Towards Understanding the Robustness of Sparse Autoencoders

Ahson Saiyed[1] **Sabrina Sadiekh**[2] **Chirag Agarwal**[1]
[1]University of Virginia [2]Independent Researcher

Abstract

Large Language Models (LLMs) remain vulnerable to optimization-based jailbreak attacks that exploit internal gradient structure. While Sparse Autoencoders (SAEs) are widely used for interpretability, their robustness implications remain underexplored. We present a study of integrating pretrained SAEs into transformer residual streams at inference time, without modifying model weights or blocking gradients. Across four model families (Gemma, LLaMA, Mistral, Qwen) and two strong white-box attacks (GCG, BEAST) plus three black-box benchmarks, SAE-augmented models achieve up to a 5× reduction in jailbreak success rate relative to the undefended baseline and reduce cross-model attack transferability. Parametric ablations reveal (i) a monotonic dose–response relationship between L_0 sparsity and attack success rate, and (ii) a layer-dependent defense–utility tradeoff, where intermediate layers balance robustness and clean performance. These findings are consistent with a representational bottleneck hypothesis: sparse projection reshapes the optimization geometry exploited by jailbreak attacks. [1]

1 Introduction

Large Language Models (LLMs) are increasingly deployed in safety-critical settings, yet they remain vulnerable to *jailbreaks*: carefully constructed inputs that elicit unsafe or policy-violating behavior (Yi et al., 2024). Many state-of-the-art jailbreak attacks explicitly exploit gradient-aligned directions in internal activations, suggesting that adversarial success is tightly coupled to the geometry of intermediate representations.

Recent advances in mechanistic interpretability have revealed that transformer residual streams are not amorphous vector spaces but exhibit structured latent features (Chatterley et al., 2025; Skean et al., 2024; Yugeswardeenoo et al., 2025). Sparse Autoencoders (SAEs) provide a practical method for decomposing dense activations into sparse, feature-aligned components that often correspond to interpretable concepts (Cunningham et al., 2023; Li et al., 2025b). By reparameterizing activations in a sparse basis, SAEs impose a structured bottleneck on the residual stream. However, the robustness implications of SAE-based representations remain unclear. Prior work has shown that SAE features can be sensitive to small perturbations (Li et al., 2025a), and that reconstruction errors may induce behavioral shifts when reinjected into the model (Gurnee, 2024). These findings raise concerns about using SAEs for monitoring or control, but they do not address a complementary question: *how does inserting sparse feature projections into LLM residual streams affect adversarial robustness under adaptive jailbreak attacks?*

Present Work. We conduct a controlled empirical study of inserting pretrained SAEs into transformer layers at inference time. The intervention does not modify model weights, does not block gradients, and preserves end-to-end differentiability, allowing us to isolate the representational effect of sparse projection on adversarial optimization dynamics.

We study SAE insertion across three open-source model families (Gemma, LLaMA, and Mistral). To further examine how robustness depends on insertion depth, we additionally analyze Qwen2.5 using a multi-layer SAE suite that enables controlled layer-placement experiments. Robustness is measured under strong white-box jailbreak attacks (GCG and BEAST) as well as black-box jailbreak benchmarks. Across evaluation settings, SAE-augmented models reduce jailbreak success rates by up to **5**× relative to the undefended baseline.

Beyond aggregate performance, we investigate how robustness varies with SAE configuration through sparsity and layer-placement ablations, and

[1]Code available at https://github.com/AikyamLab/sparse-jailbreak.

Findings of the Association for Computational Linguistics: ACL 2026, pages 26066–26086

analyze cross-model transferability of adversarial suffixes. Our results show that SAE routing systematically reduces attack transferability and alters the geometry of adversarial optimization, suggesting that structured sparse projections can act as lightweight representation-level defenses against optimization-based jailbreak attacks.

2 Related Work

Our work connects two lines of research: adversarial jailbreak attacks on LLMs and sparse autoencoders (SAEs) for mechanistic interpretability.

Adversarial Jailbreak Attacks. Early adversarial NLP work focused on discrete perturbations such as token substitutions (Ebrahimi et al., 2018) and paraphrasing (Alzantot et al., 2018). More recent jailbreak methods directly target alignment objectives. Zou et al. (2023) introduced Greedy Coordinate Gradient (GCG), a gradient-based adversarial suffix optimizer with strong cross-model transfer. BEAST improves attack efficiency via beam-search heuristics (Sadasivan et al., 2024). Extensions include prompt injection (Perez and Ribeiro, 2022), indirect exploits (Greshake et al., 2023), and automated multi-turn attacks (Reddy et al., 2025). These methods exploit the smooth internal optimization landscape of LLMs, enabling gradient-aligned suffix construction under white-box access.

Sparse Autoencoders for Interpretability. Mechanistic interpretability aims to decompose dense transformer activations into structured, interpretable components. SAEs reconstruct activations as sparse linear combinations of latent features (Bricken et al., 2023; Cunningham et al., 2023), and have been scaled to large models with high reconstruction fidelity (Gao et al., 2024; Templeton et al., 2024). SAEs enable feature discovery, attribution, and feature-level interventions (e.g., steering or concept erasure) (Marks et al., 2025; O'Brien et al., 2025).

Recent work has also identified limitations of SAE representations. Li et al. (2025a) show that SAE latents can be manipulated by small input perturbations, challenging their stability as monitoring tools. Gurnee (2024); Engels et al. (2025) demonstrate that reconstruction errors may induce structured semantic shifts. These findings raise open questions about the robustness properties of SAE-based representations when embedded into model pipelines.

LLM Defense Mechanisms. A variety of defenses have been proposed against jailbreak attacks (Yi et al., 2024). Randomized smoothing methods such as Erase-and-Check (Kumar et al., 2025), SmoothLLM (Robey et al., 2024), and Semantic Smoothing (Ji et al., 2024) apply stochastic perturbations or auxiliary paraphrasing models. Noise-based approaches such as Smoothed Embeddings (Hase et al., 2025) trade off attack success rate (ASR) and utility via input-level noise injection. Other methods intervene at decoding time (Zhao et al., 2026) or rely on adversarial training (Fu et al., 2026).

These approaches primarily evaluate aggregate ASR and utility trade-offs. In contrast, we study how inserting a sparse encode–decode bottleneck into the residual stream alters adversarial optimization dynamics and cross-model transferability. A detailed comparison with representative smoothing-, noise-, and decoding-based defenses is provided in Appendix F.

3 Method

We study an activation-space intervention against adversarial suffix attacks by inserting pretrained Sparse Autoencoders (SAEs) into intermediate transformer layers. The SAE acts as a sparse encode–decode operator applied to the residual stream during inference. Model weights remain unchanged and gradients are fully preserved, allowing white-box attackers to optimize directly through the SAE transformation.

Our evaluation spans four model families (Gemma-2, LLaMA-3, Mistral, and Qwen2.5) and scales from 2B to 70B parameters. Robustness is tested under strong white-box jailbreak attacks (GCG and BEAST) and black-box jailbreak benchmarks.

3.1 SAE-Augmented Language Models

Let $\mathbf{h}_\ell \in \mathbb{R}^d$ denote the residual stream activation at transformer layer ℓ. We insert a pretrained Sparse Autoencoder (SAE) that applies a sparse encode–decode transformation to the activation:

$$\hat{\mathbf{h}}_\ell = \mathcal{D}(\mathcal{E}(\mathbf{h}_\ell)),$$

where the encoder and decoder are defined as $\mathbf{z} = \mathcal{E}(\mathbf{h}) = \mathrm{ReLU}(W_{\text{enc}}\mathbf{h} + \mathbf{b}_{\text{enc}})$, $\hat{\mathbf{h}} = \mathcal{D}(\mathbf{z}) = W_{\text{dec}}\mathbf{z} + \mathbf{b}_{\text{dec}}$.

The latent representation $\mathbf{z} \in \mathbb{R}^{d_{\text{hidden}}}$ is sparse, with $d_{\text{hidden}} \gg d$ (typically a $16\times$ expansion). SAEs are trained offline to minimize reconstruction

error with an ℓ_1 sparsity penalty, yielding a sparse feature basis that approximately reconstructs the original residual activation.

Let f_θ denote the base language model and let $\Phi_\ell(\cdot)$ denote the SAE routing inserted at layer ℓ. The resulting SAE-augmented model is defined as

$$\tilde{f}_\theta(x) = f_\theta^{>\ell}\big(\Phi_\ell(f_\theta^{\leq\ell}(x))\big),$$

where $f_\theta^{\leq\ell}$ denotes the transformer prefix up to layer ℓ and $f_\theta^{>\ell}$ the remaining layers. The routing operator

$$\Phi_\ell(\mathbf{h}_\ell) = \mathcal{D}(\mathcal{E}(\mathbf{h}_\ell))$$

replaces the residual activation at that layer during inference:

$$\mathbf{h}_\ell \to \hat{\mathbf{h}}_\ell.$$

All subsequent transformer blocks therefore operate on the reconstructed activation $\hat{\mathbf{h}}_\ell$.

