Interpretable Guardrails: Jailbreak Detection and Debugging with Sparse Autoencoder Features

Anonymous ACL submission

Abstract

This paper presents an interpretable jailbreak detection system that enables mechanistic debugging of safety classifiers without retraining. The paper leverages Sparse Autoencoder (SAE) features from intermediate transformer layers to decompose model decisions into semantically meaningful components. By using linear classifiers on sparse feature activations and applying statistical tests to discover causal failure mechanisms, the approach achieves comparable performance with state-of-the-art (SOTA) methods, outperforming the specialized LlamaGuard-3-8B guardrail with 16 times fewer parameters while approaching dense hidden state performance. The paper introduces a three-stage debugging framework that discovers functional feature roles, validates causal influence through activation patching, and applies targeted intervention to recover false negatives. The framework successfully reduced false negatives by 33%, improving recall by 3 percentage points while limiting precision loss to just one point. By operating at inference time with negligible computational overhead, the approach enables rapid adaptation to emerging failure modes without costly retraining cycles, demonstrating the practical value of interpretability for safety-critical systems.

1 Introduction

The rapid integration of large language models (LLMs) across enterprise applications represents one of the most significant technological shifts in recent history. The global LLM market expanded from $6.4 billion in 2024 to an estimated $36.1 billion by 2030, with over 750 million business applications expected to integrate LLMs by 2025 (MarketsandMarkets, 2024). However, this exponential growth has heightened security concerns. LLMs' powerful generative abilities, while transformative, pose significant risks when exploited maliciously. Sophisticated adversarial attempts have developed jailbreak techniques that systematically bypass safety mechanisms to elicit prohibited responses violating ethical and legal boundaries (Dong et al., 2024).

A common practice to mitigate these risks is the deployment of guardrail systems, which act as safety checkpoints to filter input prompts before they reach the main LLM and, in some cases, to monitor the model's output (Dong et al., 2024; Wang et al., 2025). While contemporary approaches have achieved strong detection performance, most rely on dense neural representations or complex language models that operate as black boxes. This opacity makes it difficult for engineers to understand why a classifier made a particular decision or to diagnose the cause of misclassifications. The standard response to failures involves collecting substantial cases and retraining, which is costly and struggles to keep pace with the continuous emergence of new attacks (Ouyang et al., 2022; Dong et al., 2024; Pantha et al., 2024).

The proposed approach addresses the guardrail transparency problem by leveraging Sparse Autoencoders (SAEs) to decompose model activations into interpretable, sparse feature representations (Cunningham et al., 2023). By building guardrails on these interpretable features, the method enables feature-level attribution for every classification decision. This transparency allows engineers to diagnose which specific features contribute to misclassifications and adjust feature weights or thresholds accordingly, enabling targeted fixes without full model retraining. The presented approach achieves competitive performance with state-of-the-art methods on jailbreak detection, while a mechanistic debugging framework recovers 33% of false negatives with minimal precision loss, demonstrating the practical value of interpretability for safety-critical systems. The end-to-end inference workflow is shown in Appendix A.

The contributions of this work are as follows:

1. **Interpretable guardrails via SAE features:** A guardrail system where safety decisions decompose into sparse, semantically meaningful features and achieve SOTA performance.

2. **Mechanistic debugging framework:** A three-stage pipeline that diagnoses and corrects classifier failures at inference time without model retraining. The framework partitions features into functional roles (intent features that drive harm prediction, veto features that indicate benign context), validates causal influence through activation patching, and applies targeted intervention to recover false negatives while protecting precision. This approach enables engineers to address specific failure modes through feature-level surgery rather than costly data collection and retraining cycles.

3. **Empirical analysis of failure modes:** Feature analysis revealing that successful jailbreak arises from weak intent signals rather than signal cancelation. Jailbreak attacks succeed by attenuating harmful intent expression, not by introducing competing benign signals that mask the harm. This finding motivates selective amplification as the intervention strategy and clarifies the asymmetric roles of intent and veto features in the debugging framework.

2 Related Works

2.1 LLM Safety Guardrails

LLM safety guardrails are systematic methods that monitor LLM inputs and outputs to ensure compliance with ethical and policy constraints (Dong et al., 2024; Wang et al., 2025; Pantha et al., 2024). Early approaches to content moderation relied on rule-based systems and keyword filtering, which proved insufficient for the nuanced linguistic patterns of adversarial prompts. Tools like Perspective API and OpenAI's Content Moderation API represented initial efforts but demonstrated limited effectiveness against sophisticated jailbreaking techniques (OpenAI, 2024; Lees et al., 2022). Contemporary guardrail systems have evolved toward more sophisticated neural approaches. LlamaGuard utilizes instruction-tuned Llama2-7B models trained on safety datasets with rich categorical risk taxonomies (Inan et al., 2023). NeMo Guardrails implements a programmable framework using text embeddings to match user inputs against predefined conversation flows (Rebedea et al., 2023). Guardrails AI provides a modular system for combining multiple validators into input and output guards, supporting both rule-based and machine learning-based detection mechanisms.

Furthermore, recent work has explored leveraging language models as feature extractors for lightweight safety classification. Research demonstrates that for classification tasks, linear probing on small local models can match or exceed the performance of large commercial models (Cho et al., 2023; Gao et al., 2024; Buckmann and Hill, 2024). Building on these foundations, methods that train classifiers on hidden states from optimal intermediate layers of small language models have achieved strong performance on content safety and prompt injection detection (Buckmann and Hill, 2024; Sawtell et al., 2024). However, these systems face a fundamental limitation: opacity in failure diagnosis. Current neural guardrail approaches, including those using classifiers on dense hidden state representations, provide classification scores or decisions without explanation of which semantic factors drove the decisions. When guardrails produce incorrect classifications, engineers lack diagnostic tools to understand underlying causes or address failures efficiently. The standard response involves collecting substantial amounts of failure cases, retraining the entire model or classifier, and validating that fixes do not degrade performance on other inputs (Dong et al., 2024; Pantha et al., 2024; Wang et al., 2025). This creates a critical need for guardrail systems that maintain high detection performance while providing transparency into decision-making processes.

