# SAGE: An Agentic Explainer Framework for Interpreting SAE Features in Language Models

**Jiaojiao Han**[1] **Wujiang Xu**[2] **Mingyu Jin**[2] **Mengnan Du**[3†]

[1]New Jersey Institute of Technology [2]Rutgers University
[3]The Chinese University of Hong Kong, Shenzhen

liuliujiujiu05@gmail.com mengnandu@cuhk.edu.cn
[†]Corresponding author

## Abstract

Large language models (LLMs) have achieved remarkable progress, yet their internal mechanisms remain largely opaque, posing a significant challenge to their safe and reliable deployment. Sparse autoencoders (SAEs) have emerged as a promising tool for decomposing LLM representations into more interpretable features, but explaining the features captured by SAEs remains a challenging task. In this work, we propose **SAGE** (SAE AGentic Explainer), an agent-based framework that recasts feature interpretation from a passive, single-pass generation task into an active, explanation-driven process. SAGE implements a rigorous methodology by systematically formulating multiple explanations for each feature, designing targeted experiments to test them, and iteratively refining explanations based on empirical activation feedback. Experiments on features from SAEs of diverse language models demonstrate that SAGE produces explanations with significantly higher generative and predictive accuracy compared to state-of-the-art baselines. The code is available at https://github.com/jiujiubuhejiu/SAGE.

## 1 Introduction

Large language models (LLMs) have achieved remarkable progress across diverse domains, including natural language understanding, generation, and reasoning. However, despite their impressive capabilities, LLMs remain largely opaque systems, often regarded as black boxes whose internal mechanisms are poorly understood (Zhao et al., 2024). To address this opacity, the research community has increasingly focused on decoding the information encoded in LLM representations, seeking to understand how these models process and store knowledge. Among various interpretability approaches, sparse autoencoders (SAEs) have attracted growing attention due to their ability to decompose dense neural activations into sparse, potentially interpretable features (Shu et al., 2025). Recent work has demonstrated that SAEs can identify meaningful feature dimensions in transformer representations, with applications ranging from circuit discovery to activation steering (Ferrando et al., 2025; He et al., 2025).

Despite this progress, interpreting SAE features remains a significant challenge. As SAEs are trained using unsupervised learning objectives, the semantic meaning of their learned features must be inferred post-hoc through analysis of their activation patterns. Current approaches, exemplified by Neuronpedia (Lin, 2023), rely on automated interpretation pipelines that generate natural language explanations for each SAE feature using large language models such as GPT-4 and Claude 4.5. While these methods have produced preliminary results, two fundamental problems persist. First, the generated explanations lack consistency and rigor. When different LLMs are used to explain the same feature, they often produce divergent explanations, undermining confidence in the interpretations. Second, although SAEs are explicitly designed to decompose polysemous LLM representations into monosemantic features, where each feature captures a single, coherent concept. In practice, many SAE features still exhibit polysemantic behavior, activating in response to multiple distinct semantic or structural patterns. Existing methods like Neuronpedia provide only a single explanation per feature, failing to account for this multi-faceted activation behavior and potentially missing important aspects of feature functionality.

To address these challenges, we propose **SAGE** (SAE AGentic Explainer), an agent-based framework that transforms feature interpretation from passive observation into active, explanation-driven experimentation. Rather than relying on single-pass interpretations from off-the-shelf LLMs, SAGE implements a rigorous scientific methodology that systematically formulates multiple ex-

*Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 483–495
March 25-27, 2026 ©2026 Association for Computational Linguistics

planations about each feature's behavior, designs targeted experiments to test these explanations, and iteratively refines its understanding based on empirical evidence. Furthermore, by maintaining multiple parallel explanations throughout the interpretation process, SAGE naturally captures polysemantic features, producing comprehensive multi-faceted explanations when appropriate. The major contributions of this work can be summarized as:

- We propose SAGE, a novel agent-based framework that reformulates feature interpretation as an active, explanation-driven scientific process rather than a passive, single-pass generation task.
- SAGE formulates, tests, and iteratively refines multiple parallel explanations for each feature based on empirical activation feedback.
- We perform experiments on features from diverse LLMs, demonstrating that SAGE produces more accurate, consistent, and actionable feature interpretations compared to existing methods.

## 2 Problem Formulation

In this section we first provide the technical background Sparse Autoencoders (SAEs), and then formulate the task of SAE feature explanation.

### 2.1 Sparse Autoencoders

SAEs (Bricken et al., 2023b; Cunningham et al., 2023; Templeton et al., 2024) are designed to address the opacity of large models by decomposing dense neural activations $x \in \mathbb{R}^{d_{\text{model}}}$ into sparse, potentially interpretable features $f \in \mathbb{R}^{d_{\text{sae}}}$. This is achieved by projecting the input into a much higher-dimensional feature space, where $d_{\text{sae}} \gg d_{\text{model}}$.The architecture consists of an encoder that computes the sparse features $f$, and a decoder that uses these sparse features to reconstruct the original activation, $\hat{x}$:

$$f = \text{ReLU}(W_e(x - b_{\text{pre}}) + b_e), \;\; \hat{x} = W_d f + b_{\text{dec}}. \tag{1}$$

Here, $W_e$ and $W_d$ are the encoder and decoder weight matrices, while $b_{\text{pre}}$, $b_e$, and $b_{\text{dec}}$ are bias terms. The model is trained to balance two competing objectives: reconstruction fidelity and feature sparsity, achieved with the loss function $\mathcal{L}$:

$$\mathcal{L} = \underbrace{\|x - \hat{x}\|_2^2}_{\text{Reconstruction Loss}} + \underbrace{\lambda \|f\|_1}_{\text{Sparsity Penalty}} \tag{2}$$

The first term ensures the reconstructed vector $\hat{x}$ is close to the original input $x$. The second term, an $L_1$ penalty on the feature activations $f$, encourages most features to be zero. The hyperparameter $\lambda$ controls the trade-off between these two objectives.

