obvious interpretation.  
1: The feature may be broad or noisy, but ALL given examples still adhere to the \ generated explanation. The explanation may not be obvious.  
2: The feature appears polysemantic. Some examples do not clearly adhere to the \ generated explanation. The explanation does not cleanly explain the given examples.

Respond with JSON in exactly this format:

```
{{
  "explanation": "your concise explanation in just a few words",
  "classification": <0, 1, or 2>,
  "classification_reasoning": "brief justification for your classification"
}}
```

## B.2 Autocategorization

We use claude-opus-4.1 to generate feature categories given feature interpretations using the following prompt:

You will be provided with a list of interpretations of MLP neurons from a \ large language model. Each interpretation describes what a particular learned \ feature detects or responds to during tasks. Your task is to identify high-level functional role categories that capture what \ these features are doing computationally.

Read through all feature interpretations carefully  
Identify natural groupings based on the computational or functional role each \  
feature plays  
Generate 5-8 high-level categories that capture the major functional roles

Important considerations:

- Focus on computational function (what role the feature plays) rather than \  
surface-level similarity
- Categories should be mutually exclusive when possible, though some features \  
may have dual roles
- Aim for categories that would generalize across different types of tasks
- Include 3-5 example feature interpretations for each category (copied exactly \  
from the list provided)

Output your response as a JSON object with the following structure:

```
““json
{
  "categories": [
    {
      "string_id": "category_identifier_in_snake_case",
      "name": "Human Readable Category Name",
      "definition": "A 1-2 sentence description of what functional role these features serve.",
      "examples": [
        "exact feature interpretation from the list",
        "another feature interpretation from the list",
        "additional examples as needed"
      ]
    }
  ],
  "summary": "Your overall categorization logic and any notable patterns you observed"
}
““
```

Ensure your response contains ONLY the JSON object, with no additional text before or after.

Here are the feature interpretations to categorize:

{feature\_list}

Given these categories, we categorize each feature using gpt-5-mini using the following prompt:

Categorize this neural network feature into ONE of the given categories.

Feature explanation: "{feature.explanation}"

Activation examples (showing what tokens/contexts activate this feature):

{examples\_str}

Available categories:

{categories\_str}

Based on the feature explanation and activation examples, which category best fits this feature?  
Reply with ONLY the category string\_id, nothing else.

Your response:

## C Additional Data

<b>Task</b>	<b>Full LoRA</b>	<b>Full LoRA (attn-ablated)</b>	<b>Full LoRA (mlp-ablated)</b>
AIME'24 (no-figures)	0.5000 (100.00%)	0.3667 (50.02%)	0.1333 (−37.50%)
MATH500	0.9100 (100.00%)	0.9000 (86.84%)	0.8440 (13.16%)
GPQA-Diamond	0.5808 (100.00%)	0.5152 (27.83%)	0.5051 (16.72%)

Table 2: Absolute scores with percentages in parentheses, computed relative to the Full Rank-1 LoRA’s recovered gain:  $(x - b)/(\ell - b) \times 100\%$ , where  $b$  is the baseline and  $\ell$  is the Full LoRA score (so Full LoRA is 100%).