2.2 Mechanistic Interpretability and Sparse Autoencoders

Mechanistic interpretability (MI) is the study of the internal processes and representations that drive a model's outputs, aiming to reverse-engineer neural networks by breaking down their internal computations into understandable components (Olah et al., 2020; Bricken et al., 2023; Wang et al., 2022; Elhage et al., 2024). As large language models grow in size and are deployed in safety-critical applications, understanding their internal decision-making processes becomes increasingly important for trustworthiness and oversight. One of the challenges in MI is polysemanticity, the situation that individual

neurons in neural networks typically respond to multiple unrelated concepts rather than representing single interpretable features (Olah et al., 2020; Bricken et al., 2023; Cunningham et al., 2023). This entanglement makes it difficult to understand what factors drive model predictions or to intervene on particular concepts without affecting others.

Sparse Autoencoders (SAEs) have emerged as a promising technique to address polysemanticity by decomposing dense, entangled neural network activations into sparse, interpretable monosemantic features (Cunningham et al., 2023; Zhang et al., 2025). The advantage of this approach is that each learned SAE feature represents a single interpretable semantic pattern, enabling human understanding of what concepts the model has learned and how these concepts combine to produce predictions. This approach has successfully revealed interpretable features in language models including syntax patterns, sentiment, factual knowledge, and domain-specific concepts (Cunningham et al., 2023; Anthropic Interpretability Team, 2024).

Recent work has been exploring explored SAE-derived features can serve as effective representations for downstream classification tasks. Studies demonstrate that SAE features can achieve competitive or superior performance compared to hidden state baselines on tasks including toxicity detection and multilingual classification (Kantamneni et al., 2024; Zhang et al., 2025; Anthropic Interpretability Team, 2024). However, the literature also reveals variability in SAE performance across different tasks and domains (Wu et al., 2025; Gallifant et al., 2025). Critically, existing SAE classification work has focused primarily on general toxicity detection or non-adversarial tasks. Jailbreak detection presents distinct challenges beyond standard toxicity classification: adversarial prompts are deliberately crafted to appear benign while eliciting harmful outputs, often employing sophisticated obfuscation techniques such as role-play scenarios, encoded instructions, or multi-turn manipulation (Shayegani et al., 2023; Schulhoff et al., 2023). These attacks exploit subtle semantic patterns that differ from the overtly toxic language targeted by traditional content moderation systems. Prior SAE research has not systematically evaluated whether sparse features can capture these adversarial patterns or whether the interpretability of SAE features remains meaningful when applied to intentionally deceptive inputs.

This gap motivates the present study: to empirically investigate whether SAE-based classification can achieve both strong detection performance and practical interpretability specifically for adversarial jailbreak detection. The study examines whether the interpretability benefits of sparse representations can be realized without sacrificing detection performance in this critical application domain, and whether SAE features provide actionable debugging capabilities when guardrails encounter sophisticated adversarial attacks.

3 Methodology

3.1 Problem Formulation

Safety classifiers built on SAE features face a systematic failure mode: false negatives often involve samples where harmful intent is expressed weakly or ambiguously. The classifier is underconfident rather than actively misled, as the harmful signal exists but fails to cross the decision threshold.

Consider a linear classifier over SAE features:

$$\text{logit}(x) = \sum_i w_i \cdot f_i(x) \tag{1}$$

For correctly classified harmful samples, harmful intent-associated features activate strongly, producing high logits. For false negatives, however, the same features activate weakly, as the harmful content is present but expressed in ways the classifier underweights. A naive solution, lowering the decision threshold, would recover these false negatives but at substantial cost to precision. The challenge is selective amplification: boost confidence for samples with genuine harmful intent while protecting benign samples from false escalation.

3.2 Feature Discovery

The framework partitions SAE features into two functional roles based on their relationship to classifier behavior.

Intent features: Intent features identify harmful content signals, including weak or ambiguous expressions that the classifier underweights. These features amplify weak harmful signals to improve recall. A feature qualifies as intent if it satisfies two criteria:

1. **Positive coefficient:** $w_i > \tau_w$, indicating the feature contributes positively to harm prediction

2. **Harm-predictive:** $\text{PPV}_{\text{harm}}(f_i) > \tau_p$, where PPV (positive predictive value) measures how reliably feature activation indicates actual harm

The conjunction ensures selected features both influence the classifier toward harm predictions and do so reliably. Features with high coefficients but low PPV may reflect spurious correlations; features with high PPV but near-zero coefficients lack predictive leverage. A feature with high activation but low PPV contributes less than a moderately-activated high-PPV feature.

Veto features: Veto features indicate benign contextual markers, and their presence suggests a sample is likely safe despite any surface-level intent signal. These features protect precision by suppressing intervention when benign context is present. Selection uses a stringent criterion:

$$\mathcal{V} = \{f_i : \text{PPV}_{\text{benign}}(f_i) > \tau_v\} \tag{2}$$

With $\tau_v = 0.95$, a veto feature's activation implies 95%+ probability the sample is truly benign. This high threshold reflects the feature's protective role: veto features gate whether intervention occurs, so false benign signals would suppress legitimate harm detection.

The partition reflects an asymmetric intervention strategy: amplify where intent is present, but protect precision by gating the amplification based on benign context.

