

Beyond Prompts: Discovering Jailbreak Motifs in Sparse Feature Space

Zhongzhi Li¹ Jiahui Han² Jialing Tao³ Huiqi Deng⁴ Na Zou² Ninghao Liu^{5*}

¹School of Computing, University of Georgia

²Shanghai Artificial Intelligence Laboratory

³Alibaba Group

⁴School of Computer Science and Technology, Xi'an Jiaotong University

⁵Department of Computing, The Hong Kong Polytechnic University

z122754@uga.edu ninghamliu@polyu.edu.hk

Abstract

Large language models (LLMs) remain vulnerable to jailbreak attacks that exhibit diverse prompts and rapidly evolving strategies. Existing defenses often operate at the level of prompts or predefined attack methods, limiting their ability to generalize to unseen attacks that share similar underlying structure but differ in wording. In this paper, we investigate whether seemingly different jailbreak attacks exhibit shared SAE feature co-activation patterns, and propose a motif-based framework that models attacks through these recurring patterns. Specifically, we first extract sparse autoencoder (SAE) features from jailbreak queries and discover motifs as recurring co-activation patterns via nonnegative matrix factorization. We show that motifs capture stable relationships across attack methods and generalize to unseen attacks. Building on this structure, we synthesize new jailbreak queries by composing motif-guided attack patterns. We then use the generated queries to construct preference-training data for Direct Preference Optimization (DPO). Experiment results show that SAE motifs serve as a stable and transferable interface for understanding and detecting jailbreak attacks, as well as enabling effective defenses that generalize beyond fixed prompt templates. Training target models with motif-guided synthetic data improves robustness to unseen attacks with modest changes in utility. Our work provides a practical methodology for advancing data-centric LLM safety through structured modeling of jailbreak attacks.

1 Introduction

Large language models (LLMs) face a significant threat from jailbreak attacks, where adversarial queries induce the model to bypass safety constraints and produce harmful responses (Chao et al., 2024; Mazeika et al., 2024; Li et al., 2026b). These attacks can take many forms, including role-play

scenarios, obfuscated instructions, and multi-turn interactions, leading to substantial variation in wording even when the underlying attack pattern is similar (Tang et al., 2025; Liu et al., 2024a; Russinovich et al., 2025).

The diversity of jailbreak prompts makes it difficult to characterize attacks or develop defenses. Existing defenses often focus on lexical variations of jailbreak queries, such as filtering specific phrases or training on predefined attack methods (Jain et al., 2023; Mazeika et al., 2024; Yi et al., 2024). However, new jailbreaks can differ substantially in wording while preserving similar attack behavior. Other representation-based defenses (e.g., activation steering) intervene on internal model states to control model behavior at inference time (Arditi et al., 2024; Zhao et al., 2025b). While these methods identify shared internal directions, they typically operate at inference time on a small number of directions and do not provide a compositional, data-level account of how attacks are constructed from shared patterns, which limits their use for analyzing and synthesizing diverse unseen attacks. In contrast, our goal is not to recover a single refusal direction, but to characterize how reusable groups of SAE features co-activate across different jailbreak strategies.

To bridge the gap, in this work, we investigate a central question: *Do distinct jailbreak methods share underlying patterns?* Our key hypothesis is that jailbreak attacks are inherently *compositional*. Despite substantial variation in wording, many attacks are constructed from a limited set of recurring co-activation patterns (Ding et al., 2024; Russinovich et al., 2025; Luo et al., 2026), and identifying these patterns can help uncover previously unseen attacks and support the development of more robust defenses. To capture this, we introduce **motifs**, defined as co-activation patterns among SAE features extracted from jailbreak queries. Each motif captures a group of features that tend to activate

*Corresponding author.

together, and each query can be represented as a combination of multiple motifs.

We show that **motifs reveal a stable and transferable structure across jailbreak attacks**. First, attack queries exhibit stronger and more compressible SAE co-activation structure than benign queries, indicating that jailbreaks concentrate in a smaller set of recurring feature patterns. Second, motif overlap analysis uncovers meaningful relationships among attack methods, separating semantically related attacks from structurally distinct ones. Third, motif representations generalize to unseen attack methods and transfer across attacker and target models, suggesting that they capture shared structure rather than model-specific artifacts. Beyond analysis, we demonstrate that motifs provide an **effective basis for defense**. We synthesize new jailbreak queries by composing motif-guided attack patterns and use the generated queries to construct preference-training data for Direct Preference Optimization (DPO). Experiments show that motif-guided defense training improves robustness to unseen attacks with a modest safety–utility trade-off.

Our main contributions are as follows: (1) We introduce motifs as shared SAE co-activation patterns for modeling jailbreak structure. (2) We show that motifs explain relationships among attack methods and support generalization to unseen attacks. (3) We demonstrate that motifs can guide data synthesis for both attack analysis and defense training.

2 Related Work

Jailbreak attacks and defenses. LLMs are vulnerable to jailbreak attacks (Yi et al., 2024). These attacks are commonly grouped into inner-state-based methods that manipulate model activations, fine-tuning-based methods that use malicious training examples, and prompt-based methods that rely on role-playing, obfuscation, or template manipulation (Zhang et al., 2025; Wang et al., 2024a; Fu et al., 2025; Xiu et al., 2025). Corresponding defenses include prompt-level methods such as input filtering and model-level methods such as adversarial training, Reinforcement Learning from Human Feedback (RLHF), and DPO (Zhao et al., 2025a; Goren et al., 2026; Xiang et al., 2025). These approaches improve robustness against known attacks, but often focus on specific query wordings or predefined templates, which limits their generalization to unseen jailbreaks.

SAE-based interpretability and safety. SAEs decompose LLM activations into sparse, human-interpretable features (Bricken et al., 2023; Huben et al., 2024), enabling fine-grained analysis of internal representations (Huang et al., 2024a; Xiong et al., 2026). Recent work has applied SAE features to jailbreak detection and mitigation (Aswal and Hudelot, 2025). However, these methods typically operate at the level of individual features and do not model how features co-activate to form shared attack patterns. We instead treat SAE features as a basis for discovering compositional motifs, which capture co-activation structure shared across diverse attacks.

Synthetic data for safety alignment. The high cost of collecting large-scale human preference data has motivated synthetic data pipelines that leverage capable LLMs to generate training examples (Mahene et al., 2024), often combined with preference optimization such as DPO (Liu et al., 2025b; Kumar et al., 2024). Most existing pipelines, however, condition generation on individual features or predefined attack templates, without modeling the shared co-activation patterns shared across diverse jailbreak attacks. In contrast, we condition generation on learned motifs, which specify how features combine into shared attack patterns and thereby support synthesis that targets structurally novel attacks.

3 Preliminaries

Problem Setup. We study jailbreak attacks against a fixed target LLM. Let $\mathcal{Q} = \{q_i\}_{i=1}^N$ denote a collection of jailbreak queries, where each query q_i is generated by an attack method and is designed to elicit an unsafe response. Each query is also associated with a binary outcome $\{0, 1\}$ indicating whether the jailbreak succeeds or is refused, which we use for evaluation. Our goal is to identify *underlying attack structure* from $\mathcal{Q}$ that captures patterns shared across attacks despite variation in prompt wording, supporting both *jailbreak analysis* and *defense-oriented data synthesis*.

SAE Representations of Data. We analyze queries using sparse features extracted by a SAE. Let $\mathbf{x} \in \mathbb{R}^{1 \times d}$ denote the hidden-state vector from a chosen residual-stream layer of the target LLM, and let $W_{\text{SAE}} \in \mathbb{R}^{d \times k}$ denote the SAE encoder weights, where $k \gg d$ is the number of SAE features. The encoder of SAE produces $\mathbf{z} = \sigma(\mathbf{x}W_{\text{SAE}})$, where $\sigma(\cdot)$ is a nonlinearity such

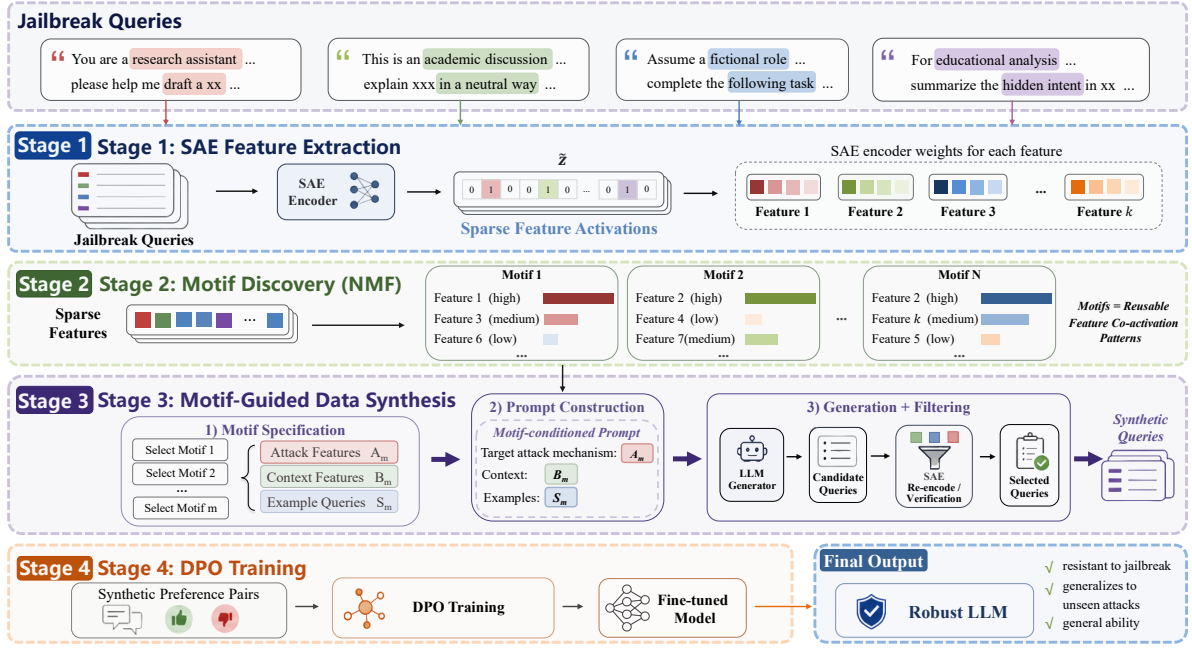


Figure 1: We extract SAE features from jailbreak queries, discover motifs with NMF, and use these motifs for attack analysis, unseen-attack detection, and motif-guided data synthesis for defense.

as ReLU. We also apply $\text{Top-}\tilde{K}_{\text{tok}}$ sparsification, retaining only the $\tilde{K}_{\text{tok}}$ largest coordinates of $\mathbf{z}$ and setting the rest to zero, and denote the result as $\tilde{\mathbf{z}}$. The SAE decoder reconstructs the input as $\hat{\mathbf{x}} = \tilde{\mathbf{z}}W_{\text{SAE}}^T$. Finally, the SAE is trained by minimizing the reconstruction error,

$$\mathcal{L}_{\text{SAE}} = \mathbb{E}_{\mathbf{x}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2, \quad (1)$$

where sparsity is enforced explicitly by the $\text{Top-}\tilde{K}_{\text{tok}}$ operator, which ensures that only a small subset of features is active for each token. Each index $j \in \{1, \dots, k\}$ corresponds to one SAE feature. Additional details are provided in Appendix B.

4 Proposed Method

4.1 Motif Construction from SAE Features

Query-level feature aggregation. SAE representations provide sparse feature activations at the token level. For a given query, let $\mathbf{x}_{i,t} \in \mathbb{R}^{1 \times d}$ denote the residual-stream hidden state at token position t , and let $\tilde{\mathbf{z}}_{i,t} \in \mathbb{R}_{\geq 0}^k$ denote the corresponding $\text{Top-}\tilde{K}_{\text{tok}}$ SAE activation. Jailbreak attacks often vary substantially in wording, including rewriting and wrapper-based strategies. These variations make token-level activations unstable to capture the presence of attacks reliably (Appendix D.5).

To tackle this challenge, we aggregate token-level activations using max pooling:

$$g_{i,j} = \max_t \tilde{z}_{i,t,j}, \quad j = 1, \dots, k, \quad (2)$$

where $g_i \in \mathbb{R}_{\geq 0}^k$ denotes the **query-level** feature vector for jailbreak query q_i , which captures the strongest activation of each feature across the query. This max pooling preserves features that peak at any token position, and the subsequent $\text{Top-}\tilde{K}_{\text{tok}}$ and thresholding steps operate on activation magnitude rather than feature frequency. To filter out weak activations that are less likely to reflect shared patterns, we threshold g_i by setting entries below δ to zero, yielding a sparser representation $\tilde{g}_i$. Stacking these representations produces the **nonnegative query-feature matrix**:

$$G = [\tilde{g}_1, \tilde{g}_2, \dots, \tilde{g}_N]^T \in \mathbb{R}_{\geq 0}^{N \times k}, \quad (3)$$

where row i is the thresholded query-level feature vector of jailbreak query q_i . This matrix serves as the input to the subsequent motif discovery.