### 2.2 SAE Feature Explanation

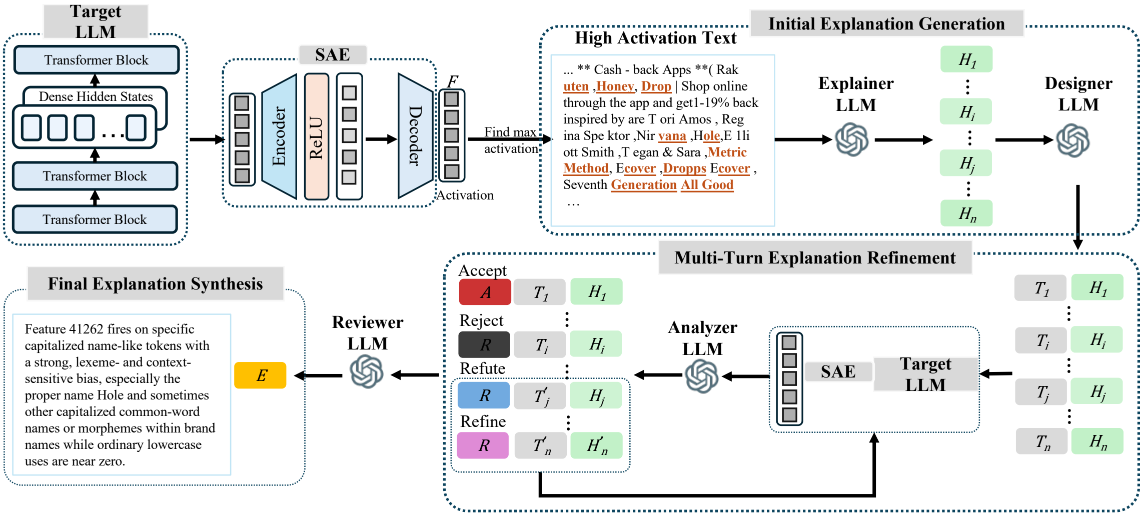
Since SAEs are trained on unsupervised objectives, the semantic meaning of their learned features, specific directions in the activation space, must be inferred post-hoc. An SAE model projects activations into a high-dimensional feature space $f \in \mathbb{R}^{d_{sae}}$, so a trained SAE with $d_{sae} = 16,000$, for example, contains 16K individual features. The ultimate goal of our work is to provide a natural language explanation $E_j$ for each of the $j \in \{1, ..., d_{sae}\}$ features. We formally define the task of SAE feature explanation for a single feature $f_j$ as finding a natural language explanation, $E_j$, that accurately describes the set of semantic or structural input patterns that cause that feature to activate.

As noted in the introduction, current single-pass generation methods often produce explanations that lack this empirical validation and fail to account for polysemantic features that respond to multiple distinct patterns. To address these limitations, we reformulate the task: instead of seeking a single, static $E_j$, our agent-based framework discovers an empirically validated explanation $E$ through an iterative process of testing and refining multiple explanations $\{H_1, ..., H_n\}$ based on multi-turn interactions with the SAE model.

## 3 The Proposed SAGE Framework

In this section, we present **SAGE** (SAE Agentic Explainer), a novel agent-based framework designed to address the challenge of SAE feature explanation1. Instead of relying on passive, single-pass generation, SAGE transforms this task into an active, iterative scientific process (see Figure 1).

The process begins when an Explainer LLM generates an initial set of explanations, $\{H_i\}$, based on high-activation text from the target LLM and SAE. A Designer LLM then creates targeted test text, $T_i$, to validate these explanations, which initiates the multi-turn explanation refinement loop. Within this loop, an Analyzer LLM observes the empirical feature activations produced when $T_i$ is processed by the target LLM. A Reviewer LLM evaluates this activation feedback and decides the next step: to accept, reject, refute, or refine the current explanations. This iterative, feedback-driven process continues until an explanation is accepted,



... ** Cash - back Apps **( Rak uten ,Honey, Drop | Shop online through the app and get1-19% back inspired by are T ori Amos , Reg ina Spe ktor ,Nir vana ,Hole,E 1li ott Smith ,T egan & Sara ,Metric Method, Ecover ,Dropps Ecover , Seventh Generation All Good …

Feature 41262 fires on specific capitalized name-like tokens with a strong, lexeme- and context-sensitive bias, especially the proper name Hole and sometimes other capitalized common-word names or morphemes within brand names while ordinary lowercase uses are near zero.

Figure 1: Overview of the SAGE framework. The process begins when an explainer LLM generates an initial explanations ($H_i$) from high-activation text derived from the target LLM and SAE. A designer LLM then creates test text ($T_i$) to validate this explanation, initiating a multi-turn explanation refinement loop. Within this loop, an analyzer LLM observes the activations produced when $T_i$ is fed into the target LLM. A reviewer LLM then evaluates this feedback and decides whether to accept, reject, refute, or refine the current explanations. This iterative process continues until an explanations is accepted, culminating in the final explanation synthesis ($H*$).

culminating in the final explanation synthesis, $E$.