3.3 Causal Validation

Correlation-based discovery identifies features associated with outcomes but does not establish causal influence. For intent features, which drive the intervention, this distinction is critical, as effective amplification requires features that actually move predictions when modified. Veto features play a different role: they gate whether intervention occurs rather than driving it directly. High PPV ($> 95\%$) already ensures reliable indication of benign context.

The framework validates intent features through activation patching (Vig et al., 2020; Meng et al., 2022), isolating individual feature contributions:

1. Identify false negative samples $\mathcal{F} = \{x : y = 1, \hat{y} = 0\}$
2. Identify true positive samples $\mathcal{T} = \{x : y = 1, \hat{y} = 1\}$
3. For each candidate feature f_i and sample $x \in \mathcal{F}$, create x' by replacing $f_i(x)$ with $f_i(x_d)$ from donor $x_d \in \mathcal{T}$
4. Measure whether the prediction flips: $\hat{y}(x') \neq \hat{y}(x)$

The flip rate quantifies the causal effect:

$$\text{flip\_rate}(f_i) = \frac{|\{x \in \mathcal{F} : \hat{y}(x') \neq \hat{y}(x)\}|}{|\mathcal{F}|} \tag{3}$$

Features with a flip rate below the threshold are discarded, ensuring that the intervention uses features that demonstrably change the predictions.

3.4 Intervention Design

Given validated intent features $\mathcal{I}$ and veto features $\mathcal{V}$, the framework applies an intervention that selectively amplifies weak intent signals while maintaining precision.

The intervention adds a nudge to the classifier's logit:

$$\text{nudge}(x) = S_{\text{intent}}(x) \cdot \alpha \cdot p_0(x)^2 \cdot \gamma^{|\mathcal{V}(x)|} \tag{4}$$

where:

- $S_{\text{intent}}(x) = \sum_{f_i \in \mathcal{I}} \mathbf{1}[f_i(x) > 0] \cdot \tilde{f}_i(x) \cdot w_i \cdot \text{PPV}_i$ aggregates the intent signal, weighting by the magnitude of the coefficient and its reliability
- α controls the strength of the intervention
- $p_0(x)$ is the predicted probability of the baseline.
- $\gamma \in (0, 1)$ is the veto decay factor
- $|\mathcal{V}(x)|$ counts active veto features

The final prediction applies in logit space:

$$p_{\text{final}}(x) = \sigma\left(\text{logit}_0(x) + \text{nudge}(x)\right) \tag{5}$$

The term p_0^2 concentrates on intervention in uncertain samples. When $p_0 \approx 0.5$, the multiplier is ≈ 0.25; when $p_0 \approx 0.1$, it drops to 0.01. This reflects the selective amplification goal: samples that the classifier is already confident about need no

adjustment. The quadratic form provides smooth attenuation rather than a hard threshold.

Meanwhile, the term $\gamma^{|\mathcal{V}(x)|}$ protects precision during recall improvement. Amplifying weak intent signals risks boosting benign samples that happen to trigger intent features spuriously. The veto mechanism identifies such cases through benign contextual markers and suppresses the intervention accordingly. This gating allows the framework to pursue recall improvement without introducing excessive false positives. The intervention acts strongly only when an intent signal is present and a benign context is absent.

4 Experiment

4.1 Data

The experiment used WildGuardMix, a curated safety dataset that covers 13 harmful categories and has been cited in numerous works in the LLM safety area (Han et al., 2024). WildGuardMix intentionally contains offensive and harmful content, as this is essential for training and evaluating LLM safety systems - the core research objective of this work. Its comprehensive coverage of jailbreak patterns collected systematically from multiple sources make it suitable for evaluating detection systems against varied adversarial techniques. No personally identifiable information is present in the dataset; all prompts consist of synthetic generations and anonymized user-LLM interactions without names, addresses, or other identifying details. The data was used solely for classifier training and evaluation, with no redistribution or modification of the original content.

During preprocessing, categories appearing in less than 1.5% of data were removed to prevent class imbalance, in which only one class "others" did not pass the test, retaining 12 harm categories plus benign. In addition, there was deduplication before splitting, in which duplicate prompts are identified and one instance is randomly retained (fixed seed=42), preventing identical prompts across splits. Finally, the data set is stratified by subcategory, producing 80% train (14,817 samples), 10% validation (1,852 samples) and 10% test (1,853 samples), with approximately 57% harmful and 43% benign prompts per split.

4.2 Model Architecture

Experiments are conducted on Gemma-2B-IT (Gemma Team, 2024) with pretrained SAE, GEMMASCOPE on layer 12 (Lieberum et al., 2024). The SAE dictionary contains 16,384 features trained on residual stream activations of the base Gemma-2-2B model. Although trained in the base model, previous work demonstrates that SAE features learned in base models generalize to fine-tuned versions, as the core semantic representations remain largely stable in instruction-tuning (Lieberum et al., 2024; Yeo et al., 2025). This transfer property has enabled SAE application in safety analysis across model variants. Layer selection proceeds via systematic linear probe sweep across layers 6-20, since previous literature has shown that the LLM's middle range captures both low-level features and high-level semantics (Jin et al., 2024; Sawtell et al., 2024). Layer 12 achieves optimal performance and was chosen as the layer for the main experiment. Regarding GPU, the experiment was conducted on NVIDIA A100 GPU.

4.3 SAE-based Guardrails Training

The baseline classifiers are trained on SAE features with hyperparameter tuning.

Binary classifier: A logistic regression model is trained to distinguish harmful from benign prompts. Hyperparameter search is conducted over regularization strength $C \in \{0.001, 0.01, 0.1, 1.0, 10.0\}$, penalty type $\{L1, L2\}$, with 3-fold cross-validation optimizing F1 score. Class weights are balanced to handle label imbalance. Standard scaling is applied within cross-validation folds to prevent leakage.