Importantly, the SAE layer remains fully differentiable. Under white-box attacks, adversarial suffix tokens x_{adv} are optimized to minimize the attack objective

$$\min_{x_{\text{adv}}} \mathcal{L}\big(\tilde{f}_\theta(x_{\text{prompt}} \oplus x_{\text{adv}})\big).$$

Because the routing function Φ_ℓ is differentiable, gradients propagate through the SAE reconstruction during optimization. For gradient-based attacks such as GCG, suffix tokens are updated using gradients that pass through the SAE layer at each step:

$$\nabla_{x_{\text{adv}}}\mathcal{L} = \frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}_\ell}\frac{\partial\hat{\mathbf{h}}_\ell}{\partial\mathbf{h}_\ell}\frac{\partial\mathbf{h}_\ell}{\partial x_{\text{adv}}}.$$

Thus adversarial optimization proceeds through the SAE-modified representation space rather than the original residual stream. In contrast, gradient-free attacks such as BEAST construct adversarial suffixes via beam search without accessing gradients; candidate suffixes are therefore evaluated directly on the SAE-augmented model $\tilde{f}_\theta$. While the optimization procedures differ, both attacks operate over the same modified representation induced by the SAE routing.

3.2 Model Suite

We evaluate SAE insertion across multiple open-source LLMs: Gemma-2 (2B, 9B, 27B), LLaMA-3 (8B, 70B), Mistral-7B, and Qwen2.5-7B.

For Gemma, LLaMA, and Mistral we insert a single SAE into an intermediate transformer layer (layer mappings in Appendix Table 7). To analyze the effect of insertion depth, we additionally evaluate Qwen2.5 using a multi-layer SAE suite that enables controlled layer-placement experiments.

3.3 Threat Model

We consider adversaries that append adversarial suffixes to prompts in order to induce harmful model outputs.

White-box attacks. We evaluate two strong jailbreak methods. **Greedy Coordinate Gradient (GCG)** (Zou et al., 2023) iteratively optimizes suffix tokens using gradients from a harmful objective. In our setting, gradients are backpropagated through the SAE layer, meaning the attacker directly optimizes through the sparse representation. **BEAST** (Sadasivan et al., 2024) is a beam-search-based jailbreak method that leverages model logits and internal signals. This provides a complementary non-gradient attack that still assumes white-box access.

Black-box attacks. We additionally evaluate robustness under query-only access using 1,500 jailbreak prompts drawn from Salad-Data (Li et al., 2024), Prompt Injections Benchmark, and SafeEval (Manczak et al., 2024).

3.4 Implementation Details

All models are run in PyTorch with bfloat16 precision on NVIDIA GPUs. Special tokens are excluded from activation analysis. We evaluate 24 HarmBench batches and report aggregated ASR and mechanistic metrics.

4 Experiments

We conduct a systematic empirical evaluation of SAE insertion as an inference-time intervention against adversarial suffix attacks. Our experiments address four research questions:

RQ1 (Effectiveness). Does inserting a pretrained SAE reduce attack success rates under adaptive white-box and black-box jailbreak attacks?

RQ2 (Transferability). How does SAE insertion affect the cross-model and cross-configuration transferability of adversarial suffixes?

RQ3 (Parametric Dependence). How do robustness outcomes depend on SAE configuration, including sparsity level (L_0) and insertion depth?

RQ4 (Optimization Dynamics). How does SAE insertion alter adversarial optimization dynamics and sparse feature usage?

4.1 Experimental Setup

Model suite. We evaluate SAE insertion across six open-source models spanning three families and a wide range of scales: Gemma-2 (2B, 9B, 27B), LLaMA-3 (8B, 70B), and Mistral-7B. For each model, a pretrained SAE is inserted into a single intermediate transformer layer following the procedure described in Section 3. Layer mappings are reported in Appendix Table 7.

To study the effect of insertion depth, we additionally perform controlled layer-placement experiments on Gemma-2-9B and Qwen2.5 using a multi-layer SAE suite.

Attack protocol. We evaluate robustness under both gradient-based and non-gradient white-box jailbreak attacks. For GCG, adversarial suffixes are optimized for 500 gradient steps with a fixed suffix length of 20 tokens. For BEAST, we use beam-search parameters $k_1{=}k_2{=}15$ and search depth $L{=}20$, following the original attack configuration.

Each prompt is evaluated under three configurations:

- PROMPT: clean harmful prompt without an adversarial suffix,
- BASE (**B**): suffix optimized against the baseline model,
- SAE (**S**): suffix optimized against the SAE-augmented model.

This protocol allows us to evaluate baseline vulnerability, adaptive attack performance, and cross-configuration transferability.

In addition to white-box attacks, we evaluate robustness under black-box access using three jailbreak datasets described in Section 3.3. In this setting, attackers interact with the model through prompt queries only.

4.2 Evaluation Metrics

Our primary metric is attack success rate (ASR), defined as the fraction of prompts that produce harmful outputs under evaluation. Harmful responses are detected using three independent evaluators: WildGuard, the HarmBench classifier, and a refusal-based heuristic (Appendix Table 10).

Beyond aggregate ASR, we analyze cross-model transferability of adversarial suffixes and examine representation-level changes induced by SAE insertion.

4.3 Mechanistic Analyses

To study how SAE routing alters adversarial representations and optimization dynamics, we perform two additional analyses.

Sparse feature analysis. We analyze SAE latent activations $\mathbf{z}$ induced by adversarial suffixes. For each prompt we extract the top-k activated features and measure similarity using the Jaccard index:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Gradient spectral analysis. To characterize adversarial optimization dynamics under GCG, we compute the singular value decomposition of the gradient matrix $G \in \mathbb{R}^{L \times d}$ for suffix length L. We measure effective rank, spectral gap, and inter-step cosine similarity to quantify changes in gradient concentration.

5 Results

RQ1: Does SAE insertion reduce attack success rates under adaptive white-box attacks?
White-box (GCG). We first evaluate the effectiveness of SAE insertion under adaptive white-box attacks, where adversarial suffixes are optimized directly against the evaluation model. As summarized in Table 1 and illustrated in Fig. 1, SAE-augmented models exhibit substantially lower attack success rates than their baseline counterparts. Across the six evaluated models, the median ASR decreases from 55.0% for baseline models to 19.05% for SAE-augmented models. This reduction is statistically significant under a two-sided Mann–Whitney U test ($p = 0.0087$). Paired ASR scores for both attacks are reported in App. B.
White-box (BEAST). A similar trend is observed under the non-gradient BEAST attack. Median ASR decreases from 19.35% for baseline models to 9.7% for SAE-augmented models (Table 1, Fig. 1). This reduction is statistically significant under the Mann–Whitney U test ($p = 0.0411$), indicating that SAE insertion improves robustness even for attacks that do not rely on gradient-based optimization.
Black-box benchmarks. We additionally evaluate SAE robustness under black-box jailbreak prompts using three external datasets (Appendix B.3). Across benchmarks, SAE-augmented models consistently exhibit lower ASR than their baseline counterparts, although the magnitude of improve-

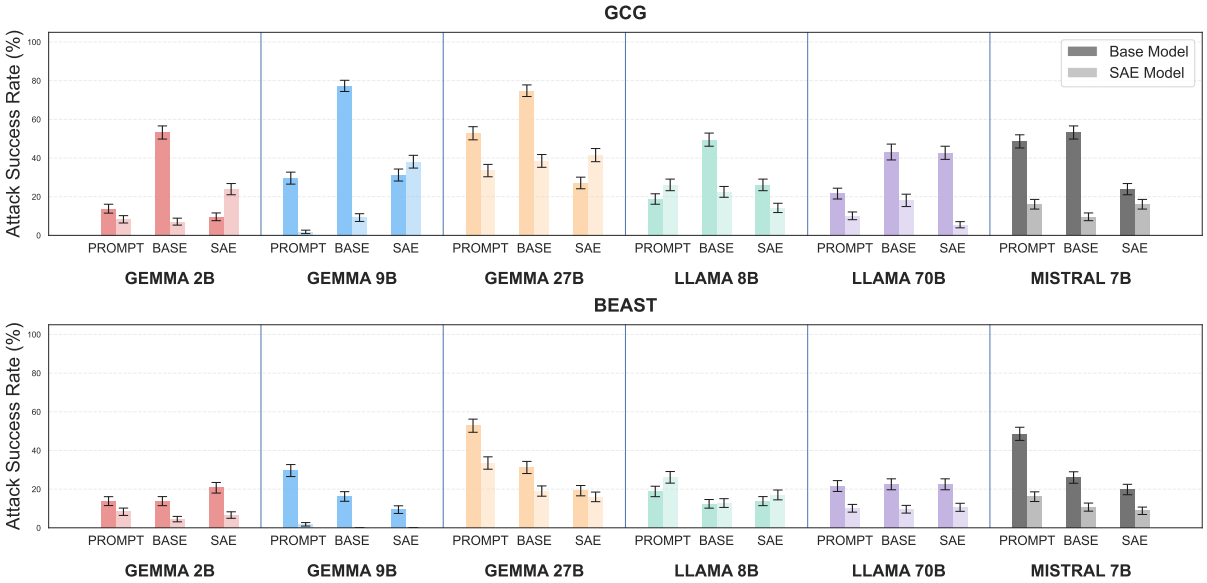

Figure 1: Attack success rate (ASR) for GCG and BEAST attacks on HarmBench across baseline (dark) and SAE-augmented (light) models. Results are shown for six models (Gemma-2 2B/9B/27B, LLaMA-3 8B/70B, Mistral-7B) under three configurations: PROMPT, BASE, and SAE. Under adaptive attacks (SAE), SAE-augmented models exhibit substantially lower ASR than their baseline counterparts. Under BASE transfer evaluation, median ASR decreases from 55.0% to 19.05% for GCG and from 19.35% to 9.7% for BEAST. Error bars denote standard error over HarmBench prompts ($n \approx 218$ per model–condition).