### 3.1 Initial Explanation Generation

The interpretation process of our SAGE framework for a single target SAE feature $f_j$, a learned direction in the model's activation space, begins with standard feature analysis. We first extract the top-k text segments from a corpus that maximally activate this feature $f_j$. These high-activation examples serve as the empirical foundation for explanation generation. The explainer LLM then analyzes these examples using prompt $P_{init}$ (see Appendix) to formulate an initial set of $n$ explanations, $\{H_1, H_2, ..., H_n\}$, about the semantic concept encoded by $f_j$. Unlike single-pass methods that commit to a single interpretation, SAGE maintains multiple parallel explanations to capture potentially complex, context-dependent, or polysemantic activation patterns. Each explanations $H_i$ represents a distinct, testable theory about what interpretable concept or pattern triggers the feature's activation.

### 3.2 Multi-Turn Explanation Refinement

The second stage of SAGE is a multi-turn execution loop, where each explanation undergoes iterative refinement through empirical testing. For each active explanation $H_i$ at turn $t$, the system executes a structured testing cycle.

First, the explainer LLM generates test text $T_i$ designed to validate explanation $H_i$ using prompt $P_{\text{test}}$. This generated text represents a concrete prediction: if $H_i$ correctly captures the concept encoded by the SAE feature, then $T_i$ should strongly activate $F_j$. The text generation process is guided by both the explanation and accumulated evidence from previous iterations, enabling increasingly sophisticated probes of feature boundaries. Next, we obtain empirical feedback by passing $T_i$ through the target LLM and measuring the SAE feature activation: $a_i = \text{SAE}_j(\text{TargetLLM}(T_i))$. The activation magnitude $a_i$ provides direct evidence about explanation validity. Based on activation analysis, the analyzer LLM determines the next state for each explanation using system prompt $P_{\text{analyze}}$. Our framework supports four state transitions that capture different experimental outcomes:

- *Accept*: When test text $T_i$ produces strong activations matching predictions, explanation $H_i$ is accepted as a valid interpretation.
- *Reject*: If repeated tests fail to produce meaningful activations or consistently contradict predictions, explanation $H_i$ will be rejected.
- *Refine*: Partial activation matches suggest the explanation captures some aspect of feature behavior but requires modification. The system generates refined explanation $H'_i$ and updated test text $T'_i$ for the next iteration.
- *Refute*: When activation patterns directly contradict explanation predictions, the system main-

tains $H_i$ but generates alternative test text $T_i'$ to explore why the expected behavior didn't occur.

The state transition logic is formalized as:

$$(H_i^{(t+1)}, T_i^{(t+1)}, \text{status}_i) = \text{Transition}(H_i^{(t)}, T_i^{(t)}, a_i^{(t)}), \tag{3}$$

where the transition function is implemented through structured prompting of the analyzer LLM with activation analysis results. The multi-turn execution continues until all explanations reach terminal states (accepted or rejected) or maximum turns are met. Through successive iterations, initial broad and rough explanations evolve into precise descriptions of SAE feature behavior.

This iterative process enables several key capabilities. Complex conditional features emerge through refinement what begins as "technical terms" might evolve into "technical discussions in formal contexts" through testing. Polysemantic features are naturally discovered when multiple non-overlapping explanations are accepted. Edge cases and boundary conditions surface through the refute-retry cycle. Each iteration adds to an accumulating evidence base:

$$\mathcal{E}^{(t)} = \mathcal{E}^{(t-1)} \cup \{(H_i^{(t)}, T_i^{(t)}, a_i^{(t)})\}_{i=1}^{n}. \tag{4}$$

This evidence history informs subsequent explanation refinement and test generation, creating a feedback loop that drives increasingly sophisticated understanding.

### 3.3 Final Explanation Synthesis

After the iterative process converges, SAGE synthesizes final interpretations from accepted explanations. The reviewer LLM reviews all accepted explanations $\mathcal{H}_{\text{accepted}}$ and their supporting evidence using prompt $P_{\text{synthesize}}$ to generate comprehensive feature explanations $E$.

For monosemantic features, this typically yields a single refined explanation with extensive empirical validation. For polysemantic features, the synthesis identifies distinct behavioral facets and their activation conditions. The final output includes both natural language explanations and concrete examples that reliably trigger feature activation.

## 4 Experiments

In this section, we conduct experiments to evaluate the proposed SAGE framework.

Table 1: This table outlines the experimental setup, detailing the diverse set of open-source LLMs, corresponding SAE models, and the specific transformer layers selected for feature evaluation.

| LLMs | SAE Model | Layers |
|---|---|---|
| Qwen3-4b | `transcoder-hp` | 3, 7, 11, 23 |
| Gemma-2-2b | `gemmascope-res-16k` | 3, 7, 11, 23 |
| GPT-OSS-20b | `resid-post-aa` | 3, 7, 11, 23 |

### 4.1 Experimental Setup

**Implementation Details.** We evaluate SAE features from a diverse set of open-source language models using pre-trained SAEs [1]. The specific configurations of models, SAEs, and their corresponding layers employed in this study are as given in Table 1. We evaluate SAGE across these transformer architectures, focusing on layers 3, 7, 11, and 23 to capture feature behaviors spanning from early semantic processing to high-level abstraction. For each target layer, we randomly sample 10 features to ensure representative evaluation while maintaining computational feasibility. We employ GPT-5 [2] as the core language model for all agents within the SAGE framework, including the Explainer, Designer, Analyzer, and Reviewer components. A critical component of our evaluation methodology, and for our baseline comparison against Neuronpedia, our top-$k$ activating exemplars are taken from the "dashboard" of Neuronpedia. For the parameters introduced in Section 3.1, we set the number of top-k text segments $k$ to 10 and the number of initial explanations $n$ to 4.