Multiclass classifier: A separate logistic regression model is trained on harmful samples only to categorize harm subtypes. The same hyperparameter search procedure is applied with macro-F1 as the optimization target. This classifier enables fine-grained analysis of which harm categories the intervention affects.

4.4 Debugging and Intervention Pipeline

The mechanistic intervention pipeline operates in three stages on the validation set to avoid data leakage, progressively refining feature selection from correlation to causation.

Feature discovery: The first stage partitions SAE features into functional roles based on their statistical relationship to classifier behavior. Intent

features are selected using coefficient threshold $\tau_w = 0.05$ and harm PPV threshold $\tau_p = 0.70$, yielding 22 candidates. The coefficient threshold ensures selected features have meaningful influence on classifier output; the PPV threshold ensures they reliably indicate harmful content when active. Veto features use a stricter benign PPV threshold $\tau_v = 0.95$, yielding 128 features. The high threshold reflects veto's protective role: these features must reliably signal benign context to prevent false positives when amplifying weak intent signals.

Causal validation: Correlation-based discovery identifies features associated with outcomes but does not guarantee causal influence on predictions. The second stage validates intent features through activation patching, copying feature values from correctly-classified samples into misclassified ones and measuring prediction change. Patching is performed on validation-set false negatives ($n = 109$) with 5 donor samples per test to reduce variance. Features achieving flip rate $\geq 1\%$ are retained as causally validated, reducing intent features from 22 to 17. The five rejected features show near-zero flip rates despite positive coefficients, suggesting their correlation with harm reflects confounding rather than direct influence. Veto features bypass causal validation since they gate rather than drive the intervention; their high PPV threshold already ensures reliability.

Parameter tuning: The final stage determines intervention strength through grid search over $\alpha \in \{6, 7, 8, 9, 10, 12, 15\}$, veto decay $\gamma \in \{0.30, 0.35, 0.40, 0.45, 0.50\}$, and prediction threshold $\tau \in \{0.45, 0.46, 0.47, 0.48, 0.49\}$. Conservative selection chooses the lowest α among configurations improving validation F1 by ≥ 0.5 points. This policy favors robust generalization: higher α values achieve greater validation recall but risk overfitting. The configuration ($\alpha = 6.0$, $\gamma = 0.35$, $\tau = 0.46$) achieves meaningful recall improvement while maintaining generalization to held-out data.

5 Results and Discussion

5.1 SAE-based Guardrails

Table 1 presents the performance comparison of the SAE-based classifier against two alternatives: LlamaGuard-3-8B (the specialized guard mdoel with SOTA performance) and hidden state linear probe (same architecture but using dense representations). This comparison strategy isolates different aspects of the approach: LlamaGuard-3 provides context for performance relative to current industry standards, while the hidden state baseline controls for model architecture and layer selection, attributing performance differences solely to SAE versus dense feature representation.

Table 1: F1 scores comparison between alternatives

Method	Binary F1	Multiclass F1
LlamaGuard-3-8B	0.85	0.37
Hidden Probe	0.95	0.63
SAE (this work)	0.92	0.58

For binary classification, the SAE classifier achieves an F1 of 0.92, substantially outperforming LlamaGuard-3 (0.85) and approaching the hidden state baseline (0.95). The 3-point gap relative to the dense baseline suggests that SAE sparsity introduces a modest performance trade-off compared to using all 2,048 dense hidden state dimensions. However, this trade-off enables the interpretability and debugging capabilities demonstrated in subsequent sections, as dense representations lack the feature-level decomposition necessary for mechanistic analysis. For harm categorization across 13 types of violations, the SAE classifier achieves macro-F1 of 0.58, outperforming LlamaGuard-3 (0.36) by 22 points but trailing the hidden state baseline (0.63) by 5 points. The hidden state baseline's superior performance (0.95 binary, 0.63 multiclass) establishes an upper bound for linear classification on Gemma layer 12 representations.

5.2 Debugging and Intervention Results

Table 2 presents the impact of the mechanistic intervention on binary classification performance. The debugging framework achieves F1 of 0.93, narrowing the gap with the hidden-state upper bound (0.95) by approximately 30%.

Table 2: Comparison of intervention methods (FN: false negatives, FP: false positives, Prec: precision).

Method	F1	Prec.	Recall	FN	FP
Baseline	0.92	0.93	0.90	104	70
Threshold (t=0.37)	0.92	0.91	0.94	64	101
Intent only	0.92	0.91	0.94	68	93
Intent + Veto (this work)	0.93	0.92	0.93	70	87

Recall improvement: For guardrail systems, recall

(the detection success rate) provides the most operationally relevant metric. The asymmetry stems from downstream consequences: a missed jailbreak (false negative) may lead to harmful model outputs that damage user trust or cause real-world harm, while a false positive merely triggers an overly cautious refusal that users can clarify or rephrase. This asymmetry is particularly acute for high-stakes deployments where even a small number of successful jailbreaks can have outsized negative impact.

The proposed intervention improves recall from 0.90 to 0.93, translating to 34 fewer missed jailbreaks per 1,000 harmful queries, a meaningful reduction in attack surface. The intervention accepts a small precision reduction (0.93 to 0.92) to achieve these recall gains. The tradeoff is favorable when examined concretely: per 1,000 queries, the baseline misses 100 harmful samples and incorrectly flags 70 benign ones. After intervention, 70 harmful samples are missed (a reduction of 30) while 87 benign ones are incorrectly flagged (an increase of 17). The exchange rate of roughly 2 recovered jailbreaks per added false positive favors intervention under most reasonable risk weightings. Organizations with different risk profiles can adjust the α parameter to shift this balance: higher α recovers more false negatives at the cost of more false positives, while lower α achieves more modest recall gains with smaller precision impact.