Table 1: Median ASR (%) for adversarial suffixes optimized against the evaluation model and evaluated on baseline versus SAE-augmented models ($N = 6$). Statistical significance is assessed using a two-sided Mann–Whitney U test. An asterisk (*) indicates statistical significance at $p < 0.05$.

Attack	BASE	SAE	MWU p
GCG	55.0	19.05	0.0087*
BEAST	19.35	9.7	0.0411*

ment varies depending on the evaluator and prompt distribution.

RQ2 (Transferability): SAE insertion reduces cross-model and cross-configuration transfer. We analyze transferability of adversarial suffixes under four source–target configurations: BASE→BASE (BB), SAE→SAE (SS), BASE→SAE (BS), and SAE→BASE (SB). We distinguish *same-configuration cross-model transfer* (BB and SS) from *cross-configuration transfer* (BS and SB). For BB and SS, we exclude same-model attacks (the main diagonal) and therefore evaluate only off-diagonal pairs; with six models this yields $6 \times 5 = 30$ directed source–target pairs. For BS and SB, each base/SAE variant of a model is treated as a distinct source/target, so we evaluate all off-diagonal cross-configuration pairs, yielding $6 \times 6 = 36$ directed pairs per direction.[2]

GCG. Under GCG, SAE insertion reduces cross-model transfer within the same configuration: SS exhibits a lower median ASR than BB (2.65% vs. 8.75%; Table 2), and the confidence intervals also shift downward (BB: [6.25, 13.75] vs. SS: [1.25, 8.20]). Cross-configuration transfer further reveals a pronounced asymmetry: suffixes optimized on BASE models transfer less effectively to SAE targets than SAE-optimized suffixes transfer to base targets (BS median 7.50%, CI [2.50, 10.00] vs. SB median 12.65%, CI [11.00, 17.90]). Notably, the BS interval lies substantially below SB, indicating that SAE insertion weakens transferred attacks from baseline models while SAE-optimized suffixes remain comparatively more transferable to baseline targets.

BEAST. A similar pattern holds for the non-gradient BEAST attack. Within-class cross-model transfer is lower for SAE models (SS median 5.35%) than for baseline models (BB median 15.25%), with non-overlapping shifts in the corresponding confidence intervals (BB: [11.85, 19.30] vs. SS: [3.15, 9.15]). Cross-configuration transfer is again asymmetric: BS achieves a median ASR of 8.20% (CI [5.20, 11.00]) whereas SB achieves

[2]All transfer analyses exclude same-model attacks; see Appendix for full matrices.

Table 2: Cross-model transfer of median ASR (%) for GCG and BEAST. BB and SS exclude same-model attacks and therefore contain $6 \times 5 = 30$ directed off-diagonal pairs. BS and SB evaluate all directed cross-configuration pairs ($6 \times 6 = 36$). Confidence intervals are 95% bootstrap intervals over model pairs.

Attack	Transfer	ASR	95% CI	N
GCG	B→B	8.75	[6.25, 13.75]	30
	S→S	2.65	[1.25, 8.20]	30
	B→S	7.50	[2.50, 10.00]	36
	S→B	12.65	[11.00, 17.90]	36
BEAST	B→B	15.25	[11.85, 19.30]	30
	S→S	5.35	[3.15, 9.15]	30
	B→S	8.20	[5.20, 11.00]	36
	S→B	11.90	[9.40, 16.95]	36

11.90% (CI [9.40, 16.95]). Compared to GCG, the asymmetry is weaker but still consistent in direction, suggesting that SAE insertion reduces transferability from baseline to defended targets even when the attacker does not rely on dense gradients.

Overall, these results show that SAE insertion reduces cross-model transfer (BB→SS shift) and induces an asymmetric cross-configuration transfer pattern (BS $<$ SB), consistent with the view that SAE routing changes the geometry of transferable adversarial solutions rather than acting as output filtering.

RQ3 (Parametric Dependence): robustness varies with sparsity and insertion depth. We perform ablations over two key parameters: sparsity level (L_0) and insertion depth. We focus on model families for which pretrained open-source SAE suites are available (Gemma-2, LLaMA-3.1), enabling controlled interventions without retraining. Because our primary robustness signal arises from adversarial transfer, we focus on transfer settings (BASE→SAE and SAE→BASE) rather than within-model attacks.

Sparsity ablation. We vary SAE sparsity while keeping the insertion layer fixed and evaluate its effect on adversarial transfer. Across both Gemma-2 9B and LLaMA-3.1-8B-Instruct, we observe a consistent monotonic relationship between sparsity and robustness.

For Gemma-2 9B, increasing L_0 (lower sparsity) leads to a substantial degradation in robustness: BASE→SAE ASR increases from 0.9% at $L_0 = 11$ to 21.2% at $L_0 = 310$ (Table 3). A similar trend holds for reverse transfer, where SAE→BASE ASR increases from 26.0% to 43.8%.

We observe the same pattern on LLaMA-3.1-8B-Instruct when varying the number of active features k (a proxy for effective sparsity). Under BASE→SAE, ASR remains low at 0.5% for $k = 32$ and $k = 64$, but increases to 2.3% for $k = 128$ and $k = 256$ (Table 4). Reverse transfer again follows the same trend, with SAE→BASE ASR increasing from 35.0% to 48.2%.

Table 3: Effect of SAE sparsity (L_0) on transfer ASR (%) for Gemma-2 9B (Layer 20, Width 16K). Each cell reports ≈218 prompts. The monotonic increase in B→S ASR with larger L_0 indicates that stronger sparse compression more effectively disrupts transferable adversarial representations.

L_0	B→S	S→B
11	**0.9**	26.0
36	3.2	18.1
68	14.4	28.8
138	15.6	35.8
310	21.2	**43.8**

Table 4: Effect of SAE sparsity (controlled via top-k active features) on transfer ASR (%) for LLaMA-3.1-8B-Instruct (Layer 19, Width 131K). Increasing k (lower sparsity) degrades robustness, consistent with Gemma results.

k	B→S	S→B
32	**0.5**	35.0
64	**0.5**	38.2
128	2.3	43.1
256	2.3	**48.2**

Together, these results reveal a consistent dose–response relationship: stronger sparse compression more effectively disrupts transferable adversarial representations, while relaxing sparsity restores transferability across models.

Layer placement ablation. We next vary insertion depth while keeping sparsity fixed, and observe a consistent dependence of robustness on representational depth across both Gemma-2 9B and LLaMA-3.1-8B-Instruct (with similar trends observed on Qwen2.5; see App. C).

Early-to-mid layers yield the strongest suppression of transfer attacks. For Gemma-2 9B, BASE→SAE ASR drops to 0.9% at Layer 5 (Table 5), while for LLaMA-3.1-8B-Instruct the best-performing layer is Layer 7 with 0.0% ASR (Table 6). This indicates that intervening before adversarial features are fully formed can significantly disrupt transfer.