**Baseline Comparison.** We conduct systematic comparisons against Neuronpedia, the current state-of-the-art automated interpretation system. To ensure fair comparison with Neuronpedia, we maintain strict experimental controls: (1) *Consistent Exemplar Data*: All top-$k$ exemplars are obtained through Neuronpedia's standardized activation sampling interface; (2) *Uniform Explanation Models*: Both systems utilize the same LLM (GPT-5) for generating natural language explanations; (3) *Standardized Activation Measurement*: Ground-truth activation values are retrieved using Neuronpedia's evaluation APIs; (4) *Identical Test Sets*: Feature selection and test sentence sampling procedures are identical across methods.

[1] https://www.neuronpedia.org/

[2] https://platform.openai.com/docs/models/gpt-5

Table 2: Comparison of explanation quality between SAGE and Neuronpedia baseline using generative accuracy and predictive accuracy metrics.

| **Method** | **GPT-OSS-20b** | | | **Qwen3-4b** | | | **Gemma-2-2b** | | |
|---|---|---|---|---|---|---|---|---|---|
| | Layer | Gen. Acc.↑ | Pred. Acc.↑ | Layer | Gen. Acc.↑ | Pred. Acc.↑ | Layer | Gen. Acc.↑ | Pred. Acc.↑ |
| Neuronpedia | 3 | 0.26 | 0.62 | 3 | 0.22 | 0.68 | 3 | 0.75 | 0.68 |
| **SAGE** | 3 | **0.59** | **0.80** | 3 | **0.54** | **0.72** | 3 | **0.97** | **0.83** |
| Neuronpedia | 7 | 0.57 | 0.60 | 7 | 0.25 | 0.64 | 7 | 0.30 | 0.65 |
| **SAGE** | 7 | **0.77** | **0.71** | 7 | **0.54** | **0.66** | 7 | **0.80** | **0.70** |
| Neuronpedia | 11 | 0.30 | 0.67 | 11 | 0.12 | 0.65 | 11 | 0.36 | 0.70 |
| **SAGE** | 11 | **0.52** | **0.71** | 11 | **0.23** | **0.65** | 11 | **0.56** | **0.74** |
| Neuronpedia | 23 | 0.12 | 0.52 | 23 | 0.09 | 0.65 | 23 | 0.28 | 0.64 |
| **SAGE** | 23 | **0.67** | **0.68** | 23 | **0.28** | **0.67** | 23 | **0.56** | **0.67** |

**Evaluation Metrics.** We evaluate the quality and utility of the generated feature explanations using two complementary metrics. The first, Generative Accuracy, assesses the causal validity of an explanation by measuring whether it can be used to generate novel text that reliably activates the target feature. The second, Predictive Accuracy, assesses the descriptive power of an explanation by measuring its ability to predict feature activations on held-out data. Full details on the implementation of these metrics are provided in Appendix A.

### 4.2 Explanation Results Comparisons

Table 2 compares SAGE against the Neuronpedia baseline across three language models using generative and predictive accuracy metrics. SAGE demonstrates substantial generative accuracy improvements across all configurations, with gains ranging from 29% to 458%. The most pronounced improvements occur at deeper layers where Neuronpedia deteriorates significantly. At layer 23, SAGE achieves 0.67 for GPT-OSS-20B versus Neuronpedia's 0.12, representing a 458% improvement. Predictive accuracy shows more modest but consistent gains, with SAGE scoring 0.65-0.83 compared to Neuronpedia's 0.52-0.70.

This performance divergence reveals a key distinction between the approaches. While both methods adequately describe existing activation patterns, SAGE's explanations possess significantly greater causal validity for generating novel feature-activating content. Unlike generative accuracy, predictive performance remains stable across network depths for both methods. The generalizability across model architectures confirms that iterative experimental validation benefits extend across diverse model families and scales.

### 4.3 Qualitative Evaluation

In this section, we provide several case explanations in Table 3 to qualitatively demonstrate the precision and faithfulness of SAGE's explanations.

The baseline's tendency to over-generalize is evident in feature 24625 from Qwen3-4b, described as detecting "English negative contractions using ’n’t’". Our empirical validation, however, reveals a far more specific function. SAGE's process (e.g., Test 2: "won't" and Test 10: "don't") explicitly refutes this broad hypothesis, showing zero activation. Instead, SAGE correctly identifies the feature's true, narrow scope: an "*English ’can’t’ contraction suffix detector*," defining a sharp, accurate boundary.

This rigorous validation is equally critical for polysemous features. For feature 5125 from Gemma-2-2b, the baseline provides a vague description of "multithreading synchronization" without defining its limits. SAGE's iterative validation, in contrast, not only confirms activation on multi-language code constructs (Python RLock, C++ mutex, Java synchronized) but also actively tests and refutes activations on natural-language uses of the word "lock" (e.g., Test 3: "He turned the lock on the door."). SAGE's final description, "Code synchronization/locking constructs... Natural-language uses... remain at baseline," provides a far more complete and useful explanation.