Ablation study: Comparing the Intent-only and Intent+Veto rows reveals how the veto mechanism protects precision during recall improvement. Both approaches target weak intent signals for amplification and recover a similar number of false negatives (36 vs 34). However, Intent-only produces 93 false positives compared to 87 for Intent+Veto. This 6 false positives difference reflects a key principle: without veto gating, the intervention acts on any sample with intent signal, including benign samples that happen to trigger intent features spuriously. The veto mechanism identifies such cases through benign contextual markers and suppresses the intervention accordingly. In deployment, this selectivity compounds: across millions of queries, even small per-query precision improvements translate to substantially fewer frustrated users encountering unnecessary refusals. Meanwhile, threshold adjustment ($t = 0.37$) achieves the highest recall (0.94) but represents a blunt instrument. Lowering the threshold uniformly increases all borderline predictions, regardless of whether the uncertainty reflects a genuinely weak harmful signal or classifier noise on benign content. The result is 101 false positives, 31 more than baseline and 14 more than the proposed approach.

5.3 Feature Analysis

5.3.1 Feature Activation Patterns

Figure 1 reveals how intent and veto features distribute across classifier outcomes. Intent features show strong activation in true positives (0.36) but weak activation in false negatives (0.14), similar to true negatives (0.10). Veto features are sparse across all groups, with only true negatives showing measurable activity (0.02).

The pattern supports the weak-signal interpretation: false negative intent activation (0.14) sits much closer to true negatives (0.10) than to true positives (0.36). If jailbreaks caused false negatives by triggering competing veto signals, false negatives would show high intent and high veto. Instead, they show weak intent and near-zero veto, indicating an attenuated harmful signal rather than a masked signal. The absence of veto features in false negatives confirms that they do not cause misclassification; their presence in true negatives confirms their protective role during intervention.

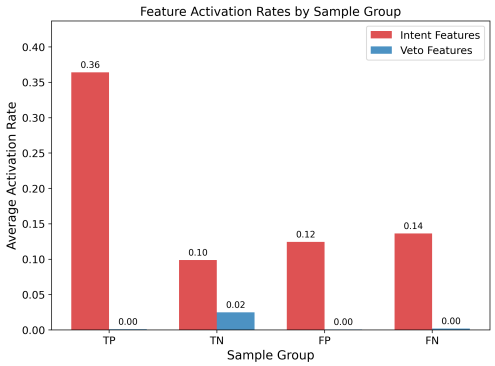

Figure 1: Feature activation rates by sample group.

5.3.2 Intervention Dynamics

Figure 2 shows the probability distribution of false negative samples before and after intervention. The distribution shifts rightward, with samples near the decision boundary (0.3 to 0.5) crossing the threshold. This confirms the p_0^2 borderline-focus works as designed: uncertain samples receive the largest boost while confident predictions remain stable.

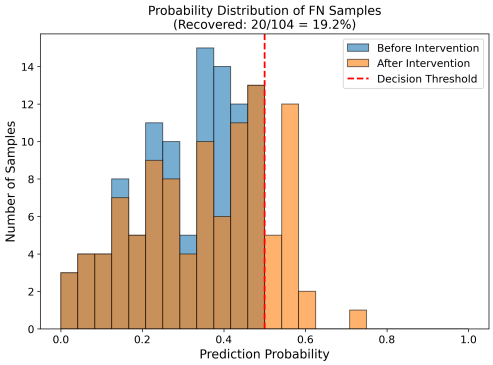

Figure 2: Probability distribution of false negatives before (blue) and after (orange) intervention.

5.4 Feature Interpretation

To understand the discovered features' meaning, each intent and veto feature was examined through Neuronpedia (Neuronpedia, 2025). This experiment used Neuronpedia with GPT-5-mini as the explanation model for its balance of up-to-date capabilities and cost-effectiveness.The full list appears in Appendix E.

Intent features: The 17 causally validated intent features cluster into four interpretable groups summarized in Table 3. The analysis reveals that jailbreak detection operates through multiple complementary mechanisms. The jailbreak assistance group is particularly revealing: features 5583 and 11551 fire not on harmful content itself but on cooperative framing patterns. This explains why false negatives show weak intent activation, as successful jailbreaks obscure assistance-seeking markers through obfuscation techniques like roleplay or hypothetical framing.

Table 3: Intent feature groups with examples.

Group	Captures	Examples
Direct harm	Explicit harmful content	11047 (abusive), 4331 (villains), 8836 (negative)
Attack vector	Domain threats	11320 (hacking), 282 (scams), 9215 (espionage)
Jailbreak assistance	Cooperative framing	5583 (facilitating), 11551 (offering help)
Structural	Technical patterns	3932 (numeric), 16170 (jargon)

Veto features: The 128 veto features cluster into four interpretable groups summarized in Table 4. The analysis reveals that veto features detect benign contexts rather than anti-harm signals. Feature 676 (recipes) illustrates this: cooking instructions share structural similarity with harmful how-to content, but recipe-related tokens indicate a domain where harm is unlikely. The feature identifies the entire context as benign rather than detecting safe versus dangerous procedures.

Table 4: Veto feature groups with examples.

Group	Captures	Examples
Technical	Professional context	3021 (legal), 2163 (config), 2971 (code)
Everyday benign	Safe queries	676 (recipes)
Informational	Discussion vs instruction	568 (missing persons), 2377 (despair)
Structural	Text formatting	742 (punctuation), 1713 (BOS), 2653 (ranges)

Implications: The semantic coherence of discovered features validates the practical value of SAE-based classification. The interpretations clarify why the intervention succeeds: it amplifies residual intent signals when veto features do not indicate benign context. The favorable 2:1 recovery ratio reflects veto features successfully distinguishing genuine jailbreaks from benign samples with incidental intent activation.