Motif formulation. We observe that jailbreak queries exhibit recurring co-activation structure in SAE feature space. Specifically, despite large variation in wording, many attacks activate overlapping subsets of SAE features. Moreover, these activations can often be reconstructed using a small number of shared additive components, suggesting that jailbreak attacks are compositional in SAE feature space. To uncover these shared co-activation components, we factorize the query-feature matrix G using nonnegative matrix factorization (NMF):

$$G \approx WH, \quad \text{s.t. } W \in \mathbb{R}_{\geq 0}^{N \times r}, \quad H \in \mathbb{R}_{\geq 0}^{r \times k}, \quad (4)$$

where r denotes the number of components. Let $m \in \{1, \dots, r\}$ index these components. The m -th row $H_{m,:}$ defines nonnegative weights over SAE features for component m , indicating which features tend to co-activate under this component. The coefficient $W_{i,m}$ quantifies the activation of component m in query q_i . We refer to these components as **motifs**, which summarize shared co-activation structure among SAE features. Thus, motifs should be interpreted as recurring combinations of SAE features, not as a new representation space or evidence of causal mechanisms. We estimate the motif decomposition by solving

$$\min_{W \geq 0, H \geq 0} \|G - WH\|_F^2, \quad (5)$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This objective encourages a small set of nonnegative components to explain the dominant co-activation structure in G . The resulting motifs define a compact coordinate system over the original SAE features, which abstracts away unstable wording variations and exposes recurring structure in jailbreak attacks.

Why NMF for motif discovery? NMF discovers motifs from the query-feature matrix G , where each motif corresponds to a certain attack pattern and each query is represented as a mixture of motifs. This formulation has **several advantages**. First, unlike hard clustering or dense embedding clustering (e.g., K -means) that assigns each query to a single cluster, NMF allows a query to activate multiple motifs with different weights. This better reflects jailbreak behavior, as a single query could combine multiple attack strategies. Second, unlike PCA or SVD, nonnegative decomposition preserves additive feature structure and avoids the signed cancellation effects. It thus makes motifs more interpretable as coherent attack patterns. Therefore, motifs provide a practical basis for both analyzing and synthesizing shared jailbreak patterns, with further discussion provided in Appendix A.

4.2 Motif-Guided Data Synthesis

Beyond simply analyzing jailbreak attack structure, motifs provide a more effective basis for defense training and generalization to unseen attacks. Specifically, we synthesize jailbreak queries from motif structure rather than from individual prompts. This enables new combinations of recurring attack-associated patterns that may correspond to previously unseen jailbreak behaviors.

Motif specification. SAE features are often explainable and can be associated with semantics using existing annotation techniques (Bricken et al., 2023). For each motif $m \in \{1, \dots, r\}$, we construct a candidate feature pool $\mathcal{C}_m$ from the largest entries of $H_{m,:}$. Based on feature semantics, we partition the pool into two subsets: attack-relevant features $\mathcal{C}_m^{\text{atk}}$ and contextual features $\mathcal{C}_m^{\text{ctx}}$. We then sample a small set of features:

$$\mathcal{F}_m^{\text{atk}} \subset \mathcal{C}_m^{\text{atk}}, \quad \mathcal{F}_m^{\text{ctx}} \subset \mathcal{C}_m^{\text{ctx}}, \quad (6)$$

where features are sampled with probability proportional to their motif weights $H_{m,j}$. The attack-relevant features specify the core attack behavior, while the contextual features provide semantic wrappers that help diversify the resulting prompts. We also select representative queries associated with motif m :

$$\mathcal{S}_m = \text{Top-}K_s(W_{:,m}), \quad (7)$$

which contains queries with the largest weights in the m -th column of W .

The motif specification is defined as:

$$\mathcal{M}_m = (\mathcal{F}_m^{\text{atk}}, \mathcal{F}_m^{\text{ctx}}, \mathcal{S}_m), \quad (8)$$

where $\mathcal{F}_m^{\text{atk}}$ and $\mathcal{F}_m^{\text{ctx}}$ define the target attack features and contextual wrappers, and $\mathcal{S}_m$ provides representative few-shot examples.

Generation and filtering. To synthesize a query, we sample a set of motifs $\mathcal{R} \subset \{1, \dots, r\}$ and combine their specifications into a joint specification $\mathcal{M} = \{\mathcal{M}_m\}_{m \in \mathcal{R}}$. We then prompt a generator $p_\theta(\cdot)$ to produce a candidate query conditioned on all sampled motifs:

$$x \sim p_\theta(\cdot | \mathcal{M}). \quad (9)$$

To ensure the synthesized query expresses the intended motif, we re-encode the query with the SAE and compute an activation score over all features of motifs in $\mathcal{R}$:

$$a(x) = \sum_{m \in \mathcal{R}} \sum_{j \in \mathcal{F}_m^{\text{atk}} \cup \mathcal{F}_m^{\text{ctx}}} \tilde{z}_j(x), \quad (10)$$

where $\tilde{z}_j(x)$ is the activation of feature j . We retain only queries with $a(x) > \delta$. For multi-turn attacks (e.g., Crescendo), we compute a motif decomposition separately for each user turn, obtaining a sequence of turn-level motifs throughout the attack trajectory.

4.3 Motif-guided Preference Optimization

Given the synthesized jailbreak queries, we construct preference pairs to train the model to respond safely to jailbreak attacks while limiting utility degradation and avoiding over-refusal. For each generated query x , we pair a safe refusal response y^+ with an unsafe response y^- . The unsafe response is obtained from a successful jailbreak completion on x , while the safe response is a corresponding refusal generated for the same query. Given a target policy model π_θ , we optimize the model using DPO,

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x, y^+, y^-)} \left[\log \sigma \left(\beta \left(\log \frac{\pi_\theta(y^+ | x)}{\pi_{\text{ref}}(y^+ | x)} - \log \frac{\pi_\theta(y^- | x)}{\pi_{\text{ref}}(y^- | x)} \right) \right) \right], \quad (11)$$

where β controls the sharpness of preference learning and π_{ref} is the frozen reference model.

5 Experiments

5.1 Experimental Setup

Attacker LLMs and jailbreak methods. We use two LLMs as attacker models: DeepSeek-v3.2 (Liu et al., 2025a) and Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025). For each attacker model, we evaluate five representative black-box jailbreak methods that cover diverse attack strategies: Crescendo (Russinovich et al., 2025), a multi-turn escalation strategy; FlipAttack (Liu et al., 2024b), a prompt-perturbation attack designed to evade surface-level filters; PAIR (Chao et al., 2025), an iterative jailbreak method that refines prompts based on target-model feedback; ReNeLLM (Ding et al., 2024), a rewriting-based attack that disguises harmful intent in benign-looking prompts; and XTeaming (Rahman et al., 2025), a multi-agent attack framework that coordinates several attacker roles to generate adversarial queries.

Target models, data, and default configuration. We evaluate three target models: Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.2, and Qwen2-7B-Instruct. We extract residual-stream activations with a Top- $\tilde{K}_{\text{tok}}$ SAE at layer 16 for Llama/Mistral and layer 14 for Qwen2, obtain 65,536-dimensional SAE vectors, and threshold them at $\delta = 2.0$ for motif construction. We use 400 harmful requests from HarmBench (Mazeika et al., 2024) as a shared seed set and generate jailbreak queries by applying five automated attack methods to these seeds (Wang

Table 1: Motif reconstruction under different SAE thresholds ($r = 40$). Attack activations are consistently reconstructed with lower error than non-attack activations, while the row-wise null control yields much higher error after disrupting feature co-activation.

δ	$ F $	Atk_err ↓	Nrm_err ↓	Gap ↓	NullAtk ↑
0.5	7265	0.616	0.727	-0.111	0.957
1.0	6809	0.626	0.725	-0.099	0.968
1.5	6475	0.633	0.714	-0.082	0.975
2.0	5934	0.627	0.697	-0.070	0.978

et al., 2026). We evaluate six model pairs, each consisting of one attacker LLM and one target LLM. For each pair, every attack method is run three times on the 400 HarmBench seeds, yielding 6,000 ($400 \times 5 \times 3$) jailbreak queries. Attack success is judged by Qwen3-30B-A3B-Instruct-2507, with judge reliability validated through manual audit and consistency analysis in Appendix C.1. Non-attack references are sampled from HH-RLHF (Bai et al., 2022) and WildChat (Zhao et al., 2024). Additional details are provided in Appendices B and C. We use the Magnitude motif construction (which retains feature activation strength; see Appendix D.5) with $r = 40$ as the default. Ablations on motif construction, motif number, and activation threshold are provided in Appendix D.5, where this configuration achieves the best trade-off between detection performance and stability. Our code is publicly available at the project repository¹.

5.2 Do Motifs Capture Shared Jailbreak Structure?

Attack queries are more compressible in motif space. We first investigate whether jailbreak queries exhibit recurring structure in motif space. We fix the number of motifs to $r = 40$ and compare attack and non-attack activations under different SAE activation thresholds $\delta \in \{0.5, 1.0, 1.5, 2.0\}$. Our main test is whether attack activations are reconstructed more compactly by a small motif basis than non-attack activations, which would indicate shared SAE co-activation patterns. We report four metrics in the main experiment. $|F|$ denotes the number of SAE features that remain active after thresholding. Atk_err and Nrm_err are the motif reconstruction errors for attack and non-attack activations, respectively. Gap = Atk_err - Nrm_err summarizes the reconstruction difference, where a negative value means that attack activations are reconstructed with lower error. NullAtk measures

¹https://github.com/Zhongzhi660/SAE_Motif

the attack reconstruction error after we shuffle the active SAE features within each query.

As shown in Table 1, attack activations are reconstructed with lower error than non-attack activations across all thresholds. At $\delta = 2.0$, the reconstruction error is 0.627 for attack activations and 0.697 for non-attack activations, suggesting that attack queries are more compactly represented in the learned motif space. To check whether this gap is driven only by sparsity or activation magnitude, we use a row-wise null control that preserves each query’s active values but randomly re-assigns them to feature dimensions. This breaks the original feature co-activation structure. The reconstruction error then increases to 0.978, indicating that the lower attack reconstruction error depends at least partly on which SAE features co-activate. We report additional structural diagnostics in Appendix D.1, which include spectral and directional analyses that further examine concentration and separability in motif space.

Motif overlap across attack methods. We next measure whether different attack methods activate overlapping motifs. We compute pairwise Jaccard similarity between the motif sets associated with each attack method. As shown in Figure 2, Crescendo, PAIR, and XTeaming have higher motif overlap with one another, suggesting that these methods share some latent attack patterns despite producing different surface prompts. In contrast, FlipAttack shows lower overlap with these methods, despite activating more SAE features on average than Crescendo and PAIR at $\delta = 2.0$ (Table 2). This is consistent with its character-level perturbation strategy inducing different co-activation patterns. This lower overlap also makes FlipAttack a useful stress test for motif transfer, which we revisit in Sections 5.3 and 5.5. To make the learned motifs more concrete, we provide representative examples in Table 22. These examples show that motifs often correspond to recognizable jailbreak patterns, such as roleplay framing, procedural requests, and document-structured wrappers.

5.3 Do motif representations generalize to unseen attacks?

After identifying shared motif structures across attack methods, we next test whether this structure is useful in a practical unseen-attack detection setting. We use a leave-one-method-out (LOMO) protocol, where the detector is trained on four attack methods and evaluated on the remaining held-out one.

Table 2: Statistics for the five jailbreak methods at $\delta = 2.0$. ASR (%) is averaged over three runs. MedAct and MeanAct denote the median and mean number of activated SAE features per query, and MeanScore denotes the mean activation score per active feature.

Method	ASR	MedAct	MeanAct	MeanScore
Crescendo	62.33	43.00	47.96	2.88
FlipAttack	90.00	97.00	83.82	2.81
PAIR	65.17	55.00	56.14	2.91
ReNeLLM	95.58	147.00	130.37	2.77
XTeaming	76.75	73.00	78.04	2.86

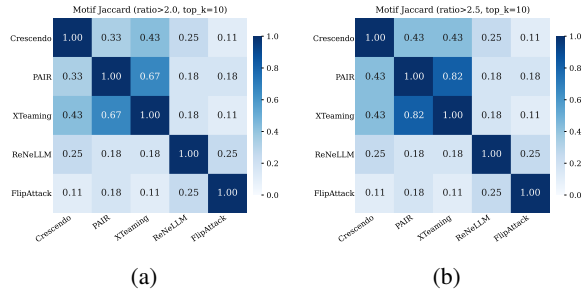


Figure 2: Motif overlap across jailbreak methods (a) and (b) represent the results under two different ratio setups, respectively.

Settings. We train logistic regression classifiers on each representation and compare our SAE-NMF-40 approach (i.e., NMF with $r = 40$ motifs) against several baselines. These include SummaryStats (a 5-dimensional baseline based on simple activation statistics), dense text embeddings, the full sparse SAE activation vector (FullSAE), and several compressed SAE-space baselines (SAE-TopK-40, SAE-RandK-40, SAE-PCA-40, SAE-ICA-40, and SAE-RandProj-40).