However, very early insertion can introduce behavioral side effects. In Gemma, early layers

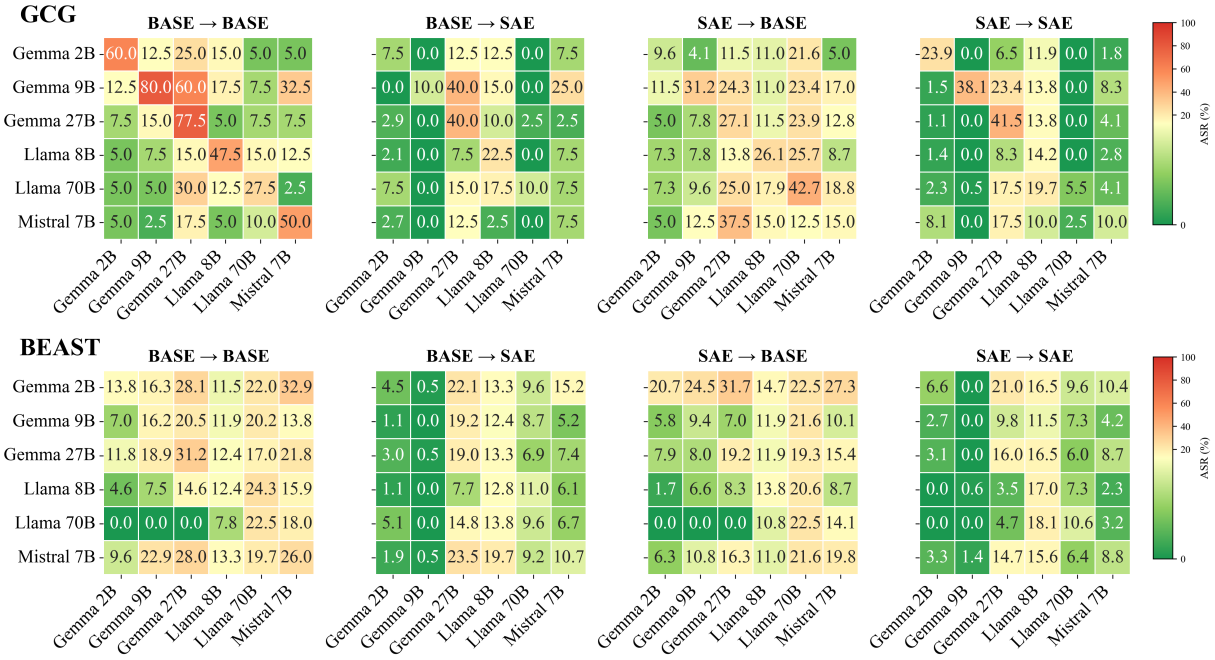

Figure 2: Attack Success Rate (ASR) transfer matrices for GCG (top) and BEAST (bottom) attacks across evaluation models. Rows correspond to the suffix source model used to generate adversarial suffixes, while columns denote the evaluation (target) models. Values are normalized to the range $[0, 100]$, where higher values indicate a higher attack success rate.

increase No Suffix ASR (12.2% @ Layer 5), while in LLaMA late layers exhibit even stronger degradation, with No Suffix ASR rising to 23.9% @ Layer 27. Intermediate layers provide a more favorable tradeoff. For example, in Gemma, Layer 20 maintains low transfer ASR (13.9%) while preserving clean behavior, and in LLaMA, Layers 11–19 achieve similarly low transfer with minimal degradation.

In contrast, late-layer insertion substantially weakens the defense. In Gemma, BASE→SAE ASR increases to 51.4% at Layer 35, while in LLaMA it rises to 7.3% at Layer 23 and remains elevated for deeper layers. Reverse transfer (SAE→BASE) also increases with depth in both models, indicating that adversarial signals become more stable in later representations.

Overall, these results reveal a consistent layer-dependent defense–utility tradeoff: early layers maximize robustness but may perturb model behavior, intermediate layers offer the best balance, and late layers are largely ineffective.

RQ4 (Optimization Dynamics): SAE insertion alters adversarial feature usage and gradient dynamics. To understand why SAE insertion reduces adversarial robustness and transferability, we analyze how it affects both the representation space used by attacks and the optimization dynamics of adversarial suffix generation.

Table 5: Effect of SAE layer placement on transfer ASR (%) for Gemma-2 9B (Width 16K). **B** denotes suffixes optimized on the base model and **S** suffixes optimized on the SAE model. BASE→BASE baseline ASR = 77.3%. Each cell reports ≈218 prompts.

Layer	B→S	S→B
5	**0.9**	13.4
10	3.7	15.3
20	13.9	30.5
30	8.1	40.6
35	51.4	**53.2**

Sparse feature structure of adversarial suffixes.

We first examine which SAE features are activated by adversarial suffixes. For each suffix we extract the top-k activated SAE features and measure overlap using the Jaccard index. As shown in Fig. 3, adversarial suffixes exhibit substantially higher feature overlap with each other (0.36 ± 0.16) than with random text baselines (0.09 ± 0.05), a $4\times$ difference ($p < 0.001$). This indicates that adversarial optimization does not produce arbitrary token perturbations, but instead converges on a shared sparse feature subspace.

Within model families, feature overlap is consistently higher than across families, suggesting

Table 6: Effect of SAE layer placement on ASR (%) for LLaMA-3.1-8B-Instruct (Width 131K). Early layers provide strongest robustness, while deeper layers degrade both robustness and clean performance.

Layer	S→S	B→S	S→B	No Suffix
3	0.0	0.5	19.7	0.0
7	0.5	**0.0**	34.4	0.0
11	1.4	0.9	**45.9**	0.0
15	0.9	0.5	39.2	0.0
19	0.0	0.5	35.9	0.9
23	5.5	7.3	38.1	10.1
27	4.6	6.0	28.6	23.9

that different model architectures learn partially aligned but not identical adversarial feature representations. These structured feature patterns provide a mechanistic explanation for the cross-model transferability observed in RQ2.

Gradient dynamics during adversarial optimization. We next analyze gradient trajectories during GCG optimization. For each attack run we compute spectral statistics of the token-level gradient matrix across optimization steps (Table 18; Fig. 5).

Across both LLaMA-8B and Gemma-9B, SAE insertion substantially alters optimization dynamics. Gradients become significantly more correlated between successive optimization steps: the cosine similarity between gradients increases from 0.50 ± 0.15 to 0.80 ± 0.09 for LLaMA-8B ($p = 10^{-36}$). At the same time, effective rank decreases ($1.88 \rightarrow 1.72$) while the spectral gap σ_1/σ_2 increases ($24.0 \rightarrow 28.2$), indicating that gradient signal becomes concentrated into fewer dominant directions.

These changes coincide with a substantial increase in final attack loss ($0.82 \rightarrow 1.08$, $p = 10^{-20}$) and visibly slower convergence during optimization (Fig. 4). Similar patterns are observed for Gemma-9B (Table 18), suggesting that these effects are consistent across model families.

Together, these results suggest that SAE routing imposes a representational bottleneck on adversarial optimization. Attacks converge on a shared sparse feature subspace, but the SAE projection compresses gradient signal and increases optimization stagnation, making it harder for gradient-based methods to discover effective adversarial suffixes.

6 Discussion

Our results demonstrate that inserting pretrained SAEs into transformer residual streams systematically improves adversarial robustness. Across model families, SAE-augmented models exhibit lower jailbreak success rates and reduced cross-model transfer under strong white-box attacks. These effects persist despite full gradient access during optimization, indicating that robustness does not arise from gradient masking, but from changes to the underlying representation space.

Representation-level defenses. Unlike prior defenses that operate at the input or output level (e.g., paraphrasing, smoothing, or filtering), SAE insertion modifies intermediate activations. Even without retraining model weights, this lightweight intervention significantly reduces transferability of adversarial suffixes, suggesting that sparse projection disrupts the structure of transferable adversarial representations.

Role of sparsity and depth. Robustness depends systematically on SAE configuration. Increasing sparsity (lower L_0) yields a clear dose–response effect, with stronger compression leading to lower transfer ASR. Insertion depth introduces a defense–utility tradeoff: early layers provide the strongest robustness but may affect clean behavior, intermediate layers offer a balanced regime, and late layers are largely ineffective. This pattern is consistent across architectures (Gemma, LLaMA, and Qwen), indicating that it reflects general properties of transformer representations.

Mechanistic interpretation. Feature-level analysis shows that adversarial suffixes concentrate in a shared sparse feature subspace, explaining their cross-model transferability. SAE insertion disrupts this structure. Complementary gradient analysis shows increased inter-step correlation, reduced effective rank, and larger spectral gaps, consistent with a contraction of the effective optimization subspace. Together, these results support a representational bottleneck hypothesis: sparse projection constrains the geometry available to adversarial optimization.

Optimization budget ablation. To assess whether extended optimization can overcome the SAE defense, we compare GCG suffixes optimized for 500 and 1500 steps on the same 50 prompts across Gemma-2 9B and LLaMA-3.1-8B-Instruct. While SAE-optimized suffixes gain substantially from additional steps—particularly in reverse transfer, where SAE→Base ASR rises from 32.0% to 38.0% for Gemma and from 24.0% to 64.0% for LLaMA—the SAE defense against base-optimized suffixes remains stable or even strengthens (Base→SAE: 10.0%→4.0% for

Gemma, 34.0%→34.0% for LLaMA), indicating that the representational bottleneck imposed by sparse routing is not easily bypassed by longer optimization alone.