6 Conclusion

This work presents an interpretable approach to jailbreak detection that leverages sparse autoencoder features to enable mechanistic debugging of safety classifiers. The framework decomposes model activations into semantically meaningful features, providing transparency into guardrails decisions while achieving competitive detection performance. A three-stage debugging pipeline discovers functional feature roles, validates causal influence through activation patching, and applies targeted intervention that reduces false negatives by 33% with one percentage point precision loss.

The results demonstrate that interpretability and detection performance need not be at odds. For safety-critical systems where auditability matters, the ability to examine which features drove a decision and to correct specific failure modes without retraining provides practical value beyond raw classification metrics. As adversarial attacks continue to evolve, interpretable detection systems that support rapid diagnosis and adaptation offer a promising path toward robust AI safety.

7 Limitation

While the proposed framework demonstrates the feasibility of interpretable jailbreak detection and mechanistic debugging, several directions merit further investigation. Firstly, the intervention framework operates with fixed parameters determined through validation-set tuning, which assumes that the distribution of jailbreak attacks remains stable between tuning and deployment. In practice, adversarial actors continuously develop new obfuscation techniques, and an intervention calibrated for current attack patterns may become less effective as the threat landscape evolves. Future work could explore adaptive intervention mechanisms that update feature weights or thresholds based on observed failure patterns, enabling the system to maintain effectiveness against emerging attacks without full retraining cycles.

Secondly, the current framework focuses exclusively on prompt-side classification, determining whether an input is harmful before the model generates a response. An extension of this work would involve analyzing response-side features to detect when a model is producing harmful output during generation. SAE features extracted from intermediate generation steps might reveal patterns indicating compliance with jailbreak requests, enabling intervention during generation rather than only at the input gate. Such an approach would provide a complementary line of defense for cases where prompt-level detection fails.

Finally, the debugging framework improves classifier decisions through post-hoc score adjustment but operates externally to the model itself. Recent work on representation engineering and activation steering demonstrates that model behavior can be modified by directly manipulating internal activations during inference. Integrating SAE-based feature discovery with these steering techniques could enable intervention that shapes model behavior rather than merely flagging problematic inputs. If intent and veto features correspond to directions in activation space that influence model compliance with harmful requests, steering along these directions might reduce harmful outputs without relying on an external classifier, bridging interpretability-focused analysis with direct behavioral intervention.

8 Ethical Considerations

This research presents defensive work aimed at improving AI safety through interpretable jailbreak detection. However, the team acknowledges dual-use concerns associated with the work's findings, specifically the in the LLM safety domain.

Firstly, the finding that jailbreaks succeed by attenuating harmful intent signals rather than introducing competing benign signals could theoretically inform attack development. This risk is mitigated by several factors: the mechanism is described at a conceptual level without providing specific attack recipes, understanding how attacks succeed is necessary to build robust defenses, and similar mechanistic insights have appeared in prior work on representation engineering and activation steering.

Secondly, publishing specific SAE feature indices that detect harmful content could help adversaries craft inputs that avoid activating these features. However, the features are derived from Gemma Scope, a publicly released SAE, and their interpretations are already available through Neuronpedia. The large feature set (17 intent, 128 veto) makes targeted evasion non-trivial. More importantly, interpretable systems enable ongoing debugging: when new attacks emerge, engineers can identify which features failed and adapt accordingly—an advantage unavailable with opaque detection methods.

The benefits of interpretable safety research, including auditability, debuggability, and human oversight, are believed to outweigh the risks of publishing high-level mechanistic findings.

Acknowledgements

The authors utilized AI assistants during the preparation of this work. Claude (Anthropic) was used for code debugging, refining ideas, improving diagram quality, and editing written content. ChatGPT (OpenAI) was used for brainstorming and idea development. Google AI Studio was used to assist with the coding process. All AI-generated suggestions were critically reviewed and verified before incorporation. The authors assume full responsibility for the content and quality of the submitted work.

References

Anthropic Interpretability Team. 2024. Features as classifiers. Transformer Circuits. Accessed: November 15, 2024.

Trenton Bricken, Jonathan Marcus, Siddharth Mishra-Sharma, Meg Tong, Ethan Perez, Mrinank Sharma, Kelley Rivoire, and Thomas Henighan. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *arXiv preprint arXiv:2309.08600*.

Marcus Buckmann and Edward Hill. 2024. Logistic regression makes small LLMs strong and explainable classifiers. *arXiv preprint arXiv:2408.03414*.

Hyunsoo Cho, Hyung Jin Kim, Junyeob Kim, Sang-Woo Lee, Sang-goo Lee, Kang Min Yoo, and Taeuk Kim. 2023. Prompt-augmented linear probing: Scaling beyond the limit of few-shot in-context learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12709–12718.

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*.

Yi Dong, Ronghui Mu, Gaojie Jin, Yi Qi, Jinwei Hu, Xingyu Zhao, and Xiaowei Huang. 2024. Building guardrails for large language models. *arXiv preprint arXiv:2402.01822*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Danny Hernandez, Dario Amodei, Tom Brown, and Jack Clark. 2024. Using dictionary learning features as classifiers. Transformer Circuits.

Jack Gallifant, Shan Chen, Kuleen Sasse, Hugo Aerts, Thomas Hartvigsen, and Danielle S. Bitterman. 2025. Sparse autoencoder features for classifications and transferability. *arXiv preprint arXiv:2502.11367*.