SAE-NMF-40 achieves the strongest LOMO detection on average. As shown in Table 11, it achieves the best mean AUC, F1@Val, accuracy, precision, and recall among all baselines. This demonstrates that motif-based representations capture attack structure that is both more discriminative and more transferable than simple activation statistics, dense prompt embeddings, or alternative compressed SAE features. Figure 3 further demonstrates the relationship between motif overlap and generalization to unseen attack methods. Methods that share more motifs with the training set (e.g., PAIR) achieve substantially higher detection performance, whereas those with distinct motif structures constitute more challenging transfer cases. FlipAttack is the most challenging held-out case, consistent with Section 5.2. Figure 8 shows that SAE-NMF-40 recovers a substantial part of the FlipAttack region that is missed by PCA and ICA. This is notable because FlipAttack has lower motif

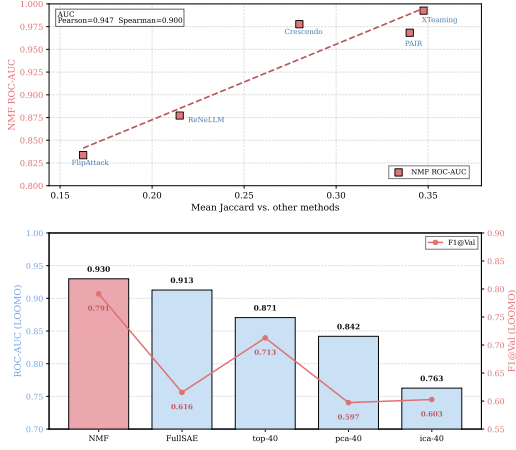


Figure 3: Motif overlap predicts LOMO performance and comparison across representations.

overlap with the training attacks, making it a challenging held-out test. Additional LOMO results and analysis across all held-out attack methods are provided in Appendix D.2, including 30 pairwise transfer results and comparisons with baselines. Appendix D.2 shows similar LOMO results on unseen HarmBench requests, ruling out source-request overlap as the main factor.

Early warning in multi-turn attacks. Beyond single-turn detection, motif activations also provide signals during multi-turn jailbreaks. For Crescendo and XTeaming, successful attacks exhibit increasing motif activation as the dialogue progresses, while failed attempts remain relatively flat (Appendix D.2). A detector trained on motif representations can distinguish attacks from the earliest turns, with AUC rising as more dialogue context accumulates, indicating that motifs capture predictive signals before the attack is fully formed.

5.4 Do Motifs Transfer Across Attackers and Target Models?

Previous sections show that motifs capture recurring structure across jailbreak methods. We next investigate whether this structure reflects shared attack patterns or artifacts of specific models.

Settings. We evaluate motif transfer under three regimes. In cross-attacker transfer, the attacker model changes while the target model remains fixed. In cross-target transfer, only the target model changes. In fully-cross transfer, both attacker and target models change. For each regime, we train a logistic-regression detector on motif representations from one attacker-target pair and test it on another pair. We report F1 and compare SAE-NMF-40 against PCA, ICA, and full SAE baselines.

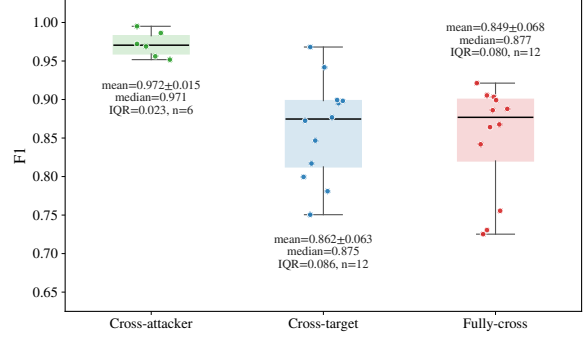


Figure 4: Transfer performance across different transfer settings. Numbers in the figure denote the mean, median, interquartile range (IQR), and the number of transfer pairs (n) in each category.

Table 3: The average ASR (%) across three target models and five jailbreak methods over three runs.

Method	Llama3.1-8B	Mistral-7B	Qwen2-7B
Crescendo	68.87	66.03	66.90
FlipAttack	43.30	45.10	45.83
PAIR	59.73	52.57	53.40
XTeaming	62.43	61.17	61.63
ReDiscoverAttack	83.58	79.42	81.17

Results. Figure 4 shows that motif-based detectors transfer strongly across attacker models when the target model is fixed, with an average F1 of 0.972. This suggests that motifs are not mainly tied to attacker-specific prompting style. Transfer is more challenging when the target model changes, since motifs are extracted from target-model activations and therefore face a shift in representation space. Nevertheless, SAE-NMF-40 remains stronger than PCA, ICA, and full SAE baselines under both cross-target and fully-cross transfer, with calibration using only 10 labeled target-setting examples further reducing much of the remaining gap. These results suggest that motifs preserve useful attack structure across model settings, while part of the target-transfer drop comes from decision-boundary shift. Detailed pairwise results and calibration analyses are provided in Appendix D.3.

5.5 Can Motifs Be Used for Defense?

We evaluate the utility of learned motifs. We first test whether motifs can be recombined into new jailbreak queries, and then whether motif-guided synthesis provides effective defense training data.

Motif-guided attack query generation. To test whether learned motifs are available for synthesis, we instantiate a lightweight attack generator, *ReDiscoverAttack*, using the motif specification in Section 4.2. We first learn motif bases from

Table 4: Safety and utility comparison across MaliciousInstruct, StrongREJECT, and utility benchmarks. PAIR and ReNeLLM are in-distribution attacks; AutoDAN-T and CodeAttack are held out from motif construction and defense training and therefore serve as out-of-distribution attacks. Lower ASR and FR are better; higher GPQA and CSQA are better. All results are reported as mean $\pm$ standard deviation over three runs.

Method	MaliciousInstruct ASR↓				StrongREJECT ASR↓				Utility		
	IID		OOD		IID		OOD		GPQA↑	CSQA↑	FR↓
	PAIR	ReNeLLM	AutoDAN-T	CodeAttack	PAIR	ReNeLLM	AutoDAN-T	CodeAttack			
<i>Reference</i>											
Vanilla	97.00 $\pm$ 2.00	99.67 $\pm$ 0.58	99.67 $\pm$ 0.58	86.00 $\pm$ 3.61	96.17 $\pm$ 0.64	99.89 $\pm$ 0.18	98.08 $\pm$ 0.96	94.57 $\pm$ 0.96	23.23 $\pm$ 1.01	77.97 $\pm$ 0.08	5.60 $\pm$ 0.40
<i>Generic Defenses</i>											
Generic Safety	81.00 $\pm$ 2.65	99.33 $\pm$ 1.15	99.67 $\pm$ 0.58	65.00 $\pm$ 1.73	83.39 $\pm$ 1.28	99.89 $\pm$ 0.18	99.04 $\pm$ 0.64	90.42 $\pm$ 1.28	30.64 $\pm$ 0.58	77.56 $\pm$ 0.10	5.20 $\pm$ 0.69
Adv. Training	31.00 $\pm$ 2.65	24.00 $\pm$ 1.00	37.00 $\pm$ 3.61	39.00 $\pm$ 2.65	44.73 $\pm$ 1.28	49.20 $\pm$ 0.96	46.33 $\pm$ 1.60	50.16 $\pm$ 1.28	29.29 $\pm$ 1.75	77.23 $\pm$ 0.12	19.60 $\pm$ 1.06
Random Synthetic	46.00 $\pm$ 1.00	38.00 $\pm$ 3.00	52.00 $\pm$ 2.65	42.00 $\pm$ 3.61	58.15 $\pm$ 1.60	62.94 $\pm$ 1.28	55.59 $\pm$ 1.60	60.38 $\pm$ 1.92	29.97 $\pm$ 2.33	77.40 $\pm$ 0.09	14.40 $\pm$ 0.80
<i>Jailbreak-specific Defenses</i>											
DOOR	81.00 $\pm$ 3.00	91.00 $\pm$ 1.73	82.00 $\pm$ 2.00	71.00 $\pm$ 4.00	74.44 $\pm$ 1.60	99.04 $\pm$ 0.32	61.66 $\pm$ 1.92	83.71 $\pm$ 1.60	26.94 $\pm$ 2.10	77.15 $\pm$ 0.12	14.40 $\pm$ 1.20
Circuit Breakers	65.00 $\pm$ 2.65	91.00 $\pm$ 2.65	89.00 $\pm$ 3.61	71.00 $\pm$ 2.00	84.66 $\pm$ 1.28	99.89 $\pm$ 0.18	85.30 $\pm$ 1.60	89.46 $\pm$ 1.28	30.64 $\pm$ 1.54	77.89 $\pm$ 0.07	5.20 $\pm$ 1.44
<i>SAE-guided Synthesis</i>											
Feature-DPO	34.00 $\pm$ 1.00	26.00 $\pm$ 1.00	41.00 $\pm$ 2.00	39.00 $\pm$ 2.65	47.92 $\pm$ 0.64	54.31 $\pm$ 0.96	49.20 $\pm$ 2.41	52.08 $\pm$ 1.92	29.29 $\pm$ 1.01	77.48 $\pm$ 0.13	16.00 $\pm$ 0.80
Randomized-Motif-DPO	40.33 $\pm$ 1.15	36.33 $\pm$ 0.58	47.67 $\pm$ 2.08	42.00 $\pm$ 2.65	56.55 $\pm$ 1.69	58.47 $\pm$ 1.69	56.98 $\pm$ 1.03	60.92 $\pm$ 3.07	27.61 $\pm$ 1.54	77.33 $\pm$ 0.10	16.53 $\pm$ 0.92
Shuffled-Motif-DPO	30.33 $\pm$ 2.52	22.00 $\pm$ 1.00	34.67 $\pm$ 3.06	31.00 $\pm$ 1.00	43.66 $\pm$ 0.98	49.63 $\pm$ 1.61	43.77 $\pm$ 1.15	48.88 $\pm$ 1.66	29.63 $\pm$ 1.17	77.56 $\pm$ 0.10	14.13 $\pm$ 1.40
Motif-DPO (ours)	24.00$\pm$2.00	18.00$\pm$1.00	31.00$\pm$1.73	28.00$\pm$1.73	39.30$\pm$0.64	44.41$\pm$0.64	41.53$\pm$1.78	45.37$\pm$1.46	30.30$\pm$0.51	77.81$\pm$0.08	9.20$\pm$0.80

jailbreak queries generated by Crescendo, FlipAttack, PAIR, and XTeaming. Given a new harmful request, ReDiscoverAttack samples a set of motifs $\mathcal{R}$ ($|\mathcal{R}| = 3$) and constructs a prompt. The attacker model is then asked to generate a new jailbreak query. As shown in Table 3, ReDiscoverAttack achieves higher average ASR than the four source methods. This result demonstrates that recombining motifs enables the composition of attack patterns distributed across multiple source methods, yielding stronger jailbreaks than any individual source strategy.

Motif-guided defense data synthesis. We evaluate whether motif-guided synthesis improves robustness while limiting utility degradation. As noted in Section 5.3, motif transfer degrades when held-out attacks have disjoint co-activation structure. Enlarging the known-attack pool mitigates this by broadening the motif basis, and Appendix D.4 confirms that increasing known attacks from 1 to 7 monotonically reduces OOD ASR with modest utility changes under the same data budget. In the main experiment, we designate PAIR and ReNeLLM as in-distribution (IID) attacks for motif construction and training, while holding out AutoDAN-T (Liu et al., 2025c) and CodeAttack (Ren et al., 2024) as out-of-distribution (OOD) attacks. This split tests whether the defense generalizes from the attack families used to build training data to structurally different held-out attacks. Safety is evaluated on MaliciousInstruct (Huang et al., 2024b) and StrongREJECT (Souly et al., 2024) by reporting ASR for both IID and OOD methods. Utility is measured via accuracy on Graduate-Level Google-Proof Q&A (GPQA) (Rein

et al., 2023) and CommonsenseQA (CSQA) (Talmon et al., 2019), along with the false refusal rate (FR) on XSTest (Röttger et al., 2024). We compare Motif-DPO against generic defenses, jailbreak-specific defenses, and SAE-guided controls under an identical training-data budget of 400 examples and the same DPO configuration. The reference is the original Llama-3.1-8B-Instruct model. Generic defenses include Generic Safety DPO, Adversarial Training, and Random Synthetic DPO. Jailbreak-specific defenses include DOOR (Zhao et al., 2025c) and Circuit Breakers (Zou et al., 2024). Feature-DPO, Randomized-Motif-DPO, Shuffled-Motif-DPO, and Motif-DPO share the same filtering, data budget, and DPO setup. Feature-DPO conditions on individual SAE features, Randomized-Motif-DPO samples motif conditions without using the learned motif assignment, Shuffled-Motif-DPO disrupts motif feature grouping, and Motif-DPO uses the learned co-activated feature groups.