This pattern of superior precision is consistent across other examples. For instance, SAGE describes feature 121075 (GPT-OSS-20b) as a "Terrestrial lexeme/morpheme detector" sensitive to exact tokens, rather than the baseline's general "terrestrial... contexts". Similarly, for feature 1 (Gemma-2-9b-it), SAGE specifies a "lexical detec-

Table 3: Comparison of explanations of SAGE with Neuronpedia. **Blue** : first semantics, **Red** : second semantics.

| **LLM** | **Example feature**<br>layer-type/id | **Description by Neuronpedia** | **Description by SAGE (Ours)** |
|---|---|---|---|
| Gemma-2-2b | `11-gemmascope-res-16k/ 5125` | mentions of multithreading synchronization and thread-safety mechanisms, especially lock-related constructs and events. | Code synchronization/locking constructs (Python lock/R-Lock/Event idioms; C++ mutex; Java-like synchronized)".<br>Natural-language uses of 'lock/unlock/Sherlock' remain at baseline. |
| Qwen3-4b | `23-transcoder-hp/ 24625` | English negative contractions using "n't," often in auxiliary or modal verb constructions. | English "can't" contraction suffix detector ("'t"/"'t" after " can"; localized, orthography/punctuation/newline robust; moderate activations with occasional spillover) |
| GPT-OSS-20b | `3-resid-post-aa/ 121075` | mentions of terrestrial, land-based contexts such as habitats, ecosystems, animals, or planets. | Terrestrial lexeme/morpheme detector: exact ' terrestrial' token (strong) and '...restrial' fragments (strong-to-moderate), with stem-only fragments moderate.<br>Habitat list co-activation: weak activation on ' aquatic' when co-listed with strongly activated ' terrestrial'. |
| Gemma-2-9b-it | `20-gemmascope-res-131k/ 1` | mentions of Java exceptions in code/logs, especially invalid-argument error types and related exception handling. | Java IllegalArgumentException lexical detector (surface-form ' IllegalArgument' with weak 'Exception' co-activation; modest sensitivity to ' Illegal' prefix). |

tor" for the exact string "IllegalArgument". In all cases, SAGE provides more specific, empirically-grounded, and faithful explanations of the feature's true behavior.

### 4.4 Ablation Studies

We ablated the number of initial explanations $k$ generated by the explainer LLM to balance interpretation quality with computational efficiency. Figure 2 shows results for $k \in \{5, 10, 15\}$. With $k = 5$, SAGE achieves the lowest token consumption (26,500 tokens per turn) but insufficient explanation diversity, yielding only 0.648 prediction accuracy. The limited hypothesis space prevents comprehensive feature understanding, particularly for polysemantic features requiring multiple interpretations. At $k = 15$, prediction accuracy peaks at 0.667 but incurs a 19% higher computational cost (31,500 tokens per turn) compared to $k = 10$. The performance gain diminishes as additional explanations often represent redundant hypotheses. The optimal configuration emerges at $k = 10$, achieving 0.664 prediction accuracy statistically equivalent to $k = 15$ (difference of 0.003) while maintaining computational efficiency at 26,500 tokens per turn. This provides sufficient explanation diversity to capture complex feature semantics without diminishing returns. We adopt $k = 10$ as the default configuration, balancing interpretive thoroughness with computational efficiency.

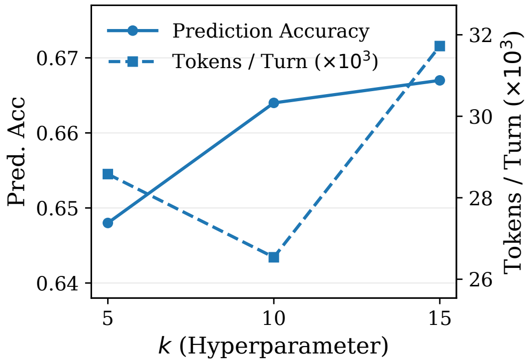


Figure 2: Ablation study on initial explanation count $k$. Prediction accuracy saturates at $k = 10$ while token consumption continues increasing, demonstrating optimal efficiency at $k = 10$.

## 5 Conclusions

In this work, we addressed the critical challenge of consistently and comprehensively interpreting features from SAEs in language models. To tackle this, we proposed SAGE, a novel agent-based framework that reformulates feature interpretation as an active, explanation-driven scientific process rather than a passive, single-pass generation task. SAGE employs a multi-turn execution loop where an explainer LLM systematically formulates, tests, and refines multiple explanations for each feature by generating targeted text and analyzing empirical activation feedback. Our comprehensive evaluations demonstrate that SAGE yields explanations with superior generative and predictive accuracy compared to existing state-of-the-art methods. Additionally, by maintaining and validating multiple parallel explanations, SAGE naturally discovers and provides multi-faceted explanations for polysemantic features, addressing a fundamental limitation of current interpretation approaches.

## Limitations

Our study has several limitations, primarily stemming from resource constraints. For each LLM and its corresponding SAE, our evaluation was conducted on only four selected layers rather than all available layers. Furthermore, within each of these layers, we randomly sampled 10 features for experimental evaluation instead of assessing the complete set of features.

## References

Andy Arditi, Oscar Balcells Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. 2024. Refusal in language models is mediated by a single direction. In *NeurIPS*.