Ling Gao, Jiaxin Geng, Xuan Zhang, Preslav Nakov, and Xi Chen. 2024. Shaping the safety boundaries: Understanding and defending against jailbreaks in large language models. *arXiv preprint arXiv:2412.17034*.

Gemma Team. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Seungju Han, Kavel Kim, Seunghyun Bae, and 1 others. 2024. WildGuard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. In *Proceedings of NeurIPS*.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, and Madian Khabsa. 2023. Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, Fan Yang, Mengnan Du, and Yongfeng Zhang. 2024. Exploring concept depth: How large language models acquire knowledge and concept at different layers? *arXiv preprint arXiv:2404.07066*.

Subhash Kantamneni, Josh Engels, Senthooran Rajamanoharan, and Neel Nanda. 2024. SAE probing: What is it good for? absolutely something! LessWrong.

Alyssa Lees, Vinh Q. Tran, Yi Tay, Jeffrey Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. 2022. A new generation of Perspective API: Efficient multilingual character-level transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3197–3207.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, Janos Kramar, Anca Dragan, Rohin Shah, and Neel Nanda. 2024. Gemma Scope: Open sparse autoencoders everywhere all at once on Gemma 2. *arXiv preprint arXiv:2408.05147*.

MarketsandMarkets. 2024. Large language model (LLM) market worth $36.1 billion by 2030.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372.

Neuronpedia. 2025. Neuronpedia: A platform for mechanistic interpretability.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*.

OpenAI. 2024. Moderation API.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc.

Nishan Pantha, Muthukumaran Ramasubramanian, Iksha Gurung, Manil Maskey, and Rahul Ramachandran. 2024. Challenges in guardrailing large language models for science. *arXiv preprint arXiv:2411.08181*.

Traian Rebedea, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. *arXiv preprint arXiv:2310.10501*.

Michael Sawtell, Tim Masterman, Sami Besen, and Joseph Brown. 2024. Lightweight safety classification using pruned language models. *arXiv preprint arXiv:2412.13435*.

Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-François Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Liu Kost, Christopher Carnahan, and

Jordan L. Boyd-Graber. 2023. Ignore this title and HackAPrompt: Exposing systemic vulnerabilities of LLMs through a global prompt hacking competition. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore.

Erfan Shayegani, Md Abdullah Al Mamun, Yu Fu, Pedram Zaree, Yue Dong, and Nael B. Abu-Ghazaleh. 2023. Survey of vulnerabilities in large language models revealed by adversarial attacks. *arXiv preprint arXiv:2310.10844*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. *arXiv preprint arXiv:2211.00593*.

Xuan Wang, Ziqi Ji, Wenxuan Wang, Zhicheng Li, Daoyuan Wu, and Shuai Wang. 2025. SoK: Evaluating jailbreak guardrails for large language models. *arXiv preprint arXiv:2506.10597*.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. 2025. AXBench: Steering LLMs? even simple baselines outperform sparse autoencoders. *arXiv preprint arXiv:2501.17148*.

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 *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 6377–6399, Suzhou, China. Association for Computational Linguistics.

Yuxuan Zhang, Shujian Li, and Yang Liu. 2025. A survey on sparse autoencoders: Interpreting the internal mechanisms of large language models. *arXiv preprint arXiv:2503.05613*.

Appendix

A System Architecture

Figure 3 presents the overall system architecture of the proposed framework at inference time

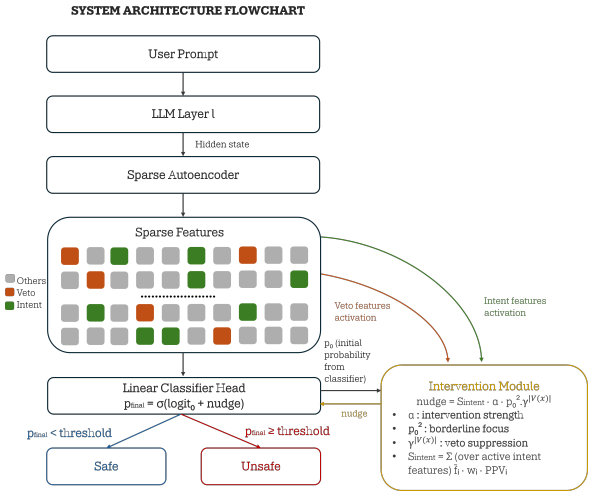

Figure 3: System architecture at inference time

B Dataset Statistics

Table 5 presents the dataset split statistics after preprocessing. Table 6 shows the distribution across harm categories. Preprocessing steps: (1) short responses under 10 tokens removed (268 samples), (2) rare category "others" filtered due to <1.5% representation (196 samples), (3) deduplication via random sampling reduced 37,470 to 18,522 unique prompts, (4) stratified 80/10/10 split by subcategory.

Table 5: Dataset split statistics after preprocessing.

Statistic	Train	Val	Test
Total samples	14,817	1,852	1,853
Harmful prompts	8,440	1,055	1,056
Benign prompts	6,377	797	797
Harmful (%)	57.0	57.0	57.0

Table 6: Harm category distribution (train set).

Category	Count
Benign	6,377
Social stereotypes & discrimination	1,448
False/misleading information	896
Toxic language & hate speech	750
Violence & physical harm	718
Sensitive info (organization/government)	702
Private info (individual)	592
Defamation & unethical actions	573
Sexual content	549
Fraud & illegal activities	532
Copyright violations	458
Mental health & over-reliance	451
Cyberattack	426
Material harm via misinformation	345

C Detailed Feature Analysis

Figure 4 maps intervention outcomes by intent score and veto count, revealing how the veto mechanism protects precision during recall improvement. Recovered false negatives (blue triangles) cluster at low veto counts. New false positives (orange crosses) occupy the same low-veto region. High veto counts protect true negatives (green circles) from false escalation.