Black-box evaluation. In black-box settings, robustness improvements depend on the evaluation protocol. SAE integration reduces ASR under classifier-based evaluation (HarmBench), while effects under other detectors are smaller. This highlights that robustness conclusions depend on how harmful behavior is defined and measured.

7 Conclusion

We study the robustness implications of inserting pretrained Sparse Autoencoders into transformer residual streams at inference time. Across multiple model families and scales, SAE routing consistently reduces jailbreak success rates under strong white-box attacks and decreases cross-model transferability of adversarial suffixes. Mechanistic analyses suggest that these effects arise from a representational bottleneck: sparse projection alters the geometry of adversarial optimization and concentrates gradient signal into fewer directions.

These findings indicate that tools originally developed for mechanistic interpretability can also function as lightweight robustness interventions. More broadly, our results suggest that representation-level operators provide a promising direction for improving adversarial robustness in large language models without requiring retraining or gradient blocking.

8 Limitations

Our study focuses on discrete adversarial suffix attacks and evaluates SAE insertion under a single SAE placement per model. While we conduct targeted ablations over sparsity and insertion depth for selected models, these analyses do not cover the full model suite. As a result, defense effectiveness may depend on SAE configuration choices—including layer placement, dictionary width, and SAE training procedure—that were not exhaustively explored across all architectures.

Additionally, our evaluation focuses on optimization-based jailbreak attacks. Although we consider both gradient-based (GCG) and non-gradient (BEAST) attacks, future work should examine whether similar robustness effects hold under broader threat models, including multi-turn attacks, prompt injection strategies, or adaptive attackers that explicitly optimize against SAE-induced representations.

9 Ethics Statement

This work studies adversarial attacks on LLMs to develop defenses, not to enable harm. All experiments use established safety benchmarks (HarmBench) designed for red-teaming research. We do not release attack code or successful adversarial suffixes. Our findings aim to improve LLM safety by demonstrating that interpretability tools can serve as lightweight robustness interventions. We acknowledge that adversarial robustness research carries dual-use risks, but believe the defensive contributions outweigh potential misuse.

Acknowledgements

We would like to thank all the anonymous reviewers of ACL for their valuable feedback. A.S. is supported by the Fellowship in AI Research from LaCross Institute for Ethical AI in Business. C.A. is supported, in part, by grants from Capital One, LaCross Institute for Ethical AI in Business, the UVA Environmental Institute, OpenAI Researcher Program, Thinking Machine's Tinker Research Grant, and Cohere. The views expressed are those of the authors and do not reflect the official policy or the position of the funding agencies.

References

Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *Preprint*, arXiv:2308.14132.

Moustafa Alzantot, Yash Sharma, and Ahmed Elgohary. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

Trenton Bricken, Adly Templeton, and Joshua Batson. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *arXiv preprint arXiv:2310.18494*.

Daniel Chatterley, Zachary Molyneux, Felix Ashworth, Benedict Sutherlands, Andrew Scolto, and Travis Connor. 2025. Contextual lattice probing for large language models: A study of interleaved multi-space activation patterns.

Hoagy Cunningham, Aidan Turner, and Michael Sapienza. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Javid Ebrahimi, Anyi Rao, and Daniel Lowd. 2018. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Joshua Engels, Logan Smith, and Max Tegmark. 2025. Decomposing the dark matter of sparse autoencoders. *arXiv preprint arXiv:2410.14670v2*. Published in Transactions on Machine Learning Research.

Shaopeng Fu, Liang Ding, Jingfeng Zhang, and Di Wang. 2026. Short-length adversarial training helps llms defend long-length jailbreak attacks: Theoretical and empirical evidence. *Preprint*, arXiv:2502.04204.

Dan Gao, Joseph Lieber, and Nazneen Rajani. 2024. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. *Preprint*, arXiv:2302.12173.

Wes Gurnee. 2024. Sae reconstruction errors are (empirically) pathological. *AI Alignment Forum*. Accessed: 2025-01-10.

Ryo Hase, Md Rafi Ur Rashid, Ashley Lewis, Jing Liu, Toshiaki Koike-Akino, Kieran Parsons, and Ye Wang. 2025. Smoothed embeddings for robust language models. *Preprint*, arXiv:2501.16497.

Jiabao Ji, Bairu Hou, Alexander Robey, George J. Pappas, Hamed Hassani, Yang Zhang, Eric Wong, and Shiyu Chang. 2024. Defending large language models against jailbreak attacks via semantic smoothing. *Preprint*, arXiv:2402.16192.

Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Aaron Jiaxun Li, Soheil Feizi, and Himabindu Lakkaraju. 2025. Certifying llm safety against adversarial prompting. *Preprint*, arXiv:2309.02705.

Aaron J. Li, Suraj Srinivas, Usha Bhalla, and Himabindu Lakkaraju. 2025a. Interpretability illusions with sparse autoencoders: Evaluating robustness of concept representations. *Preprint*, arXiv:2505.16004.

Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. 2024. Salad-bench: A hierarchical and comprehensive safety benchmark for large language models. *arXiv preprint arXiv:2402.05044*.

Yuxiao Li, Eric J. Michaud, David D. Baek, Joshua Engels, Xiaoqing Sun, and Max Tegmark. 2025b. The geometry of concepts: Sparse autoencoder feature structure. *Preprint*, arXiv:2410.19750.

Blazej Manczak, Eliott Zemour, Eric Lin, and Vaikkunth Mugunthan. 2024. Primeguard: Safe and helpful llms through tuning-free routing. *arXiv preprint arXiv:2407.16318*.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2025. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *Preprint*, arXiv:2403.19647.

Kyle O'Brien, David Majercak, Xavier Fernandes, Richard Edgar, Blake Bullwinkel, Jingya Chen, Harsha Nori, Dean Carignan, Eric Horvitz, and Forough Poursabzi-Sangdeh. 2025. Steering language model refusal with sparse autoencoders. *Preprint*, arXiv:2411.11296.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *Preprint*, arXiv:2211.09527.

Aashray Reddy, Andrew Zagula, and Nicholas Saban. 2025. Autoadv: Automated adversarial prompting for multi-turn jailbreaking of large language models. *Preprint*, arXiv:2507.01020.

Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. 2024. Smoothllm: Defending large language models against jailbreaking attacks. *Preprint*, arXiv:2310.03684.

Vinu Sankar Sadasivan, Shoumik Saha, Gaurang Sriramanan, Priyatham Kattakinda, Atoosa Chegini, and Soheil Feizi. 2024. Fast adversarial attacks on language models in one gpu minute. *arXiv preprint arXiv:2402.15570*.

Oscar Skean, Md Rifat Arefin, Yann LeCun, and Ravid Shwartz-Ziv. 2024. Does representation matter? exploring intermediate layers in large language models. *Preprint*, arXiv:2412.09563.

Adly Templeton, Tom Conerly, and Gary Marcus. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *arXiv preprint arXiv:2406.10142*.

Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. 2024. Jailbreak attacks and defenses against large language models: A survey. *Preprint*, arXiv:2407.04295.

Dharunish Yugeswardeenoo, Harshil Nukala, Ved Shah, Cole Blondin, Sean O Brien, Vasu Sharma, and Kevin Zhu. 2025. Interpreting the latent structure of operator precedence in language models. *Preprint*, arXiv:2510.13908.

Yinzhi Zhao, Ming Wang, Shi Feng, Xiaocui Yang, Daling Wang, and Yifei Zhang. 2026. Defending large language models against jailbreak attacks via in-decoding safety-awareness probing. *Preprint*, arXiv:2601.10543.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

A Experimental Setup

A.1 SAE Integration Details

Table 7: Canonical SAE configurations used in main experiments (Section 5, RQ1–RQ2). Each model uses a single pretrained SAE inserted at one intermediate layer.

Model	SAE Release	Layer	d_{model}	Width
Gemma-2-2B	`gemma-scope-2b-pt-res`	12	2304	16K
Gemma-2-9B	`gemma-scope-9b-pt-res`	19	3584	16K
Gemma-2-27B	`gemma-scope-27b-pt-res`	34	4608	131K
Llama-3.1-8B	`Goodfire/Llama-3.1-8B-Instruct-SAE-l19`	19	4096	65K
Llama-3.3-70B	`Goodfire/Llama-3.3-70B-Instruct-SAE-l50`	50	8192	65K
Mistral-7B	`JoshEngels/Mistral-7B-Residual-Stream-SAEs`	16	4096	65K

Table 8: SAE configurations used in parametric ablations (Section 5, RQ3). Layer-placement ablations vary insertion depth at fixed sparsity; sparsity ablations vary L_0 (Gemma) or TopK k (LLaMA) at a fixed layer. For LLaMA-3.1-8B ablations, we use the `andyrdt` SAE suite, which provides multiple matched-training layer and sparsity variants not available in the canonical release.