Leonard Bereska and Stratis Gavves. Mechanistic interpretability for ai safety-a review. *Transactions on Machine Learning Research*.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. Language models can explain neurons in language models. `https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html`. Accessed: YYYY-MM-DD.

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, and 6 others. 2023a. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Trenton Bricken, Adly Templeton, Joshua Batson, and 1 others. 2023b. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A Smith. 2015. Sparse overcomplete word vector representations. In *ACL*, pages 1491–1500.

Javier Ferrando, Oscar Balcells Obeso, Senthooran Rajamanoharan, and Neel Nanda. 2025. Do i know this entity? knowledge awareness and hallucinations in language models. In *ICLR*.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2025. Scaling and evaluating sparse autoencoders. In *ICLR*.

Yoav Gur-Arieh, Roy Mayan, Chen Agassy, Atticus Geiger, and Mor Geva. 2025. Enhancing automated interpretability with output-centric feature descriptions. In *Proceedings of ACL*, Vienna, Austria. Association for Computational Linguistics.

Zhengfu He, Wentao Shu, Xuyang Ge, Lingjie Chen, Junxuan Wang, Yunhua Zhou, Frances Liu, Qipeng Guo, Xuanjing Huang, Zuxuan Wu, and 1 others. 2024. Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders. *arXiv preprint arXiv:2410.20526*.

Zirui He, Mingyu Jin, Bo Shen, Ali Payani, Yongfeng Zhang, and Mengnan Du. 2025. SAE-SSV: Supervised steering in sparse representation spaces for reliable control of language models. In *EMNLP*, Suzhou, China. Association for Computational Linguistics.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *ICLR*.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, and 1 others. 2025. Exploring concept depth: How large language models acquire knowledge and concept at different layers? In *Proceedings of the 31st international conference on computational linguistics*, pages 558–573.

Connor Kissane, Robert Krzyzanowski, Joseph Isaac Bloom, Arthur Conmy, and Neel Nanda. Interpreting attention layer outputs with sparse autoencoders. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. 2024. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In *ACL BlackboxNLP Workshop*, pages 278–300.

Johnny Lin. 2023. Neuronpedia: Interactive reference and tooling for analyzing neural networks. Software available from neuronpedia.org.

Suraj Prasai, Mengnan Du, Ying Zhang, and Fan Yang. 2026. Knowthyself: An agentic assistant for llm interpretability. *AAAI Demo Track*.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. 2024a. Improving sparse decomposition of language model activations with gated sparse autoencoders. In *NeurIPS*, pages 775–818.

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

Tamar Rott Shaham, Sarah Schwettmann, Franklin Wang, Achyuta Rajaram, Evan Hernandez, Jacob Andreas, and Antonio Torralba. 2024. A multimodal automated interpretability agent. In *Forty-first International Conference on Machine Learning*.

Lee Sharkey, Dan Braun, and Beren Millidge. 2022. Taking features out of superposition with sparse autoencoders.

Wei Shi, Sihang Li, Tao Liang, Mingyang Wan, Guojun Ma, Xiang Wang, and Xiangnan He. 2025. Route sparse autoencoder to interpret large language models. In *EMNLP*, pages 6812–6826, Suzhou, China. Association for Computational Linguistics.

Dong Shu, Xuansheng Wu, Haiyan Zhao, Daking Rai, Ziyu Yao, Ninghao Liu, and Mengnan Du. 2025. A survey on sparse autoencoders: Interpreting the internal mechanisms of large language models. In *EMNLP Findings*, Suzhou, China. Association for Computational Linguistics.

Adly Templeton, Tom Conerly, Jonathan Marcus, and 1 others. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Mengru Wang, Xingyu Chen, Yue Wang, Zhiwei He, Jiahao Xu, Tian Liang, Qiuzhi Liu, Yunzhi Yao, Wenxuan Wang, Ruotian Ma, and 1 others. 2025a. Two experts are all you need for steering thinking: Reinforcing cognitive effort in moe reasoning models without additional training. *NeurIPS*.

Mengru Wang, Ziwen Xu, Shengyu Mao, Shumin Deng, Zhaopeng Tu, Huajun Chen, and Ningyu Zhang. 2025b. Beyond prompt engineering: Robust behavior control in LLMs via steering target atoms. In *ACL*, pages 23381–23399, Vienna, Austria. Association for Computational Linguistics.

Lyucheng Wu, Mengru Wang, Ziwen Xu, Tri Cao, Nay Oo, Bryan Hooi, and Shumin Deng. 2025a. Automating steering for safe multimodal large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 792–814.

Xuansheng Wu, Wenhao Yu, Xiaoming Zhai, and Ninghao Liu. 2025b. Self-regularization with sparse autoencoders for controllable llm-based classification. In *SIGKDD*, pages 3250–3260.

Xuansheng Wu, Jiayi Yuan, Wenlin Yao, Xiaoming Zhai, and Ninghao Liu. 2025c. Interpreting and steering llms with mutual information-based explanations on sparse autoencoders. *arXiv preprint arXiv:2502.15576*.

Wei Jie Yeo, Nirmalendu Prakash, Clement Neo, Ranjan Satapathy, Roy Ka-Wei Lee, and Erik Cambria. 2025. Understanding refusal in language models with sparse autoencoders. In *EMNLP Findings*, Suzhou, China. Association for Computational Linguistics.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38.

## A More Details of the Evaluation Metrics

We employ two complementary evaluation metrics to assess the quality and utility of feature explanations generated by our SAGE framework.