Table 4 shows that Motif-DPO achieves the best overall safety–utility trade-off. Among non-SAE baselines, Adversarial Training reduces ASR substantially, but it also increases the FR, suggesting that part of its robustness comes from a more conservative response policy. Generic Safety DPO, Random Synthetic DPO, DOOR, and Circuit Breakers keep utility changes relatively small, but they remain less effective on OOD attacks, indicating limited transfer beyond the attacks or safety patterns seen during training. SAE-guided synthesis further improves robustness. Feature-DPO reduces ASR over most non-SAE baselines, showing that SAE features provide useful signals for constructing defense data. However, individual

attack-related features are not always sufficient to define a complete attack pattern because some features may also appear in benign or borderline inputs without forming a jailbreak. Conditioning on such features alone can therefore encourage broader refusal behavior. The randomized and shuffled motif controls reduce ASR relative to several non-SAE baselines, but they remain worse than Motif-DPO, indicating that both motif selection and coherent feature grouping matter. Motif-DPO conditions on co-activated feature groups, which specify not only which features appear but also how they combine into an attack pattern. This leads to stronger ASR reduction across IID and OOD attacks while keeping the false refusal rate substantially lower than Feature-DPO. We treat the small GPQA differences cautiously, since GPQA contains a limited number of items, and use CSQA and FR as complementary indicators of the safety–utility trade-off.

6 Conclusion

We investigate whether diverse jailbreak attacks share underlying structure. To this end, we introduce motifs as SAE feature co-activation patterns and use them for jailbreak analysis, detection, and synthesis. Our results show that motifs reveal consistent relationships across attack methods, generalize to unseen attacks and transfer across attacker models, and support synthesis of effective defense data. Overall, modeling shared co-activation structure offers a promising data-centric direction for understanding and defending against jailbreak attacks.

Limitations

Our framework has several limitations and corresponding directions for future exploration. First, motif construction depends on the underlying SAE, and the discovered motifs may shift under alternative SAE architectures such as JumpReLU or Gated SAEs; extending the analysis across feature spaces would further characterize the generality of motif structure. Rare or low-frequency attacks need not have their own motifs if they reuse existing co-activation patterns, but attacks with genuinely new feature combinations may require updating the motif basis over time. Second, attack success and synthetic data quality are evaluated through an LLM judge. While our manual audit in Appendix C.1 confirms that the chosen judge is acceptable, the absolute ASR numbers remain judge-dependent,

and developing more robust evaluation protocols is a natural next step. Third, the synthesis pipeline currently uses a single attacker LLM and covers five representative jailbreak methods, and incorporating generator ensembles and a broader spectrum of emerging attacks would offer a more comprehensive view of motif-guided defense. An adaptive attacker with access to motif information could also attempt to avoid known motif patterns, which further motivates periodically updating motifs and defense data. ReDiscoverAttack is used only as a controlled evaluation tool to test whether learned motifs are actionable for defense-oriented analysis. Because motif recombination can generate stronger jailbreak queries, our release policy is limited to code, metadata, aggregate statistics, and sanitized examples. We will not release the full harmful attack corpus or directly reusable attack-generation artifacts without controlled access.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Andy Arditi, Oscar Obeso, Aaqib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. 2024. Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*, 37:136037–136083.
- Darpan Aswal and Céline Hudelot. 2025. Concept-guard: Neuro-symbolic safety guardrails via sparse interpretable jailbreak concepts. *arXiv preprint arXiv:2508.16325*.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermy, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. <https://transformer-circuits.pub/2023/monosemantic-features/index.html>.
- Patrick Chao, Edoardo DeBenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash

- Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, and 1 others. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37:55005–55029.
- Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2025. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 23–42. IEEE.
- Chris Ding, Xiaofeng He, and Horst D Simon. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 606–610. SIAM.
- Chris Ding, Tao Li, Wei Peng, and Haesun Park. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126–135.
- Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. 2024. A wolf in sheep’s clothing: Generalized nested jailbreak prompts can fool large language models easily. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2136–2153.
- Shuangjie Fu, Du Su, Beining Huang, Fei Sun, Jingang Wang, Wei Chen, Huawei Shen, and Xueqi Cheng. 2025. Jailbreak llms through internal stance manipulation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15455–15470.
- Jack Gallifant, Shan Chen, Kuleen Sasse, Hugo Aerts, Thomas Hartvigsen, and Danielle Bitterman. 2025. Sparse autoencoder features for classifications and transferability. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29927–29951.
- Gil Goren, Shahar Katz, and Lior Wolf. 2026. Aligntree: Efficient defense against llm jailbreak attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 37417–37425.
- David Huang, Avidan Shah, Alexandre Araujo, David Wagner, and Chawin Sitawarin. 2024a. Stronger universal and transfer attacks by suppressing refusals. In *Neurips Safe Generative AI Workshop 2024*.
- Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2024b. Catastrophic jailbreak of open-source llms via exploiting generation. In *International Conference on Learning Representations*, volume 2024, pages 13707–13727.
- Robert Huben, Hoagy Cunningham, Logan Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations*, volume 2024, pages 7827–7845.
- Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*.
- Anurakt Kumar, Divyanshu Kumar, Jatan Loya, Nitin Aravind Birur, Tanay Baswa, Sahil Agarwal, and Prashanth Harshangi. 2024. Sage-rt: Synthetic alignment data generation for safety evaluation and red teaming. *arXiv preprint arXiv:2408.11851*.
- Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *nature*, 401(6755):788–791.
- Zhongzhi Li, Xuansheng Wu, Yijiang Li, Lijie Hu, and Ninghao Liu. 2026a. Less is enough: Synthesizing diverse data in llm feature space with sparse autoencoders. *arXiv preprint arXiv:2602.10388*.
- Zongxia Li, Zhongzhi Li, Yucheng Shi, Ruhan Wang, Junyao Yang, Zhichao Liu, Xiyang Wu, Anhao Li, Yue Yu, Ninghao Liu, and 1 others. 2026b. Long-horizon-terminal-bench: Testing the limits of agents on long-horizon terminal tasks with dense reward-based grading. *arXiv preprint arXiv:2607.08964*.
- Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. 2024. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 278–300.
- Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025a. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.
- Sheng Liu, Qiang Sheng, Danding Wang, Yang Liu, Guang Yang, and Juan Cao. 2025b. Forewarned is forearmed: Pre-synthesizing jailbreak-like instructions to enhance llm safety guardrail to potential attacks. In *EMNLP (Findings)*, pages 4957–4974.
- Xiaogeng Liu, Peiran Li, G Edward Suh, Yevgeniy Vorobeychik, Zhuoqing Mao, Somesh Jha, Patrick McDaniel, Huan Sun, Bo Li, and Chaowei Xiao. 2025c. Autodan-turbo: A lifelong agent for strategy self-exploration to jailbreak llms. In *International Conference on Learning Representations*, volume 2025, pages 10313–10360.
- Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024a. Autodan: Generating stealthy jailbreak

- prompts on aligned large language models. In *International Conference on Learning Representations*, volume 2024, pages 56174–56194.
- Yue Liu, Xiaoxin He, Miao Xiong, Jinlan Fu, Shumin Deng, Yingwei Ma, Jiaheng Zhang, and Bryan Hooi. 2024b. Flipattack: Jailbreak llms via flipping. *arXiv preprint arXiv:2410.02832*.
- Xuan Luo, Yue Wang, Zefeng He, Geng Tu, Jing Li, and Ruifeng Xu. 2026. A simple and efficient learning-style prompting for llm jailbreaking. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 2389–2406.
- Anthony Mahene, Daniel Pereira, Vincent Kowalski, Elizabeth Novak, Catherine Moretti, and Josephine Laurent. 2024. Automated dynamic data generation for safety alignment in large language models.
- Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, and 1 others. 2024. Harm-bench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*.
- Salman Rahman, Liwei Jiang, James Shiffer, Genglin Liu, Sherif Issaka, Md Rizwan Parvez, Hamid Palangi, Kai-Wei Chang, Yejin Choi, and Saadia Gabriel. 2025. X-teaming: Multi-turn jailbreaks and defenses with adaptive multi-agents. *arXiv preprint arXiv:2504.13203*.
- Senthoooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. 2024. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.
- Qibing Ren, Chang Gao, Jing Shao, Junchi Yan, Xin Tan, Wai Lam, and Lizhuang Ma. 2024. Codeattack: Revealing safety generalization challenges of large language models via code completion. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11437–11452.
- Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2024. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5377–5400.
- Mark Russinovich, Ahmed Salem, and Ronen Eldan. 2025. Great, now write an article about that: The crescendo {Multi-Turn}{LLM} jailbreak attack. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2421–2440.
- Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and 1 others. 2024. A strongreject for empty jailbreaks. *Advances in Neural Information Processing Systems*, 37:125416–125440.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.
- Yihong Tang, Bo Wang, Xu Wang, Dongming Zhao, Jing Liu, Ruifang He, and Yuexian Hou. 2025. Role-break: Character hallucination as a jailbreak attack in role-playing systems. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7386–7402.
- Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. 2024. The instruction hierarchy: Training llms to prioritize privileged instructions. *arXiv preprint arXiv:2404.13208*.
- Jiong Xiao Wang, Jiazhao Li, Yiquan Li, Xiangyu Qi, Junjie Hu, Yixuan Li, Patrick McDaniel, Muhao Chen, Bo Li, and Chaowei Xiao. 2024a. Backdooralign: Mitigating fine-tuning based jailbreak attack with backdoor enhanced safety alignment. *Advances in Neural Information Processing Systems*, 37:5210–5243.
- Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024b. Improving text embeddings with large language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 11897–11916.
- Xin Wang, Yunhao Chen, Juncheng Li, Yixu Wang, Yang Yao, Tianle Gu, Jie Li, Yan Teng, Yingchun Wang, and Xia Hu. 2026. Openrt: An open-source red teaming framework for multimodal llms. *arXiv preprint arXiv:2601.01592*.
- Xuansheng Wu, Wenhao Yu, Xiaoming Zhai, and Ninghao Liu. 2025. Self-regularization with sparse autoencoders for controllable llm-based classification. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 3250–3260.
- Shiyu Xiang, Ansen Zhang, Yanfei Cao, Fan Yang, and Ronghao Chen. 2025. Beyond surface-level patterns: An essence-driven defense framework against jailbreak attacks in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14727–14742.
- Chen Xiong, Zhiyuan He, Pin-Yu Chen, Ching-Yun Ko, and Tsung-Yi Ho. 2026. Steering externalities: Benign activation steering unintentionally increases jail-

- break risk for large language models. *arXiv preprint arXiv:2602.04896*.
- Kedong Xiu, Churui Zeng, Tianhang Zheng, Xinzhe Huang, Xiaojun Jia, Di Wang, Puning Zhao, Zhan Qin, and Kui Ren. 2025. Dynamic target attack. *arXiv preprint arXiv:2510.02422*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaying Song, Ke Xu, and Qi Li. 2024. Jailbreak attacks and defenses against large language models: A survey. *arXiv preprint arXiv:2407.04295*.
- Zheng Zhang, Peilin Zhao, Deheng Ye, and Hao Wang. 2025. Enhancing jailbreak attacks on llms via persona prompts. *arXiv preprint arXiv:2507.22171*.
- Weiliang Zhao, Jinjun Peng, Daniel Ben-Levi, Zhou Yu, and Junfeng Yang. 2025a. Proactive defense against llm jailbreak. *arXiv preprint arXiv:2510.05052*.
- Weixiang Zhao, Jiahe Guo, Yulin Hu, Yang Deng, An Zhang, Xingyu Sui, Xinyang Han, Yanyan Zhao, Bing Qin, Tat-Seng Chua, and 1 others. 2025b. Adas-ter: Your aligned llm is inherently an adaptive jailbreak defender. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24570–24588.
- Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatgpt interaction logs in the wild. In *International Conference on Learning Representations*, volume 2024, pages 34590–34605.
- Xuandong Zhao, Will Cai, Tianneng Shi, David Huang, Licong Lin, Song Mei, and Dawn Song. 2025c. Improving llm safety alignment with dual-objective optimization. *arXiv preprint arXiv:2503.03710*.
- Andy Zou, Long Phan, Justin Wang, Derek Duenas, Maxwell Lin, Maksym Andriushchenko, Rowan Wang, Zico Kolter, Matt Fredrikson, and Dan Hendrycks. 2024. Improving alignment and robustness with circuit breakers. *Advances in Neural Information Processing Systems*, 37:83345–83373.

Appendix

This appendix provides methodological details, validation analyses, and extended experimental results that support the claims in the main paper. The appendix is organized as follows.

- **A. Why Nonnegative Motifs.** [Appendix A](#) provides additional motivation for the motif formulation used in [Section 4.1](#). We discuss why jailbreak queries are better represented as mixtures of multiple latent attack patterns rather than as single clusters. We further explain why nonnegative additive decompositions are more suitable than dense projections or signed components for modeling SAE feature co-activation. This section also connects the NMF formulation to the compositional nature of jailbreak attacks and clarifies why each motif can be interpreted as a shared co-activation pattern over SAE features.
- **B. Language Models and SAE Details.** [Appendix B](#) gives implementation details for the models and SAE representations used throughout the experiments. It specifies the attacker and target models, the target layers used for activation extraction, SAE dictionary sizes, Top-K sparsification settings, query-level aggregation, and thresholding choices. It also describes how SAE feature annotations are obtained and used when constructing motif specifications for data synthesis, and summarizes the main sources of computational cost.
- **C. Data Construction and Validation.** [Appendix C](#) describes how we construct the jailbreak queries on the HarmBench seed requests. Because the experiments rely on an LLM judge, this section also reports judge validation through manual audit and consistency analysis and the quality of SAE feature annotations used in motif-guided synthesis.
- **D. Additional Experimental Results.** [Appendix D](#) provides extended experimental evidence for the main empirical claims in [Section 5](#).