Ablation	Model	SAE Release	Layers / Sparsity	Width
Layer placement	Gemma-2-9B	`gemma-scope-9b-pt-res`	{5, 10, 20, 30, 35}	16K
	Llama-3.1-8B	`andyrdt/saes-llama-3.1-8b-instruct`	{3, 7, 11, 15, 19, 23, 27}	131K
	Qwen2.5-7B	`andyrdt/saes-qwen2.5-7b-instruct`	{3, 7, 11, 15, 19, 23, 27}	131K
Sparsity	Gemma-2-9B	`gemma-scope-9b-pt-res`	$L_0 \in \{11, 36, 68, 138, 310\}$	16K
	Llama-3.1-8B	`andyrdt/saes-llama-3.1-8b-instruct`	$k \in \{32, 64, 128, 256\}$	131K

Table 9: Deployment memory overhead of SAE insertion. We report peak GPU memory during generation (bfloat16, 128-token prompt, 128 generated tokens, batch size 1) for each base model alone versus the base model with the canonical SAE from Table 7 inserted at the corresponding layer. "SAE (bf16)" denotes the overhead in the measured deployment configuration; "SAE (fp32)" denotes the equivalent overhead if the SAE were stored in full precision. Latency is the mean generation time per token across 5 trials. Measurements performed on a single NVIDIA A100-SXM4-80GB.

Model	Width	Base (GB)	SAE (bf16)	SAE (fp32)	Latency	Δ
Gemma-2-2B	16K	5.27	0.15 (2.87%)	0.30 (5.74%)	29.84 ms/tok	+0.7%
Gemma-2-9B	16K	18.63	0.23 (1.25%)	0.47 (2.51%)	47.87 ms/tok	+0.3%
Gemma-2-27B	131K	54.61	2.49 (4.56%)	4.83 (8.84%)	67.90 ms/tok	+0.7%
Llama-3.1-8B	65K	16.11	1.09 (6.77%)	2.15 (13.38%)	27.56 ms/tok	+1.2%
Mistral-7B	65K	14.54	1.10 (7.54%)	2.15 (14.77%)	32.55 ms/tok	+0.5%

A.2 Refusal Patterns

Table 10: Refusal detection pattern categories. Full regex patterns available in supplementary code.

Category	Example Patterns
Inability	`i can't/cannot help/assist/answer`
Permission	`i am not allowed/permitted/authorized`
Discomfort	`i don't feel comfortable/ok`
Policy	`my rules/guidelines prevent/prohibit me`
Decline	`i must/have to decline/refuse`
Apology	`(i) apologize/sorry, but...`
Appropriateness	`it's not appropriate/illegal`
Request rejection	`this request is unsafe/inappropriate`

B Additional ASR Results

B.1 GCG: Full ASR Tables

Table 11: Complete attack success rate (ASR) results for GCG attacks across all models and configurations. Base = baseline model ASR; SAE = SAE-augmented model ASR; Δ = absolute change (percentage points); Rel. = relative change (%). Negative Δ indicates SAE reduced ASR; positive indicates SAE increased ASR.

Model	Config	Base ASR (%)	SAE ASR (%)	Δ (pp)	Rel. (%)
Gemma 2B	PROMPT	13.8±2.3	8.3±1.9	−5.5	−40.0
	BASE	53.2±3.4	7.1±1.8	−46.1	−86.6
	SAE	9.6±2.0	23.9±2.9	+14.2	+147.6
Gemma 9B	PROMPT	29.6±3.1	1.8±0.9	−27.8	−93.8
	BASE	77.3±2.9	9.2±2.0	−68.1	−88.1
	SAE	31.2±3.1	38.1±3.3	+6.9	+22.1
Gemma 27B	PROMPT	52.8±3.4	33.5±3.2	−19.3	−36.5
	BASE	74.8±3.0	38.5±3.3	−36.2	−48.5
	SAE	27.1±3.0	41.5±3.4	+14.4	+53.2
LLaMA 8B	PROMPT	18.8±2.7	26.1±3.0	+7.3	+39.0
	BASE	49.5±3.4	22.5±2.8	−27.1	−54.6
	SAE	26.1±3.0	14.2±2.4	−11.9	−45.6
LLaMA 70B	PROMPT	21.6±2.8	10.1±2.0	−11.5	−53.2
	BASE	43.1±4.1	18.1±3.2	−25.0	−58.1
	SAE	42.7±3.4	5.5±1.6	−37.2	−87.1
Mistral 7B	PROMPT	48.6±3.4	16.1±2.5	−32.6	−67.0
	BASE	53.2±3.4	9.6±2.0	−43.6	−81.9
	SAE	23.9±2.9	16.1±2.5	−7.8	−32.7
Overall	PROMPT	30.9±1.3	16.0±1.0	−14.9	−48.2
	BASE	59.4±1.4	17.5±1.1	−41.9	−70.5
	SAE	26.8±1.2	23.2±1.2	−3.6	−13.4
All Conditions		38.6±0.8	18.9±0.6	−19.7	−51.0

B.2 BEAST: Full ASR Tables

Table 12: Complete attack success rate (ASR) results for BEAST attacks across all models and configurations. Base = baseline model ASR; SAE = SAE-augmented model ASR; Δ = absolute change (percentage points); Rel. = relative change (%). Negative Δ indicates SAE reduced ASR; positive indicates SAE increased ASR.

Model	Config	Base ASR (%)	SAE ASR (%)	Δ (pp)	Rel. (%)
Gemma 2B	PROMPT	13.8	8.3	−5.5	−39.9
	BASE	13.8	4.5	−9.3	−67.4
	SAE	20.7	6.6	−14.1	−68.1
Gemma 9B	PROMPT	29.6	1.8	−27.8	−93.9
	BASE	16.2	0.0	−16.2	−100.0
	SAE	9.4	0.0	−9.4	−100.0
Gemma 27B	PROMPT	52.8	33.5	−19.3	−36.6
	BASE	31.2	19.0	−12.2	−39.1
	SAE	19.2	16.0	−3.2	−16.7
LLaMA 8B	PROMPT	18.8	26.1	+7.3	+38.8
	BASE	12.4	12.8	+0.4	+3.2
	SAE	13.8	17.0	+3.2	+23.2
LLaMA 70B	PROMPT	21.6	10.1	−11.5	−53.2
	BASE	22.5	9.6	−12.9	−57.3
	SAE	22.5	10.6	−11.9	−52.9
Mistral 7B	PROMPT	48.6	16.1	−32.5	−66.9
	BASE	26.0	10.7	−15.3	−58.8
	SAE	19.8	8.8	−11.0	−55.6
Overall	PROMPT	30.9	16.0	−14.9	−48.2
	BASE	20.4	9.4	−11.0	−53.9
	SAE	17.6	9.8	−7.8	−44.3
All Conditions		23.0	11.7	−11.3	−49.1

B.3 Black-box results

Black-box results. In the black-box setting, SAE integration exhibits detector-dependent effects. As summarized in Table 13, SAE reduces ASR under the HarmBench classifier, with mean ASR decreasing from $0.232 \rightarrow 0.160$ and median ASR from $0.205 \rightarrow 0.116$. Under WildGuard, reductions are more modest (mean $0.448 \rightarrow 0.414$, median $0.440 \rightarrow 0.405$) and fall within overlapping confidence intervals. Under the refusal-based heuristic, ASR remains extremely high (mean $\approx$ 96–97%, median $\approx 99\%$) with no meaningful difference between baseline and SAE models. Overall, these results show that SAEs exhibit emerging robustness properties compared to their vanilla counterparts and black-box robustness depends on the operational definition of harmful behavior used by the evaluator.

Table 13: Attack Success Rate (ASR) under black-box attacks across safety detectors. "± CI" denotes half-width of the 95% confidence interval.

Detector	Exp	Mean ASR	Med. ASR
refusal	B	$0.960_{\pm 0.030}$	$0.992_{\pm 0.031}$
refusal	S	$0.969_{\pm 0.023}$	$0.987_{\pm 0.013}$
harmbench	B	$0.232_{\pm 0.068}$	$0.205_{\pm 0.109}$
harmbench	S	$0.160_{\pm 0.068}$	$0.116_{\pm 0.093}$
wildguard	B	$0.448_{\pm 0.068}$	$0.440_{\pm 0.067}$
wildguard	S	$0.414_{\pm 0.073}$	$0.405_{\pm 0.081}$

C Cross-family validation (Qwen2.5-7B)

To assess whether the observed sparsity and layer-dependent trends generalize beyond the primary model families, we evaluate Qwen2.5-7B-Instruct using a publicly available multi-layer SAE suite for a fixed checkpoint. This setup enables controlled variation of insertion depth without modifying model weights or attack procedures. SAE insertion reduces adversarial transfer under BASE→SAE. At Layer 19, ASR decreases from 58.8% to 32.4% (Table 14). As in Gemma and LLaMA, transfer remains asymmetric, with SAE→BASE attacks exhibiting higher success rates. Varying insertion depth reveals a strong dependence on layer position (Table 15). Early-to-mid layers provide the strongest robustness (Layer 7: 0.0% BASE→SAE), while later layers show substantially weaker effects (e.g., 59.6% at Layer 23). Overall, these results are consistent with the main findings: robustness is maximized when SAE routing is applied at early-to-intermediate layers and degrades for deeper insertions, suggesting that adversarial representations become more stable in later transformer layers.