- *Generative Accuracy.* This metric assesses the causal validity of an explanation: can it be used to *generate* novel text that reliably triggers the feature? We instruct an LLM to generate $N$ sentences based solely on the feature's explanation. We define a success threshold $T_{\text{act}}$ as 50% of the maximum activation observed in the initial top-10 exemplars. The generative accuracy is the success rate: the fraction of generated sentences whose maximal token activation $F_j(G(H_i))$ exceeds $T_{\text{act}}$.
- *Predictive Accuracy.* This metric assesses the descriptive power of an explanation: can it be used to *predict* feature activations on held-out data? We use a held-out set of exemplars $D_{\text{held-out}}$, distinct from the $D_{j,k}$ used for explanation generation, sampled from high, medium, and low activation groups. Following past work (Cunningham et al., 2023), we employ a simulator $\sigma$, which is an LLM prompted with the feature explanation $E_j$. For each token $t$ in a held-out example, $\sigma$ predicts the discretized activation level. Rather than single-point prediction, we compute the expected activation value using the log-probabilities $\sigma$ assigns to the output tokens '0' through '10'. The predictive accuracy is the mean Pearson correlation coefficient ($\rho$) between the predicted activation values and the true, normalized per-token activations across $D_{\text{held-out}}$.

## B Related Work

**Sparse autoencoders (SAEs).** SAEs were introduced as an unsupervised dictionary-learning approach to address superposition (Faruqui et al., 2015) in LLM (Shu et al., 2025; Huben et al., 2024). By mapping model activations into a higher-dimensional sparse space, SAEs isolate a small number of latent features per input, yielding monosemantic features that correspond to single interpretable concepts rather than polysemantic neurons (Bricken et al., 2023a). A number of SAE variants and tools have been developed to improve their efficacy and accessibility. The vanilla SAE typically uses an $L_1$ sparsity penalty on the latent vector to encourage most neurons to stay inactive (Sharkey et al., 2022) and recent variants like the Top-$K$ SAE instead enforce a fixed number $K$ of active features per input (Gao et al., 2025). Other improvements include gated or JumpReLU SAEs that modify the activation function to better balance feature detection and strength estimation (Rajamanoharan et al., 2024b,a). Some pre-trained repositories, such as Gemma Scope (Lieberum et al., 2024) and Llama Scope (He et al., 2024), enable broader research.

**SAEs Application.** SAEs have been used to interpret model representations and understand model capabilities (Wu et al., 2025b,c). Beyond static analysis, researchers have begun leveraging SAE-discovered features to steer model behavior. Such activation steering via SAE features has been used to alter attributes like sentiment, truthfulness, or style without fine-tuning the entire model (He et al., 2025; Shi et al., 2025; Wang et al., 2025b,a). They use probing for layer locate an use SAE for steering (Jin et al., 2025; He et al., 2025). SAEs have also been applied in the context of model safety and alignment. One study showed that features learned by an SAE from a language model can serve as effective probes for classifying toxic content across languages (Bereska and Gavves). By identifying which sparse features correspond to a model's refusals or safety responses, one can understand and even adjust the model's safety mechanisms. Intervening on these features has been shown to influence the model's tendency to refuse or comply with certain prompts (Arditi et al., 2024; Yeo et al., 2025; Wu et al., 2025a). Overall, SAEs offer a transparent, feature-level handle on model behaviors that is valuable for safety research.

**SAEs Feature Explanation.** Inspired by the automated interpretability pipeline that uses GPT-4 to explain GPT-2 neurons from their activating examples (MaxAct) (Bills et al., 2023), a framework that has since become the standard for large-scale interpretation of neurons and SAE-learned features in both language and vision models (Lin, 2023; Huben et al., 2024; Gao et al., 2025). Neuronpedia combines an activation-based method (Kissane et al.) that highlights the tokens most strongly triggering a feature with a logit-projection method (Kissane et al.) that infers the feature's semantic direction by measuring its positive and negative influence on output logits. Recent work proposes an "output-centric" automated feature interpretation that interprets model features not only by considering which inputs activate them, but also by examining the impact of their activation on the model output to generate more accurate and causal interpretations (Gur-Arieh et al., 2025).

**Agents for Explainability.** Recent work has explored using agentic frameworks for explainability. For instance, MAIA (Shaham et al., 2024) employs a vision-language model equipped with a set of tools to automate the interpretation of computer vision models. MAIA iteratively designs experiments, composes tools for tasks like input synthesis and exemplar generation, and formulates explanations to explain model behaviors, such as identifying feature selectivity or failure modes. Similarly, KnowThyself (Prasai et al., 2026) provides an agentic assistant specifically for LLM interpretability. It unifies various interpretability tools into a single chat-based interface, allowing users to ask natural language questions. In contrast to these applications, our work proposes an agent framework specifically designed to interpret the features learned by SAEs.