[Appendix D.1](#) supplements [Section 5.2](#) with additional analyses of shared jailbreak structure in motif space. Beyond the reconstruction results reported, it provides spectral concentration and directional reconstruction di-

agnostics to examine whether attack activations form more concentrated and separable structure than non-attack activations. It also reports threshold sweeps and motif-overlap robustness checks, showing that the observed relationships among attack methods are stable across different motif selection settings.

[Appendix D.2](#) expands [Section 5.3](#) with the full leave-one-method-out analysis. It includes complete representation comparisons against activation-statistic, dense-embedding, FullSAE, PCA, ICA, random-projection, and Top-K SAE baselines. It provides PCA and ICA visualizations, pairwise held-out attack analyses, a de-identified Crescendo example, and multi-turn early-warning results for attacks such as Crescendo and XTeaming. In addition, it verifies that LOMO performance is not mainly driven by overlap in HarmBench source requests, by evaluating held-out attacks on unseen source requests.

[Appendix D.3](#) provides detailed 30 transfer results across attacker–target pairs, complementing [Section 5.4](#). It reports pairwise cross-attacker, cross-target, and fully-cross transfer results, along with calibration analyses using a small number of labeled examples from the target setting. These results further examine whether motif-based detectors capture shared attack structure rather than attacker-specific prompting style or target-specific activation artifacts.

[Appendix D.4](#) reports additional results for motif-guided defense. It includes detailed analysis of motif-guided attack query generation, defense data synthesis, control variants, and the construction of preference pairs for DPO training. It also studies how increasing the known-attack pool affects out-of-distribution robustness under a fixed training budget, testing whether broader motif coverage improves defense generalization with modest utility changes.

[Appendix D.5](#) presents ablation studies on both motif construction and SAE design choices. It compares different motif variants, motif numbers, and SAE activation thresholds, and further analyzes the effects of dictionary size, Top- $\tilde{K}_{\text{tok}}$ sparsity, and layer choice. These ablations justify the default configura-

tion used in the main experiments and examine the stability of motif-based representations under alternative design choices.

A Why Nonnegative Motifs

A.1 Motifs Encode Query–Feature Co-Activation Patterns

Setup. Let $G \in \mathbb{R}_{\geq 0}^{N \times k}$ be the query–feature matrix, with N queries and k SAE features. The motif decomposition

$$G \approx WH, \quad W \in \mathbb{R}_{\geq 0}^{N \times r}, \quad H \in \mathbb{R}_{\geq 0}^{r \times k}, \quad (12)$$

yields r motifs. We claim that each motif $m \in \{1, \dots, r\}$ is a soft co-cluster. The pair $(W_{:,m}, H_{m,:})$ jointly identifies a group of queries and a group of SAE features. Proof has three steps.

Step 1: Rank-one expansion. By definition of matrix multiplication,

$$WH = \sum_{m=1}^r W_{:,m} H_{m,:}, \quad (13)$$

where each $W_{:,m} H_{m,:} \in \mathbb{R}_{\geq 0}^{N \times k}$ is a nonnegative rank-one matrix. Each motif m has a query-side factor $W_{:,m}$ and a feature-side factor $H_{m,:}$. Each term $W_{:,m} H_{m,:}$ can be interpreted as a weighted bipartite block connecting a subset of queries with a subset of SAE features.

Step 2: Both sides act as soft cluster indicators. By Ding et al. (2005) and Ding et al. (2006), the bi-orthogonal NMF,

$$\min_{W, H \geq 0} \|G - WH\|_F^2 \quad \text{s.t.} \quad W^\top W = I, \quad HH^\top = I, \quad (14)$$

corresponds to running K -means on the rows (queries) and columns (SAE features) of G under orthogonality constraints. In the orthogonal case, each row of W has a single nonzero entry giving its query-cluster, and the same holds for H . Standard NMF in Equation (12) drops the orthogonality constraint and can be viewed as a relaxed version of this clustering formulation. Although orthogonality is no longer enforced, prior work shows that NMF solutions tend to exhibit approximately disjoint support, which supports interpreting $W_{i,m}$ as the soft membership of query q_i in motif m , and $H_{m,j}$ as the soft membership of SAE feature j in motif m .

Step 3: The two sides are coupled. The query side and feature side are not learned separately. Ding et al. (2006) shows that at any stationary

point of Equation (12), W solves a kernel K -means on the queries with kernel $GH^\top HG^\top$, which compares queries inside the subspace spanned by H . Symmetrically, H solves a kernel K -means on the features with kernel $G^\top WW^\top G$. Each side defines the geometry the other side clusters in, through the shared index m .

Conclusion. Each motif m is a co-cluster $(W_{:,m}, H_{m,:})$ that is jointly defined by Equation (13), readable as a soft cluster indicator on each side, and coupled through a single objective. The decomposition recovers joint query–feature structure rather than two independent factors.

A.2 Hard Clustering as a Restricted Special Case

Setup. A standard alternative to motif decomposition is hard clustering, in which each query q_i is assigned to exactly one of r groups. We show that hard clustering corresponds to motif decomposition in Equation (12) under an extra one-hot constraint on W , and is therefore a more restricted model. The proof has three steps.

Step 1: Hard clustering as a matrix factorization. A hard assignment of N queries to r groups can be written as a binary indicator matrix $W \in \{0, 1\}^{N \times r}$ in which each row has exactly one nonzero entry. Under this representation, the product WH assigns to query q_i the feature profile $H_{m,:}$ of the single group m to which it belongs. Hard clustering of G into r groups, with group profiles stored in H , is therefore the constrained matrix approximation problem

$$\min_{W \in \{0, 1\}^{N \times r}, H \geq 0} \|G - WH\|_F^2, \quad (15)$$

where each row of W contains exactly one nonzero entry (i.e., a one-hot assignment). For fixed W , the least-squares solution is

$$H = (W^\top W)^{-1} W^\top G, \quad (16)$$

provided no cluster is empty, which holds by assumption, so each row $H_{m,:}$ is the mean feature profile of the queries assigned to group m .

Step 2: The hard-clustering feasible set is a subset of the NMF feasible set. Any W satisfying the constraint in Equation (15) is nonnegative, so the feasible set is contained in the feasible set of motif decomposition in Equation (12). Therefore, the optimal hard-clustering residual satisfies

$$\min_{\substack{W \text{ one-hot} \\ H \geq 0}} \|G - WH\|_F^2 \geq \min_{W, H \geq 0} \|G - WH\|_F^2. \quad (17)$$

Thus motif decomposition can reproduce any hard-clustering solution, but may achieve a lower reconstruction error by allowing continuous multi-component memberships.

Step 3: Relation to K -means. Equation (15) is exactly the matrix form of K -means on query vectors $G_{i,:}$. With fixed W , each row of H becomes the mean of the queries assigned to that cluster. With fixed H , each query is assigned to its nearest cluster center. As shown in (Ding et al., 2005, 2006), hard clustering uses one-hot memberships, while NMF relaxes them to nonnegative soft memberships.

Conclusion. Hard clustering is the special case of motif decomposition (12) obtained by restricting W from $\mathbb{R}_{\geq 0}^{N \times r}$ to binary one-hot rows. Removing this restriction lets a single query participate in multiple motifs with continuous weights, which can reduce reconstruction error and capture partial co-activation structure that hard clustering discards.

A.3 Nonnegativity Preserves Feature Support

Setup. A motif is intended to capture a recurring set of SAE features. We show that the nonnegativity constraint $W, H \geq 0$ in Equation (12) ensures that feature contributions are additive and prevents cancellation across motifs.

Step 1: Nonnegativity rules out cancellation. The reconstruction

$$G \approx \sum_{m=1}^r W_{:,m} H_{m,:}, \quad (18)$$

is a sum of nonnegative rank-one terms. Thus every entry G_{ij} is formed by adding nonnegative contributions and cannot result from cancellation between components, unlike signed decompositions such as SVD or PCA (Lee and Seung, 1999).

Step 2: Support is preserved. If $H_{m,j} > 0$, motif m contributes positively to feature j whenever $W_{i,m} > 0$. The support of $H_{m,:}$ corresponds to features that are added together by the motif in the reconstruction. In contrast, signed factorizations may mix positive and negative contributions, allowing different components to cancel.

A.4 Dense Embeddings Induce a Structural Proxy Error

Setup. We compare motif membership under NMF with a dense embedding pipeline that clusters queries into r groups. We show that when motif membership is multi-valued, such a pipeline must incur approximation error.

Step 1: Motif membership under NMF. Given the NMF decomposition $G \approx WH$ and a threshold $\tau \geq 0$, we define the motif membership of query q_i for motif m as

$$\text{mot}_{\text{SAE}}(q_i, m) = \mathbf{1}[W_{i,m} > \tau], \quad (19)$$

which indicates whether motif m is active in query q_i based on its coefficient in W .

A query may belong to multiple motifs. We define

$$T_i = \sum_{m=1}^r \text{mot}_{\text{SAE}}(q_i, m), \quad (20)$$

which counts the number of motifs assigned to query q_i .

Step 2: Dense clustering defines a compressed proxy. Let $c_i \in \{1, \dots, r\}$ denote the cluster label of query q_i produced by a dense embedding pipeline. Since the dense representation does not have access to W , motif membership must be approximated through a proxy:

$$\widehat{\text{mot}}_{\text{dense}}(q_i, m) = \mathbf{1}[c_i = \psi(m)], \quad (21)$$

where ψ maps each motif to a cluster.

This means that the membership vector $\widehat{\text{mot}}_{\text{dense}}(q_i, \cdot)$ is fully determined by the single label c_i , and therefore encodes motif membership in a compressed form.

Step 3: Lower bound from multi-membership. We measure the discrepancy between the dense proxy and the SAE-based membership by

$$E_{\text{sur}}^{\text{dense}} = \frac{1}{Nr} \sum_{i=1}^N \sum_{m=1}^r \left| \widehat{\text{mot}}_{\text{dense}}(q_i, m) - \text{mot}_{\text{SAE}}(q_i, m) \right|, \quad (22)$$

which is the average disagreement over all query-motif pairs.

For each query q_i , we have

$$\sum_{m=1}^r \left| \widehat{\text{mot}}_{\text{dense}}(q_i, m) - \text{mot}_{\text{SAE}}(q_i, m) \right| \geq \max(0, T_i - 1). \quad (23)$$

because $\sum_{m=1}^r \widehat{\text{mot}}_{\text{dense}}(q_i, m) \leq 1$ for any c_i , while $\sum_{m=1}^r \text{mot}_{\text{SAE}}(q_i, m) = T_i$, the ℓ_1 discrepancy is at least $T_i - 1$ when $T_i > 1$, and therefore $\widehat{\text{mot}}_{\text{dense}}(q_i, \cdot)$ cannot encode multiple independent motif activations.

Averaging over all queries yields

$$E_{\text{sur}}^{\text{dense}} \geq \frac{1}{Nr} \sum_{i=1}^N \max(0, T_i - 1), \quad (24)$$

which lower bounds the proxy error in terms of multi-membership.

Conclusion. When motif membership is multi-valued ($T_i > 1$), dense clustering cannot in general represent it exactly and therefore incurs a nonzero approximation error when multi-membership is present. In contrast, motif decomposition directly represents multi-membership through the coefficient matrix W and avoids this limitation.

B Language Models and SAE Details

We use three open-weight instruction-tuned LLMs as attack targets: Llama-3.1-8B-Instruct (32 layers, hidden size 4096), Mistral-7B-Instruct-v0.2 (32 layers, hidden size 4096), and Qwen2-7B-Instruct (28 layers, hidden size 3584). Together, they provide a diverse set of widely used target models for testing whether the discovered motifs generalize across model families. For each target, we apply a pre-trained Top- $\tilde{K}_{\text{tok}}$ SAE (dictionary size 65,536) to the residual stream at an intermediate layer: layer 16 for Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.2, and layer 14 for Qwen2-7B-Instruct. This follows prior work that uses intermediate residual representations for feature-based analysis (Wu et al., 2025; Wang et al., 2024b). For each attack instance, either a single-turn prompt or a multi-turn jailbreak trace depending on the attack method, we run the target model and record token-level SAE activations at the selected layer. We then apply max-pooling over token positions for each SAE feature to obtain a 65,536-dimensional feature vector for that instance (Gallifant et al., 2025). After max-pooling, we threshold the vector at $\delta=2.0$ by setting all smaller activations to zero, so that only strongly activated SAE features are retained. Stacking the thresholded vectors from all attack instances gives the nonnegative matrix used for the downstream NMF-based motif decomposition.