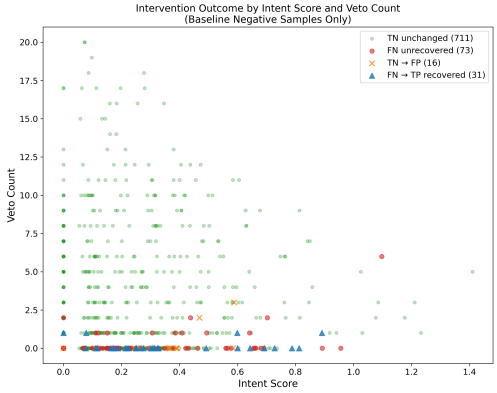

Figure 4: Intervention outcomes by intent score and veto count for baseline negative samples

Recovered false negatives cluster at veto counts of 0 to 1, where weak intent signal exists and no benign context blocks intervention. The exponential decay ($\gamma^{|\mathcal{V}|}$ with $\gamma = 0.35$) means even one or two veto features substantially attenuate the nudge. The 16 new false positives occupy the same low-veto region, yielding an approximately 2:1 ratio of recovered jailbreaks to added false positives. True negatives with high veto counts remain protected regardless of incidental intent signal. Unrecovered false negatives divide into two categories: samples with near-zero intent score (no signal to amplify) and samples with moderate intent but higher veto counts (conservative precision protection).

D Parameter Tuning Results

Table 7 presents the top parameter configurations from grid search over intervention strength (α), veto decay (γ), and prediction threshold (τ). Conservative selection chooses the lowest α that improves validation F1 by $\geq$0.5 points.

Table 7: Top 10 parameter configurations ranked by validation F1. Conservative selection chose α=6.0 (lowest improving α) rather than the oracle best.

α	γ	τ	Val F1	Test F1
15.0	0.40	0.47	0.9269	0.9259
15.0	0.45	0.47	0.9265	0.9259
15.0	0.50	0.47	0.9265	0.9259
15.0	0.30	0.47	0.9264	0.9263
15.0	0.35	0.47	0.9264	0.9259
15.0	0.35	0.45	0.9262	0.9241
9.0	0.30	0.46	0.9259	0.9266
9.0	0.35	0.46	0.9259	0.9257
9.0	0.40	0.46	0.9259	0.9253
9.0	0.45	0.46	0.9259	0.9253
Selected (conservative):				
6.0	0.40	0.45	0.9224	0.9263

The oracle best test F1 (0.9266) was achieved with α=9.0, γ=0.30, τ=0.46. However, higher α values risk overfitting to validation-specific patterns. The conservative selection (α=6.0) achieves comparable test performance (0.9263) with more moderate intervention strength, favoring robust generalization.

E Feature Interpretations

Tables 9 and 8 present the complete lists of intent and veto features with their Neuronpedia interpretations.

Table 8: Veto features (top 15 by PPV) with Neuronpedia interpretations. All features have PPV of 1.00 for benign content.

ID	Interpretation
676	Recipes or cooking instructions
3021	Legal citations, statutes, and regulations
2163	Software configuration terminology
2971	Code documentation and input references
2038	Single uppercase letters (variable/label names)
292	Incremental steps and gradual improvement
2597	Correctness and effectiveness signals
3006	Alternatives and substitutes
568	Missing or disappeared persons
2377	Hopelessness and despair expressions
742	Sentence boundary punctuation
1713	Beginning-of-sequence token
2653	Numeric ranges and measurements
3003	Named entities (people, organizations, dates)
1749	Necessity and emphasis words

Table 9: Intent features (causally validated) with Neuronpedia interpretations. PPV indicates positive predictive value for harmful content.

ID	PPV	Interpretation
4331	0.85	Mentions of villains, criminal groups, or evil/antagonistic characters
5583	0.92	Words indicating enabling, facilitating, or being complicit in wrongdoing
7196	0.81	Tokens signaling seriousness, severity, or high-impact consequences
11047	0.95	Abusive, insulting, derogatory, or disparaging language
8836	0.77	Negative sentiment (harms, losses, pain, punishment)
12847	0.86	Crime and criminal justice terminology
282	0.71	Solicitations or scams (prizes, fees, account activation)
9215	0.71	Espionage and intelligence agency references
11320	0.84	Cybersecurity and hacking content
11551	0.82	First-person offers to provide/send/share something
3932	0.89	Numeric and document-structure tokens
14752	0.79	Salient content words and proper nouns
16170	0.86	Technical or scientific terms and acronyms
1111	0.91	Speech acts and reported speech
12360	0.84	Recorded or directly quoted speech
4688	0.80	Names of groups or multi-part structures
5637	0.80	Named entities (people, places, organizations)

F Example of Neuronpedia Dashboard

Feature interpretations were obtained through Neuronpedia, an open platform that provides human-readable explanations of SAE features. For each feature, Neuronpedia displays AI-generated descriptions of activation patterns (this work used GPT-5-mini, token-level logit contributions, activation density distributions, and example text snippets with highlighted tokens that strongly activate the feature. Figure 5 shows an example dashboard for feature 4331, which detects mentions of villains, criminal groups, and antagonistic characters.

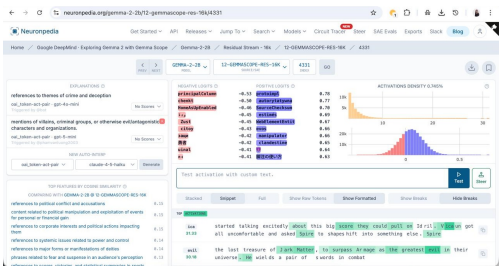

Figure 5: Example of Neuronpedia Dashboard for Feature 4331