## C Examples of SAE Explanations

### Qwen3-4b 3-transcoder-hp 148551

Lexical ’amnesty’ (lowercase common-noun event; not ’Amnesty International’ or derived forms)

Specific -mstr/-msta lexemes: ’Darmstadt’, ’hamstring’ (singular), and ’Armstrong’ (surname); excludes unrelated ’-stadt’ cities, plurals, or orthographic near-misses (e.g., ’Ingolstadt’, ’Amsterdam’, ’hamster’)

### Gemma-2-2b 11-gemmascope-res-16k 148551

sudden/suddenly” lexical-morpheme detector (incl. “all of a/the sudden”) with split-morpheme robustness and punctuation spillover

Spillover in “Suddenly, there was . . . ” raising comma and ’was’ when preceded by “Suddenly

### Gemma-2-9b-it 20-gemmascope-res-131k 2

Expository-definition scaffolding (endowed-with PPs and predicate coordination in technical/encyclopedic style)

Inert on copular/list coordinations (negative control))

### Qwen3-4b 7-transcoder-hp 158076

"Recreat-" Morpheme and "-ational" Suffix Morphological Detector (Activates on words like ’Recreational’ and ’Recreativo’ via strong peaks on ’creat’ and ’ational’ subtokens)

### GPT-OSS-20b 3-resid-post-aa 72038

Chinese lexical " 的一 " detector (strong) with weak secondary sensitivity to the character " 一 " in non-Chinese CJK contexts

### Gemma-2-2b 11-gemmascope-res-16k 13574

m-final subword detector (case-/domain-agnostic) with vowel+m hierarchy (UM $\geq$ OM > um » AM/IM) and occasional internal-’em’ spillover due to tokenization

### GPT-OSS-20b 7-resid-post-aa 74421

Apartheid lexical/subword detector with compositional co-occurrence boosts (peak on ’heid’ or ’ apartheid’; moderate ’Apart’/’apart’; boosted policy/state/government/system/regime; contextual ’South/Africa’; negatives low)

## D Agent Prompts

### Pinit

**Task**: We have executed the maximum activation test on the corpus. Your mission is to systematically analyze and interpret specific SAE features. After analyzing the exemplar data, you MUST explicitly state hypotheses.

**Real Exemplar Data from Corpus Analysis**:

```
{exemplars_summary}
```

**Required Output Format**:

```
OBSERVATION:
- Pattern 1: [specific pattern description based on real data]
- Pattern 2: [another pattern description based on real data]
- Common elements: [list of common features from real exemplars]

[HYPOTHESIS LIST]:
Hypothesis_1: [Specific, testable claim based on analysis]
Hypothesis_2: [Alternative explanation for the patterns]
Hypothesis_3: [Edge case consideration - what might NOT activate this feature]
Hypothesis_4: [Additional hypothesis covering different aspects]
```

**Analysis & Hypothesis Formation Guidelines**:

- Analyze the REAL activation values and key tokens from the exemplars
- Look for linguistic patterns (suffixes, prefixes, word types)
- Identify semantic patterns (topics, domains, concepts)
- Note structural patterns (syntax, formatting)
- Be specific: “English -tion suffixes” not “English words”
- Focus on COMMON patterns across multiple exemplars
- Consider which specific tokens have the highest activation values
- **MANDATORY**: After observations, form specific, testable hypotheses about what the feature detects
- Be precise: “This feature detects Python import statements” not “This feature detects programming”
- Each hypothesis must be testable with `model.run`
- Include at least one negative control hypothesis

**Format Requirements**:

- Always start each hypothesis with “Hypothesis_X: [your specific hypothesis]”
- Base hypotheses directly on observations, not assumptions
- Include positive and negative cases
- Cover different aspects of the feature (linguistic, semantic, structural)

**Rules**:

- Observe activation patterns, activation values and identify high-activating examples
- Do NOT issue `[TOOL]` commands
- Base analysis on the REAL exemplar data provided above
- Be scientific and evidence-based
- Focus on what the feature actually detects based on the activation patterns

### psynthesize

**Task**: Review all hypotheses and their testing results. Determine if additional testing is needed before drawing final conclusions.

**All Hypotheses Information**:

```
{hypotheses_summary}
```

**Required Output Format**:

```
REVIEW SUMMARY:
[Brief summary of all hypotheses and their current status]

ASSESSMENT:
[Are all hypotheses adequately tested?]
[Are there any gaps in evidence?]
[Are there any contradictions between hypotheses?]

DECISION:
Need more testing: [YES / NO]
[If YES: Specify which hypotheses need additional testing and suggested test sentences]
[If NO: Explain why current evidence is sufficient for final conclusion]
```

**IMPORTANT - If "Need more testing: YES"**:
When suggesting additional tests, format them EXACTLY like this so they can be automatically executed:

```
- H1: Test negative control: "She left for Paris."
- H1: Test another negative: "I bought it for $5."
- H2: Test verbal use: "Batteries last for hours."
```

**Format Requirements for Suggested Tests:**

1. Start each line with "- H[number]:"
2. Put the test sentence in double quotes: "test sentence here"
3. Keep sentences simple (3-10 words)
4. One test per line

**Review Guidelines**:

- Check if each hypothesis has sufficient test evidence (at least 2-3 tests)
- Verify that CONFIRMED/REFUTED hypotheses have strong supporting evidence
- Identify any hypotheses that may need refinement or additional testing
- Consider if there are any high-activation corpus tokens that haven't been tested
- Ensure no critical patterns are missing from the analysis
- **Limit**: Suggest a maximum of 2-3 tests per hypothesis (focus on the most critical gaps)

**Rules**:

- Be thorough: review ALL hypotheses, not just the confirmed ones
- Be honest: if evidence is insufficient, say so
- Be specific: if more testing is needed, use the format above for suggested tests
- Do NOT issue `[TOOL]` commands
- Base assessment on REAL test data provided above
- **Safety**: This is review iteration {self.sm.review_count if hasattr(self.sm, 'review_count') else 1}/3. After 3 iterations, proceed to final conclusion regardless.