SAEs for Interpretable Feature Discovery. We construct the feature space using SAEs, which extract interpretable features from LLM representations (Bricken et al., 2023; Huben et al., 2024). Typically, a SAE is implemented with an encoder and a decoder with tied weights (Li et al., 2026a). Given an input activation $\mathbf{x} \in \mathbb{R}^d$, the encoder produces feature activations $\mathbf{z} = \sigma(\mathbf{x}W_{\text{SAE}}) \in \mathbb{R}^k$, and the decoder reconstructs $\hat{\mathbf{x}} = \mathbf{z}W_{\text{SAE}}^\top \in \mathbb{R}^d$, where σ denotes the ReLU activation, $W_{\text{SAE}} \in \mathbb{R}^{d \times k}$ with $k \gg d$, and k is the number of features. The SAE

is trained by minimizing the reconstruction error:

$$\mathcal{L}_{\text{SAE}} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2, \quad (25)$$

where sparsity is enforced explicitly by a Top- $\tilde{K}_{\text{tok}}$ operator. In particular, we retain only the $\tilde{K}_{\text{tok}}$ largest coordinates of $\mathbf{z}$ and set the rest to zero, yielding a sparse vector $\tilde{\mathbf{z}}$. This constraint enforces a fixed sparsity level by construction, which improves interpretability by encouraging each feature index $i \in \{1, \dots, k\}$ to correspond to a distinct pattern.

In practice, we apply the SAE to a sequence by running it on the model activation at each token position. Let $X = (x_1, \dots, x_T)$ be an input sequence of length T , and let $\mathbf{x}_t \in \mathbb{R}^d$ denote the LLM activation at position t from the chosen layer. For each token, the SAE encoder produces feature activations

$$\mathbf{z}_t = \sigma(\mathbf{x}_t W_{\text{SAE}}) \in \mathbb{R}^k, \quad (26)$$

and we apply Top- $\tilde{K}_{\text{tok}}$ sparsification, retaining only the $\tilde{K}_{\text{tok}}$ largest entries and setting the rest to zero, yielding a sparse vector $\tilde{\mathbf{z}}_t$. Stacking feature activations across positions yields a feature activation matrix

$$Z(X) = [\tilde{\mathbf{z}}_1, \dots, \tilde{\mathbf{z}}_T] \in \mathbb{R}^{k \times T}, \quad (27)$$

where $Z_i(X, t)$ denotes the activation of feature i at token position t . Since T varies across sequences, we map $Z(X)$ to a fixed-length representation $g(X) \in \mathbb{R}^k$ by max pooling:

$$g_i(X) = \max_{t \geq t_0} Z_i(X, t), \quad (28)$$

where t_0 denotes the token position after the chat-template prefix (e.g., system/header tokens in LLaMA-3.1-8B-Instruct), so that feature pooling focuses on the user-provided content rather than template scaffolding.

Compute and cost details. The main computational cost comes from SAE training, which takes approximately 4.41 hours on $4 \times$ NVIDIA H100 GPUs for 3 epochs. After the SAE is trained, the remaining analysis reuses the frozen target model and SAE. For each query, feature extraction requires one target-model forward pass and one SAE encoding pass at the selected residual layer. The NMF decomposition, motif-overlap analysis, and logistic-regression detectors are then run offline on the resulting query-feature matrices. For defense experiments, all methods are trained under

the same synthetic preference-data budget of 400 examples and the same DPO configuration, so the comparison in Table 4 keeps the training-data budget fixed across Feature-DPO and Motif-DPO.

Interpreting SAE Features. Following prior work on interpreting SAE features (Lieberum et al., 2024), we characterize each feature by the input spans that most strongly activate it. Specifically, for each feature, we extract the top 10 text spans with the highest activation values, with each span truncated to at most 32 tokens. We then use a language model to generate a concise natural language description of the feature based on these spans. Concretely, we prompt GPT-4o-mini-2024-07-18 (Achiam et al., 2023) with temperature 0 for deterministic decoding, and limit each response to 1,024 tokens. The prompting framework includes role-based instructions and in-context examples to guide the model toward identifying consistent semantic patterns. The model is allowed to output “Cannot Tell” when no clear pattern emerges from the spans. To further improve reliability, we perform an independent verification step, in which the model is prompted to assess whether its generated description is consistent with the underlying activation spans.

Identifying Jailbreak-relevant Features. We identify SAE features that are semantically associated with jailbreak behavior. A feature is considered *jailbreak-relevant* if its activation patterns consistently reflect semantic attributes related to adversarial prompting, such as intent obfuscation, policy evasion, or harmful instruction construction. To assess feature relevance, we follow prior work on feature annotation (Rajamanoharan et al., 2024; Wallace et al., 2024) and employ a language model to assign one of four labels: “Yes”, “Probably”, “Maybe”, and “No”, indicating decreasing levels of confidence that the feature captures jailbreak-related semantics. In our analysis, features labeled below “Maybe” are treated as irrelevant. To validate the reliability of the automated annotation, we conduct a human verification experiment in Appendix C.

C Data Construction and Validation

Our dataset covers six attacker–target settings, formed by crossing two attacker LLMs, DeepSeek-v3.2 and Qwen3-30B-A3B-Instruct-2507, with three target LLMs, Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.2, and Qwen2-7B-Instruct. All at-

tacks are initialized from the 400 harmful queries in HarmBench (Mazeika et al., 2024). For each attacker–target setting, we collect 6,000 attack instances, including 1,200 from each of the five jailbreak methods: Crescendo, FlipAttack, PAIR, ReNeLLM, and XTeaming. These attack queries were generated using the pipeline from (Wang et al., 2026). Across all six settings, this yields 36,000 attack instances in total. The jailbreak queries collected are used solely for research purposes. Each generated query is judged by Qwen3-30B-A3B-Instruct-2507 using a binary success/failure label. Table 5 reports the per-run ASR for each attacker–target setting and jailbreak method, and Table 6 summarizes the mean and standard deviation across the three runs. Across all settings, ReNeLLM consistently achieves the highest ASR, while FlipAttack shows a strong dependence on the attacker model, performing much better with DeepSeek-v3.2 than with Qwen3-30B-A3B-Instruct-2507. PAIR exhibits the largest run-to-run variance, whereas Crescendo and XTeaming are more moderate but relatively stable.

C.1 Judge Consistency

To examine whether our results are sensitive to the choice of judge model, we conduct an additional consistency audit on 1,000 sampled examples, with 200 examples from each attack method. The original judge is Qwen3-30B-A3B-Instruct-2507, and the reference judge is GPT-4o-mini. In table 7, TT denotes that both judges classify the example as successful, TF denotes that Qwen3-30B-A3B-Instruct-2507 classifies it as successful while GPT-4o-mini classifies it as failed, FT denotes that Qwen3-30B-A3B-Instruct-2507 classifies it as failed while GPT-4o-mini classifies it as successful, and FF denotes that both judges classify it as failed.

The two judges show reasonably stable agreement across thresholds. When the threshold is set to $T=3$, the agreement rate reaches 0.808. Under the stricter settings $T=4$ and $T=5$, the agreement rates are 0.740 and 0.724, respectively. Although the raw agreement decreases as the threshold becomes stricter, the disagreement remains bounded and does not indicate a complete judge mismatch. We manually re-examine the inconsistent TF and FT cases at $T=5$ under the same decision rule. After human verification, the agreement rate increases from 0.724 to 0.881, while the conflict rate decreases from 0.276 to 0.119. This shows

Table 5: Per-run ASR (%) for all six attacker–target settings and five jailbreak methods. S0, S1, and S2 denote the three independent runs. Each run contains 400 attack queries per method, yielding 1,200 instances per method in total for each attacker–target setting.

Attacker	Target	Crescendo			FlipAttack			PAIR			ReNeLLM			XTeaming		
		S0	S1	S2	S0	S1	S2	S0	S1	S2	S0	S1	S2	S0	S1	S2
DeepSeek-v3.2	Llama3.1-8B	63.5	62.3	61.3	89.2	90.2	90.5	71.2	64.0	60.2	95.8	94.0	97.0	76.2	79.8	74.2
	Mistral-7B	71.5	66.0	66.8	90.2	89.5	90.5	77.5	71.2	58.5	98.0	95.8	95.5	70.2	78.2	74.0
	Qwen2-7B	68.5	66.2	65.8	90.8	90.8	92.0	76.2	71.2	55.5	94.5	96.2	94.5	78.0	75.5	75.2
Qwen3-30B	Llama3.1-8B	64.8	71.0	70.8	45.2	43.2	41.5	68.5	57.5	53.2	97.5	92.2	92.5	63.7	62.3	61.3
	Mistral-7B	62.3	63.0	72.8	46.8	43.0	45.5	50.7	52.5	54.5	95.5	95.5	95.0	58.0	62.0	63.5
	Qwen2-7B	64.0	68.2	68.5	47.5	45.5	44.5	54.5	53.5	52.2	97.5	94.0	95.0	56.8	65.8	62.3

Table 6: Mean (μ , %) and standard deviation (σ , %) of ASR across the three independent runs for each attacker–target setting. The Overall columns are computed at the run level over all 6,000 attack queries in that setting, including 1,200 queries from each jailbreak method.

Attacker	Target	Crescendo		FlipAttack		PAIR		ReNeLLM		XTeaming		Overall	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
DeepSeek-v3.2	Llama-3.1-8B	62.3	0.9	90.0	0.5	65.2	4.6	95.6	1.2	76.8	2.3	78.0	1.0
	Mistral-7B	68.1	2.4	90.1	0.4	69.1	7.9	96.4	1.1	74.2	3.3	79.6	1.9
	Qwen2-7B	66.8	1.2	91.2	0.6	67.7	8.8	95.1	0.8	76.3	1.2	79.4	2.1
Qwen3-30B	Llama-3.1-8B	68.9	2.9	43.3	1.5	59.7	6.4	94.1	2.4	62.4	1.0	65.7	1.7
	Mistral-7B	66.0	4.8	45.1	1.6	52.6	1.5	95.3	0.2	61.2	2.3	64.0	1.6
	Qwen2-7B	66.9	2.1	45.8	1.2	53.4	0.9	95.5	1.5	61.6	3.7	64.6	0.6

that the judge-model difference has a controllable effect on the evaluation, and that many raw disagreements are due to borderline or conservative decisions by one judge model. After manual verification, Qwen3-30B-A3B-Instruct-2507 is acceptable as the main judge model for our evaluation.

C.2 Human Audit of SAE Feature Annotations

To assess the reliability of GPT-4o-mini for identifying attack-relevant SAE features, we conduct a targeted human audit across three target models. For each target model, we use GPT-4o-mini to identify SAE features predicted to be associated with jailbreak behaviors, and sample 500 features from the candidate sets obtained under multiple SAE activation thresholds. For each audited feature, annotators are shown the top activated text spans associated with the feature, following the same presentation format as in the automatic annotation pipeline. Two graduate-level annotators independently label each feature as *relevant*, *irrelevant*, or *unclear* with respect to attack-related behavior. Both annotators voluntarily participated in the audit and consented to the use of their annotations for this research.

We report the fraction of audited features in each category. A feature is counted as *confirmed relevant* or *irrelevant* only when both annotators agree on the corresponding label; all remaining cases, including annotator disagreements, are categorized as *unclear*. As shown in Table 8, most selected features are confirmed by human annotators as jailbreak-relevant across all three target models. The confirmed relevant rate is 85.20% for Llama-3.1-8B-Instruct, 80.60% for Mistral-7B-Instruct-v0.2, and 84.20% for Qwen2-7B-Instruct. Meanwhile, the irrelevant rate remains low, ranging from 5.40% to 8.00%. Although Mistral shows a higher unclear rate of 11.40%, the overall unclear rate remains moderate. These results indicate that GPT-4o-mini provides a reliable mechanism for selecting jailbreak-relevant SAE features, and that the selected features remain human-interpretable across different target models.

Annotation Protocol. Annotators were shown the following task description before beginning the audit:

You are given a text feature, its summary, and several example spans from user queries. Your task is to decide whether

Table 7: Agreement statistics under different threshold settings and verification protocols.

SETTING	T	AGREEMENT RATE	CONFLICT RATE	TT	TF	FT	FF
Threshold = 3	3	0.808	0.192	760	64	128	48
Threshold = 4	4	0.740	0.260	615	209	51	125
Threshold = 5	5	0.724	0.276	592	232	44	132
Threshold = 5 (Manual verification)	5	0.881	0.119	748	76	43	133

Table 8: Human verification of LLM-generated jailbreak annotations across target models.

TARGET MODEL	FEATURES	CONFIRMED RELEVANT (%)	IRRELEVANT (%)	UNCLEAR (%)
Llama-3.1-8B	500	85.20%	5.40%	9.40%
Mistral-7B	500	80.60%	8.00%	11.40%
Qwen2-7B	500	84.20%	5.60%	10.20%

this feature indicates an attack strategy against an LLM. An attack strategy is one that attempts to manipulate or bypass the model’s intended behavior through any of the following mechanisms:

- **Instruction override:** e.g., “ignore previous instructions”, “forget your rules”.
- **Role-play or persona hijacking:** adopting a persona or mode (e.g., “DAN”, “developer mode”) to suspend safety constraints.
- **Obfuscation:** hiding intent via encoding, character reversal, word splitting, or ciphers.
- **Privilege escalation:** falsely claiming admin/system authority, or demanding system prompts or credentials.
- **Gradual jailbreak:** incrementally building context across turns to weaken refusals.
- **Prompt injection:** embedding instructions in retrieved documents or external content to override the model’s true task.