Evaluation Model	BASE Suffix	SAE Suffix
BASE	58.8	56.6
SAE (L19)	32.4	38.2

Table 14: Cross-configuration transfer matrix for Qwen2.5-7B-Instruct with an SAE inserted at Layer 19 (GCG, 500 optimization steps). Rows denote the evaluation model and columns denote the source of the adversarial suffix. BASE suffixes are optimized on the baseline model, while SAE suffixes are optimized on the SAE-augmented model. SAE insertion substantially reduces transfer from baseline attacks (BASE→SAE: 58.8% → 32.4%), while suffixes optimized on the SAE remain moderately transferable to the baseline model (SAE→BASE: 56.6%).

Layer	BASE→SAE ASR (%)
3	31.6
7	**0.0**
11	11.0
15	16.2
19	32.4
23	59.6
27	53.7

Table 15: Effect of SAE insertion depth on adversarial transfer for Qwen2.5-7B-Instruct using a 131K-width SAE. Each row reports attack success rate (ASR) for adversarial suffixes optimized on the baseline model and evaluated on the SAE-augmented model (BASE→SAE). Robustness varies strongly with layer placement: early-to-mid layers produce the strongest reduction in transfer attacks (Layer 7: 0.0% ASR), while later layers exhibit substantially weaker defensive effects.

D Feature-Level Mechanism

D.1 Jaccard Similarity

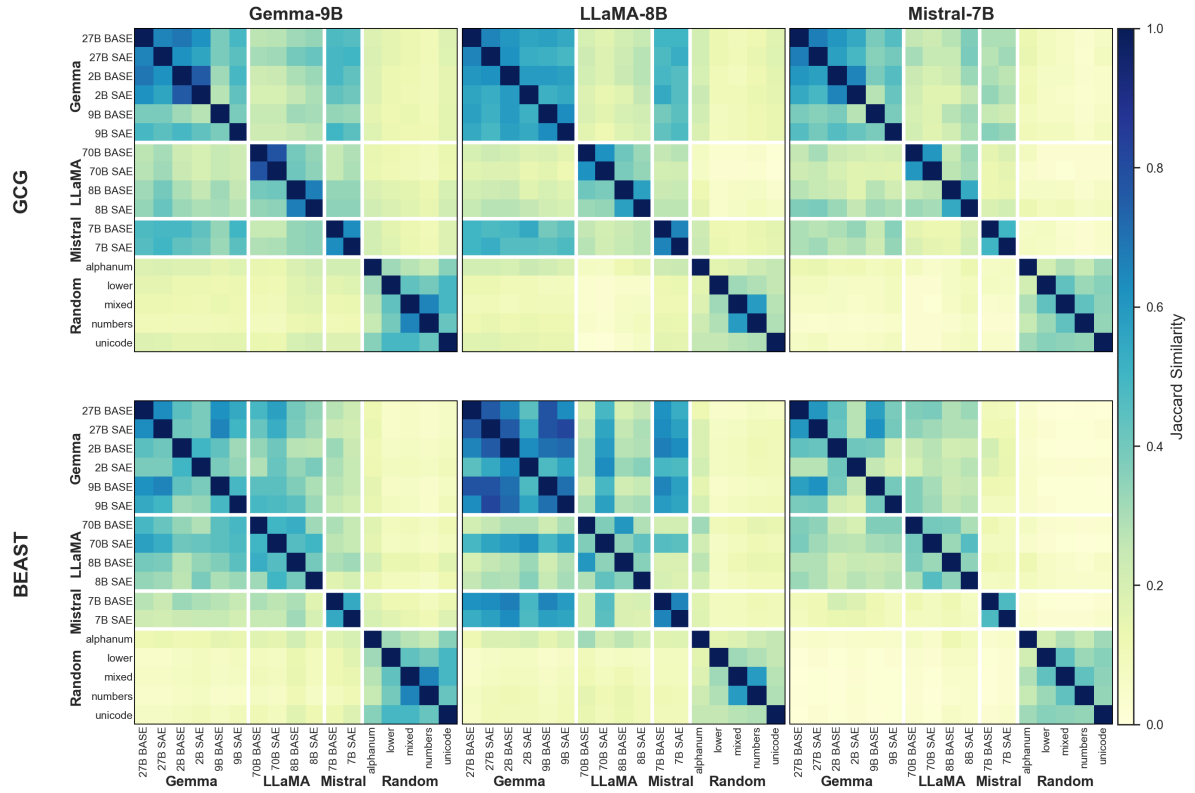

Figure 3: Adversarial suffixes from different source models exhibit high pairwise Jaccard similarity (0.36 ± 0.16), while similarity to random baselines remains low (0.09 ± 0.05). This $4\times$ difference reveals that attacks converge on a shared sparse feature subspace, explaining their cross-model transferability.

Table 16: Jaccard similarity of top-k SAE features across adversarial suffix sources. Within-attack similarity is 3–5$\times$ higher than attack-vs-random similarity, indicating adversarial suffixes converge on a shared sparse feature subspace.

Attack	SAE	Within-Atk	Atk-vs-Rnd	Ratio
GCG	Gemma-9B	$0.38_{\pm 0.13}$	$0.14_{\pm 0.03}$	2.7
	LLaMA-8B	$0.37_{\pm 0.17}$	$0.11_{\pm 0.06}$	3.2
	Mistral-7B	$0.31_{\pm 0.13}$	$0.06_{\pm 0.03}$	5.0
BEAST	Gemma-9B	$0.38_{\pm 0.11}$	$0.09_{\pm 0.04}$	4.0
	LLaMA-8B	$0.46_{\pm 0.19}$	$0.11_{\pm 0.05}$	4.2
	Mistral-7B	$0.26_{\pm 0.14}$	$0.03_{\pm 0.02}$	9.1
Combined		$0.36_{\pm 0.16}$	$0.09_{\pm 0.05}$	3.9

Table 17: Jaccard similarity statistics across attack types and SAE models. Within-attack similarity is consistently 3–5$\times$ higher than attack-vs-random similarity.

Attack	SAE Model	k	Within-Attack	Within-Random	Attack vs Random
GCG	Gemma-9B	135	$0.38_{\pm 0.13}$	$0.38_{\pm 0.12}$	$0.14_{\pm 0.03}$
	LLaMA-8B	79	$0.37_{\pm 0.17}$	$0.28_{\pm 0.11}$	$0.11_{\pm 0.06}$
	Mistral-7B	78	$0.31_{\pm 0.13}$	$0.32_{\pm 0.06}$	$0.06_{\pm 0.03}$
BEAST	Gemma-9B	107	$0.38_{\pm 0.11}$	$0.40_{\pm 0.11}$	$0.09_{\pm 0.04}$
	LLaMA-8B	83	$0.46_{\pm 0.19}$	$0.28_{\pm 0.11}$	$0.11_{\pm 0.05}$
	Mistral-7B	59	$0.26_{\pm 0.14}$	$0.32_{\pm 0.07}$	$0.03_{\pm 0.02}$
GCG (Overall)		–	$0.35_{\pm 0.15}$	$0.33_{\pm 0.11}$	$0.11_{\pm 0.05}$
BEAST (Overall)		–	$0.37_{\pm 0.17}$	$0.33_{\pm 0.11}$	$0.08_{\pm 0.05}$
Combined		–	$0.36_{\pm 0.16}$	$0.33_{\pm 0.11}$	$0.09_{\pm 0.05}$

E Optimization Dynamics

E.1 Spectral Gradient Analysis

Table 18: Spectral analysis of GCG gradients (Wilcoxon signed-rank test). LLaMA-8B: n=218; Gemma-9B: n=210; 10,900 gradient samples per condition. SAE integration significantly increases gradient similarity between steps (+60%) and final loss (+31%), while modestly reducing effective rank (-8.7%) κ: condition number. Var(σ_1): variance explained by first singular value.