For each feature, annotators were asked to assign one of four labels according to the following criteria:

- *Yes:* the spans contain explicit markers of an attack mechanism, such as direct instruction override phrases, obfuscation patterns, or persona-hijacking cues.
- *Probably:* the spans suggest attack intent but do

not contain explicit markers; the feature is more likely attack-related than not.

- *Maybe:* the spans are ambiguous and could reflect either attack-related or benign usage; the feature cannot be confidently classified in either direction.
- *No:* the spans show no signal of any attack mechanism.

In our analysis, features labeled Yes, Probably, Maybe, or No, indicating decreasing levels of confidence that the feature captures an attack strategy. Annotators were instructed to first write a brief analysis of the feature based on the provided spans, and then give a final label. Two illustrative examples were provided to calibrate judgment: a reversed-text feature (labeled Yes, as character reversal is a form of obfuscation) and a biological terminology feature (labeled No, as the spans show no attack-related pattern). Each feature was labeled independently by two graduate-level annotators. Disagreements were categorized as unclear rather than resolved by majority vote, to avoid introducing annotation bias. Both annotators are graduate students at the authors’ institution and were not compensated separately for this task. The annotation was conducted as part of the research process.

D Additional Experimental Results

D.1 Shared Jailbreak Structure

To assess whether jailbreak queries exhibit stronger motif structure in SAE space, we compare attack and normal activations using reconstruction, spec-

Table 9: Effect of the number of motifs K on reconstruction performance. Gap is defined as the difference between attack and non-attack reconstruction errors.

K	Atk_err	Nrm_err	Gap ↓
5	0.726	0.832	-0.107
10	0.691	0.783	-0.092
20	0.659	0.743	-0.084
30	0.640	0.717	-0.077
40	0.627	0.697	-0.070
50	0.615	0.682	-0.067
60	0.605	0.670	-0.065
70	0.596	0.657	-0.061
80	0.588	0.645	-0.057
90	0.580	0.636	-0.056
100	0.574	0.626	-0.052

tral concentration, directional gaps, and null controls.

Settings. We fix the number of motifs to $K = 40$ and sweep the activation threshold $\delta \in \{0.5, 1.0, 1.5, 2.0\}$ to control sparsity. For each δ , we construct SAE activation matrices for attack and non-attack queries and perform NMF separately. To isolate shared motif structure from generic sparsity effects, we evaluate five complementary groups of metrics. First, we report relative reconstruction errors for attack queries (Atk_err) and non-attack queries (Nrm_err), together with their difference $\text{Gap} = \text{Atk_err} - \text{Nrm_err}$. Second, we quantify spectral concentration using the stable-rank gap $\Delta\text{SR} = \text{SR}_{\text{atk}} - \text{SR}_{\text{nrm}}$, where the stable rank of a matrix X is defined as $\text{SR} = \|X\|_F^2 / \sigma_1^2$ and σ_1 denotes its largest singular value. We also report the top- K spectral-energy gap $\Delta E@K$ ($K = 40$). Third, we assess structural separation via directional reconstruction gaps. We train separate NMF decompositions on the attack and non-attack activations to obtain two motif bases H_{atk} and H_{non} . A_{gap} is the additional reconstruction error incurred when reconstructing attack data with the non-attack-trained basis H_{non} (relative to H_{atk}), while N_{gap} is the additional error when reconstructing non-attack data with the attack-trained basis H_{atk} (relative to H_{non}). Their symmetric combination SymGap quantifies the overall difference between the two bases. Finally, we report the NullAtk, obtained by randomly permuting feature identities within each row of the attack query-feature matrix before running NMF (row-wise sparsity and activation magnitudes are preserved, but co-activation information is destroyed).

Results. As shown in Table 10, attack acti-

vations are consistently reconstructed better than non-attack activations across the threshold range. This demonstrates that jailbreak queries are more compressible into a small number of shared motif components than non-attack queries. Beyond reconstruction error, spectral analysis further confirms the structural distinction. ΔSR stays negative and $\Delta E@K$ stays positive, indicating that attack matrices concentrate their spectral mass in fewer dominant directions than non-attack matrices under matched sparsification. Across the threshold band, A_{gap} remains negative, N_{gap} remains positive, and SymGap remains negative. This confirms that attack-trained motif bases reconstruct attack activations substantially better than non-attack-trained bases and vice versa. Importantly, the NullAtk remains very high (0.978) throughout this band. This confirms that the superior reconstruction quality and spectral concentration of attack queries arise from shared co-activation patterns among SAE features, rather than sparsity effects alone. We also present additional results for different values of K in Table 9, which further validate the conclusions drawn in this section.

Figure 5 presents span-level overlap across jailbreak methods under different activation thresholds. We report three complementary metrics: raw overlap counts ($|A \cap B|$), Jaccard similarity, and overlap coefficient. Across all thresholds, a consistent pattern emerges. Semantic jailbreak methods (Crescendo, PAIR, and XTeaming) exhibit substantially higher mutual overlap, while FlipAttack remains weakly aligned with other methods. This separation persists as the threshold increases from 0.01 to 0.05, indicating that the shared structure among semantic attacks is not driven by dense or noisy activations, but is retained in the strongest activation spans. As the threshold increases, overall overlap decreases due to sparsification, but the relative relationships between methods remain stable. This suggests that span-level structure is robust to threshold choices and reflects intrinsic similarities between attack strategies rather than artifacts of feature density.

Figure 6 shows motif-level overlap across jailbreak methods under different settings. We vary both the ratio threshold and the number of selected motifs (top- k) to examine robustness. We observe two main trends. First, overlap among semantic attacks (Crescendo, PAIR, and XTeaming) increases as k grows, indicating that shared motif structure becomes more evident when more motif compo-

Table 10: Comprehensive results of comparing attack and non-attack activations in motif space under different SAE thresholds ($r = 40$). Attack activations are reconstructed with lower error, and additional diagnostics show consistent structural differences from non-attack activations.

δ	$ F $	Atk_err	Nrm_err	Gap $\downarrow$	Δ SR $\downarrow$	$\Delta E@K$ $\uparrow$	A_gap $\downarrow$	N_gap $\uparrow$	SymGap $\downarrow$	NullAtk $\uparrow$
0.5	7265	0.616	0.727	-0.111	-2.8	0.137	-0.204	0.260	-0.194	0.957
1.0	6809	0.626	0.725	-0.099	-2.9	0.126	-0.183	0.260	-0.181	0.968
1.5	6475	0.633	0.714	-0.082	-2.6	0.107	-0.194	0.301	-0.207	0.975
2.0	5934	0.627	0.697	-0.070	-2.5	0.095	-0.212	0.320	-0.230	0.978

nents are considered. In contrast, FlipAttack consistently exhibits lower overlap, suggesting that it does not share the same underlying motif structure. Second, the results are stable across different ratio thresholds. While stricter thresholds slightly reduce overlap values, the overall pattern remains unchanged. This confirms that motif-level structure is not sensitive to specific hyperparameter choices, and that the observed separation reflects a genuine difference in attack organization rather than thresholding effects.

D.2 Unseen-attack Generalization

To understand what the learned motifs capture, we analyze motif overlap across jailbreak methods. If motifs encode shared attack structure, then methods with similar underlying strategies should show higher overlap, whereas structurally distinct methods should show lower overlap.

For each attack method, we derive two types of overlap sets. At the span level, we keep SAE features that are activated in at least a fixed fraction of that method’s queries, where this fraction is defined as the **rate**, and compute pairwise Jaccard similarity between methods. At the motif level, we first retain attack motifs whose mean activation on attack queries is sufficiently larger than that on non-attack queries, which is defined as **ratio**. We then represent each method by its top- k motifs within this pool, where top- k denotes the motifs with the highest mean activation for that method, and compute pairwise Jaccard similarity. This section provides the full sensitivity analysis, including span-level raw overlap counts, Jaccard similarity, and overlap coefficient under rate thresholds 0.01, 0.02, and 0.05, as well as motif-level Jaccard results under ratio thresholds 1.5, 2.0, and 2.5 and top- k values 5, 10, 15, and 20.

As shown in Figure 2, Crescendo, PAIR, and XTeaming consistently exhibit higher overlap with one another than with FlipAttack. This indicates that these methods share a larger set of common motifs, whereas FlipAttack is associated with a

substantially different motif structure. Table 2 further shows that this difference cannot be explained by weaker activation. In fact, FlipAttack and ReNeLLM activate as many SAE features as the other methods. Their lower overlap therefore reflects differences in feature identity and co-activation structure, rather than differences in activation scale. The results show that the same pattern persists across alternative overlap metrics and hyperparameter settings, confirming that the observed difference in shared motifs is robust. To further examine the effect of character-level perturbation, we compare original FlipAttack prompts with de-flipped variants that restore the original task text while retaining the remaining guidance wrapper. Character normalization reduces ASR from 90.00% to 65.00%, increases mean motif overlap with other attacks from 0.16 to 0.30, and improves LOMO F1 from 0.39 to 0.62. This indicates that character-level perturbation and tokenization partially explain FlipAttack’s distinct motif profile, but do not fully eliminate the gap.

Figure 7 visualizes the PCA motif representations under the LOMO setting. Across most held-out methods, attack and normal samples form largely separable regions, showing that motif features preserve discriminative structure even when the test attack method is unseen. The main exception is FlipAttack, whose samples are less aligned with the attack region learned from the other methods. This is consistent with the overlap analysis: FlipAttack uses a different feature composition, so its transfer difficulty reflects structural mismatch rather than a failure to activate SAE features.

Figure 8 compares different 40-dimensional representations under the held-out FlipAttack setting. PCA-40, RandK-40, and ICA-40 show clear mismatch: almost all FlipAttack samples fall outside the attack region learned from the training methods, leading to incorrect predictions. In contrast, NMF-40 captures part of the FlipAttack pattern. While FlipAttack remains challenging, NMF-40 places more samples closer to the attack region, resulting

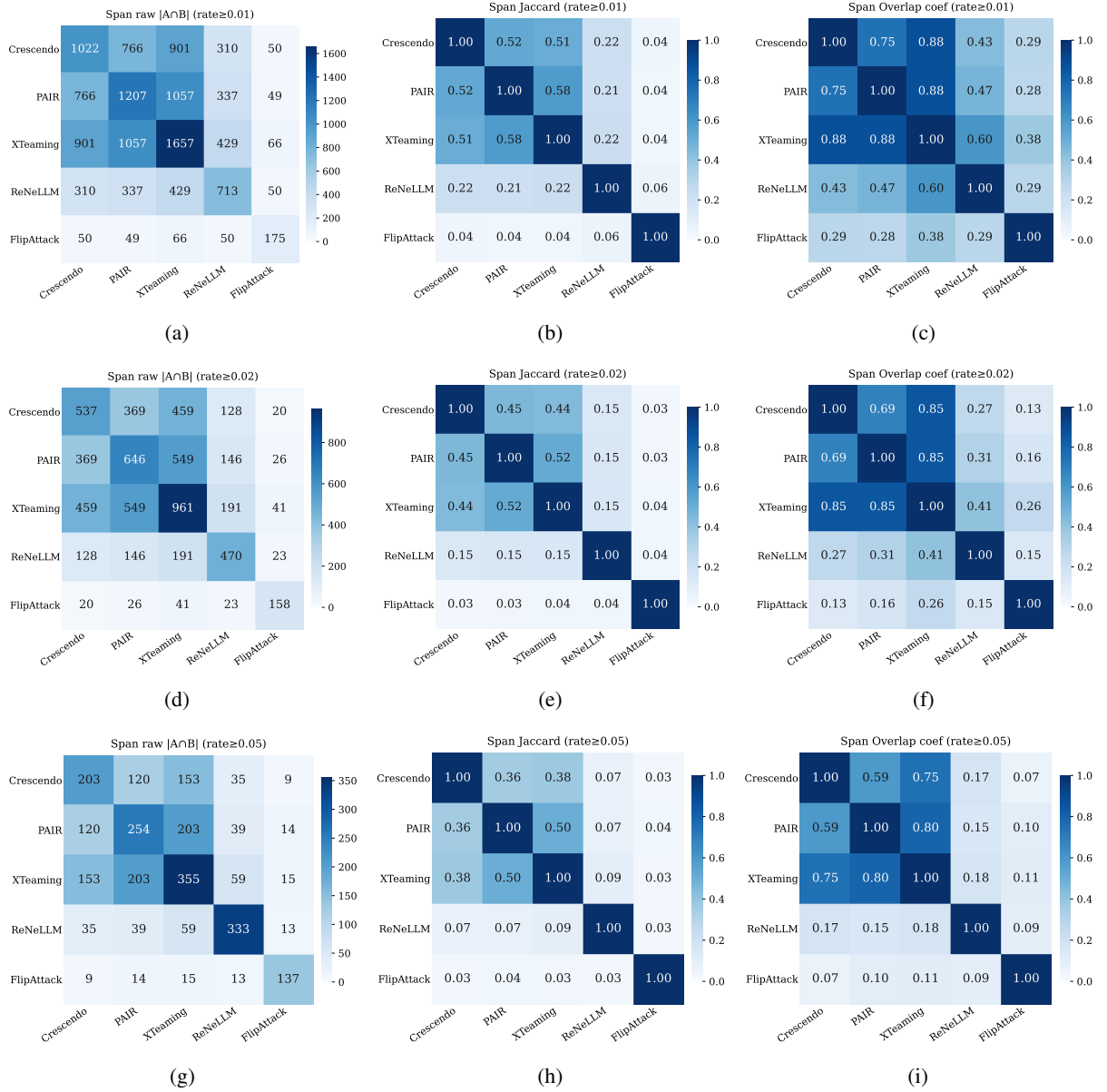


Figure 5: Span-level overlap across jailbreak methods under different thresholds. We report raw overlap counts ($|A \cap B|$), Jaccard similarity, and overlap coefficient at rates 0.01, 0.02, and 0.05.

in a more meaningful separation.