	Metric	Base	SAE	Δ	p
LLaMA-8B	$\cos(G_t, G_{t-1})$	0.50±0.15	0.80±0.09	+60%	10^{-36}
	$\mathcal{L}$	0.82±0.45	1.08±0.38	+31%	10^{-20}
	r_{eff}	1.88±0.39	1.72±0.37	-9%	10^{-6}
	σ_1/σ_2	24±16	28±19	+18%	0.005
	κ	543±447	745±534	+37%	10^{-8}
	Var(σ_1)	0.983	0.986	+0%	10^{-4}
Gemma-9B	$\cos(G_t, G_{t-1})$	0.53±0.13	0.76±0.08	+44%	10^{-33}
	$\mathcal{L}$	0.38±0.16	0.61±0.28	+62%	10^{-31}
	r_{eff}	1.85±0.12	1.84±0.10	-1%	0.644
	σ_1/σ_2	8.00±1.00	8.00±1.00	-2%	0.093
	κ	431±86	402±96	-7%	10^{-4}
	Var(σ_1)	0.975	0.975	-0%	0.403

E.2 Optimization Difficulty

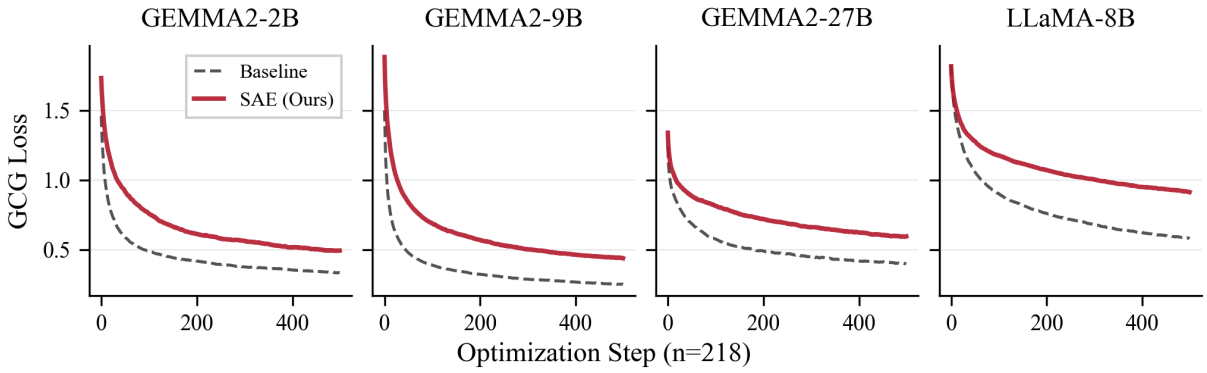

Figure 4: GCG optimization loss curves for baseline versus SAE-intervened models. Higher final loss indicates less effective adversarial suffix generation. SAE integration increases optimization difficulty by 47–74% across model families ($n = 218$ attack runs).

E.3 Spectral Monitoring Results

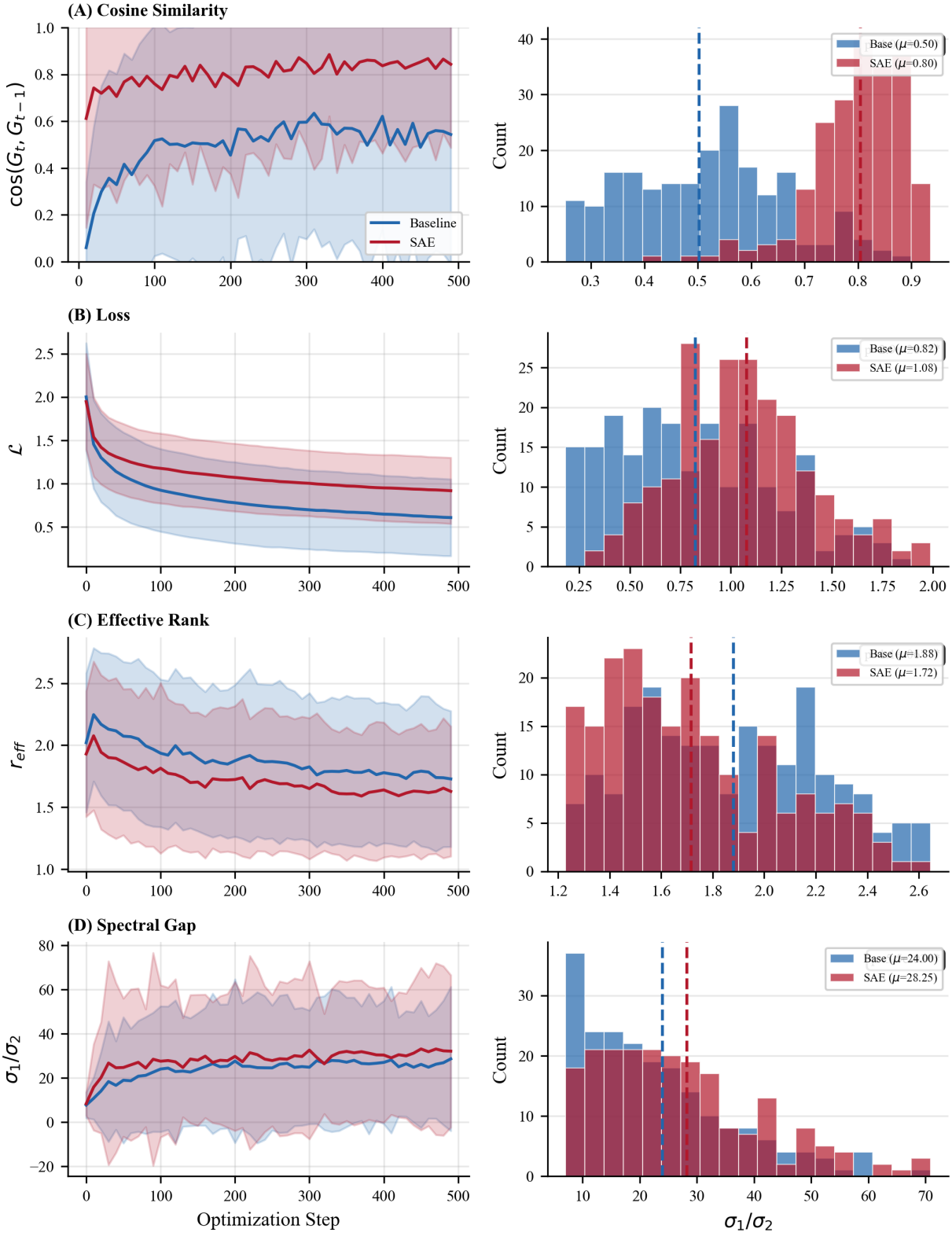

Figure 5: Spectral analysis of GCG gradients on LLaMA-8B (n=218 paired runs). Left panels show metric trajectories over 500 optimization steps (mean ± std); right panels show distributions of trajectory means. **(A)** Cosine similarity: SAE gradients are 60% more correlated between successive steps, indicating optimization stagnation. **(B)** Loss: SAE converges to 31% higher loss. **(C)** Effective rank: SAE gradients occupy 9% fewer dimensions. **(D)** Spectral gap: SAE's dominant singular value is 18% stronger relative to the second. All differences significant (Wilcoxon $p < 0.01$).

F Comparison with other protections

GCG and BEAST exploit the smooth internal optimization landscape of LLMs to construct adversarial suffixes. Defending against such attacks therefore requires not only detecting anomalous prompts (e.g., via perplexity-based filtering (Alon and Kamfonas, 2023)), but disrupting the feasibility of the optimization process itself.

A range of LLM defenses have been proposed (Yi et al., 2024). Some methods use randomized smoothing, including Erase-and-Check (Kumar et al., 2025), SmoothLLM (Robey et al., 2024), Semantic Smoothing (Ji et al., 2024). Several of these methods rely on stochastic input perturbations or auxiliary LLMs for paraphrasing or response judging, increasing computational cost and introducing the need to secure the secondary model itself.

Noise-based defenses such as Smoothed Embeddings (Hase et al., 2025) report attack success rate (ASR) and utility trade-offs as a function of noise scale (e.g., reducing GCG ASR on Vicuna from 94% to 0% for $\sigma \approx 0.01$–0.04). While effective, these approaches primarily evaluate aggregate ASR and utility, without analyzing how the defense reshapes optimization dynamics or cross-model transferability. Similarly, decoding-time interventions (Zhao et al., 2026) and adversarial training methods (Fu et al., 2026) focus on ASR reduction, but do not characterize changes in gradient structure or the geometry of adversarial subspaces.

Our approach introduces a sparse encode–decode bottleneck into the residual stream, forcing all intermediate activations to pass through a structured sparse basis before being propagated further. This intervention preserves full gradient flow but reparameterizes the internal representation space.

As a result, adversarial suffix optimization becomes less expressive and more concentrated in a reduced set of directions. Empirically, this leads to substantially lower attack success rates (up to 5× reduction under GCG), pronounced asymmetry in cross-model transfer (suffixes optimized on baseline models transfer poorly to SAE-augmented models), and measurable changes in optimization dynamics.

In particular, we observe increased gradient alignment between successive steps, reduced effective rank of the gradient matrix, and larger spectral gaps, indicating that adversarial updates collapse into fewer dominant directions and exhibit reduced exploratory capacity. Together, these effects demonstrate that SAE integration does not merely filter outputs, but structurally reshapes the optimization landscape exploited by adversarial attacks.