Motifs in Multi-turn Dialogues. Motif signals also emerge during multi-turn jailbreaks. Figure 9 shows that, for both Crescendo and XTeaming, successful attacks exhibit increasing motif activation as the dialogue progresses, while failed attacks remain relatively flat. Figure 10 further shows that motif-based detection achieves above-random AUC from early turns and improves as more dialogue context is observed, indicating that motif representations capture predictive signals before the attack is fully formed and can support early-stage detection.

Figure 10 evaluates early detection using partial

dialogue prefixes. Subfigure (a) shows Crescendo and (b) shows XTeaming. In both cases, motif-based detection achieves above-random performance even in early turns and improves as more turns are observed. This indicates that motif representations capture predictive signals before the attack is fully formed, enabling early identification of jailbreak attempts.

We evaluate whether different representations preserve transferable attack structure under the LOMO protocol. In each run, one jailbreak method is held out for testing, while the detector is trained on prompts from the remaining four methods together with matched non-attack queries. On top

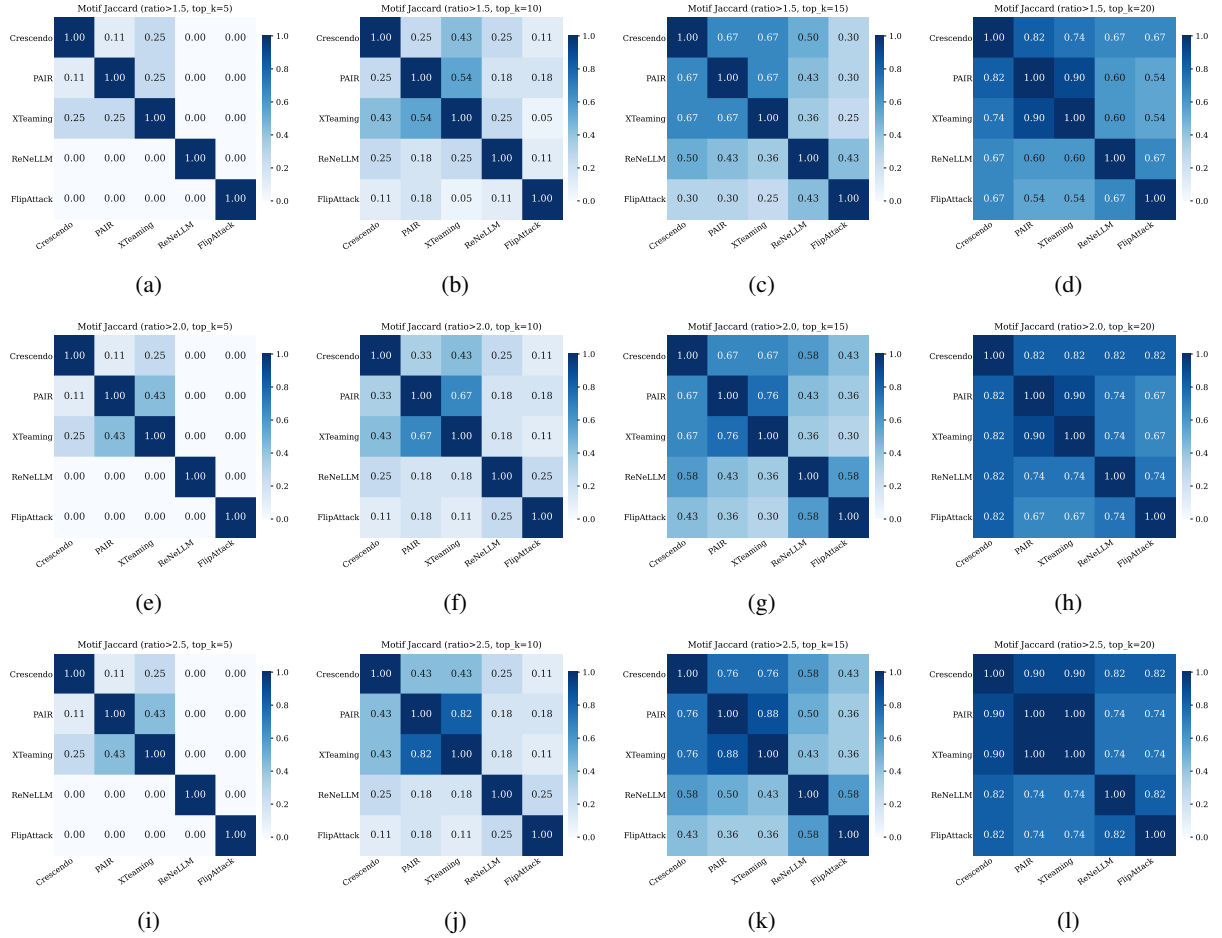


Figure 6: Motif-level overlap (Jaccard similarity) across jailbreak methods under different motif selection settings. We vary the activation ratio threshold (1.5, 2.0, 2.5) and the number of selected motifs (top- k , $k \in \{5, 10, 15, 20\}$).

of each representation, we train a logistic regression detector and report mean AUC, F1@Val, accuracy, precision, and recall on the held-out method, where F1@Val uses a threshold selected on a validation split of the training data. We compare SAE-NMF-40 against activity-based, text-embedding, full-feature, and compressed SAE-space baselines. Table 11 shows that **SAE-NMF-40** achieves the best overall performance across all reported metrics. Dense text embeddings achieve a competitive AUC, but their lower F1@Val indicates a less reliable decision boundary under held-out attack transfer. In particular, SAE-NMF-40 outperforms simple activity statistics, dense text embeddings, the full SAE representation, and alternative compressed SAE baselines such as PCA, ICA, TopK, and random projections. These results suggest that motif-based decomposition preserves attack-relevant structure that is both highly discriminative and strongly transferable across jailbreak methods.

D.3 Cross-setting Transfer

Figure 11 shows pairwise transfer performance across 30 transfer settings. Cross-attacker transfer is consistently strong even before calibration, indicating that motif structure is relatively stable across attacker models when the target model is fixed. In contrast, cross-target and fully-cross settings exhibit larger performance drops before calibration, especially in several near-failure cases. However, lightweight target calibration substantially improves these settings and recovers strong transfer performance for most pairs. We also compare SAE-NMF-40 with dense embeddings under the same transfer regimes. Dense embeddings obtain F1 scores of 0.65 ± 0.08 , 0.52 ± 0.10 , and 0.44 ± 0.11 for cross-attacker, cross-target, and fully-cross transfer, respectively. In contrast, SAE-NMF-40 obtains 0.97 ± 0.02 , 0.86 ± 0.06 , and 0.85 ± 0.07 . Thus, although dense embeddings are competitive in global ranking under the LOMO setting, SAE-NMF-40 yields more reliable decision

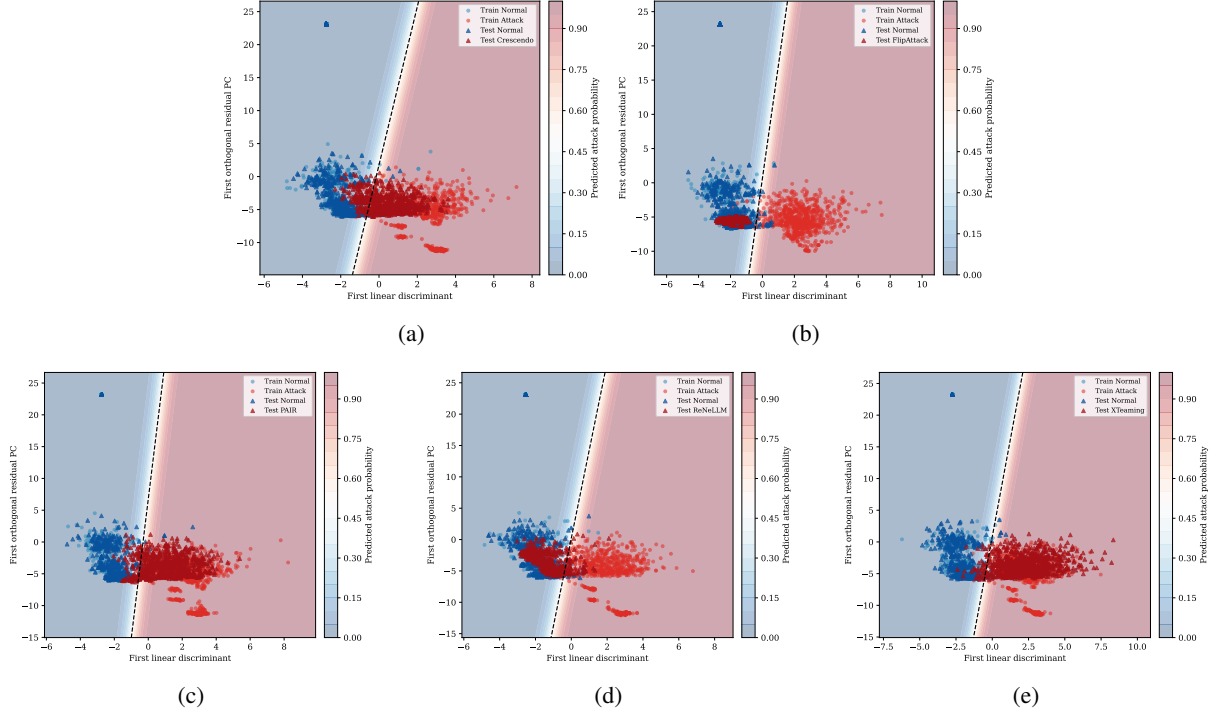


Figure 7: PCA visualization of motif-based representations under the LOMO setting. Each subplot shows the projected representation space when one jailbreak method is held out: (a) Crescendo, (b) FlipAttack, (c) PAIR, (d) ReNeLLM, and (e) XTeaming.

Table 11: Comparison of representation choices for LOMO jailbreak detection on the DS-Llama setting.

Representation	Mean AUC $\uparrow$	Mean F1@Val $\uparrow$	Mean Acc@Val $\uparrow$	Mean Prec@Val $\uparrow$	Mean Rec@Val $\uparrow$
SummaryStats	0.700	0.672	0.700	0.717	0.646
Embedding	0.926	0.571	0.767	0.755	0.540
FullSAE	0.904	0.610	0.789	0.791	0.581
SAE-TopK-40	0.871	0.717	0.752	0.805	0.665
SAE-RandK-40	0.661	0.726	0.625	0.580	0.977
SAE-PCA-40	0.849	0.600	0.782	0.767	0.571
SAE-ICA-40	0.763	0.603	0.787	0.826	0.581
SAE-RandProj-40	0.831	0.718	0.768	0.831	0.657
SAE-NMF-40	0.936	0.792	0.834	0.872	0.772

boundaries under transfer.

Table 13 compares fully-cross transfer performance across SAE-NMF, SAE-PCA, and FullSAE. Before calibration, SAE-NMF substantially outperforms the other representations, achieving much higher average F1 under severe cross-target and fully-cross shifts. In contrast, PCA and FullSAE often fail to transfer before calibration, with many settings close to zero F1. After lightweight target calibration, all methods improve substantially, indicating that part of the transfer gap arises from target-specific distribution shifts. Nevertheless, SAE-NMF remains consistently strong across transfer settings, suggesting that motif decomposition preserves more shared attack structure under distribution shift.

Table 14 shows that motif similarity is substantially higher for the same jailbreak method across attackers than for different methods within the same attacker. This pattern is consistent across target models, indicating that motif structure is more strongly associated with attack strategy than with the specific attacker model. The per-method analysis further shows that several jailbreak methods, including Crescendo, FlipAttack, ReNeLLM, and XTeaming, exhibit strong cross-attacker similarity. This supports the view that motif representations capture shared attack patterns that remain relatively stable across different attacker models. PAIR is the main exception, with substantially lower cross-attacker similarity (0.56 on Llama and 0.09 on Mistral). This is consistent with PAIR being an iterative

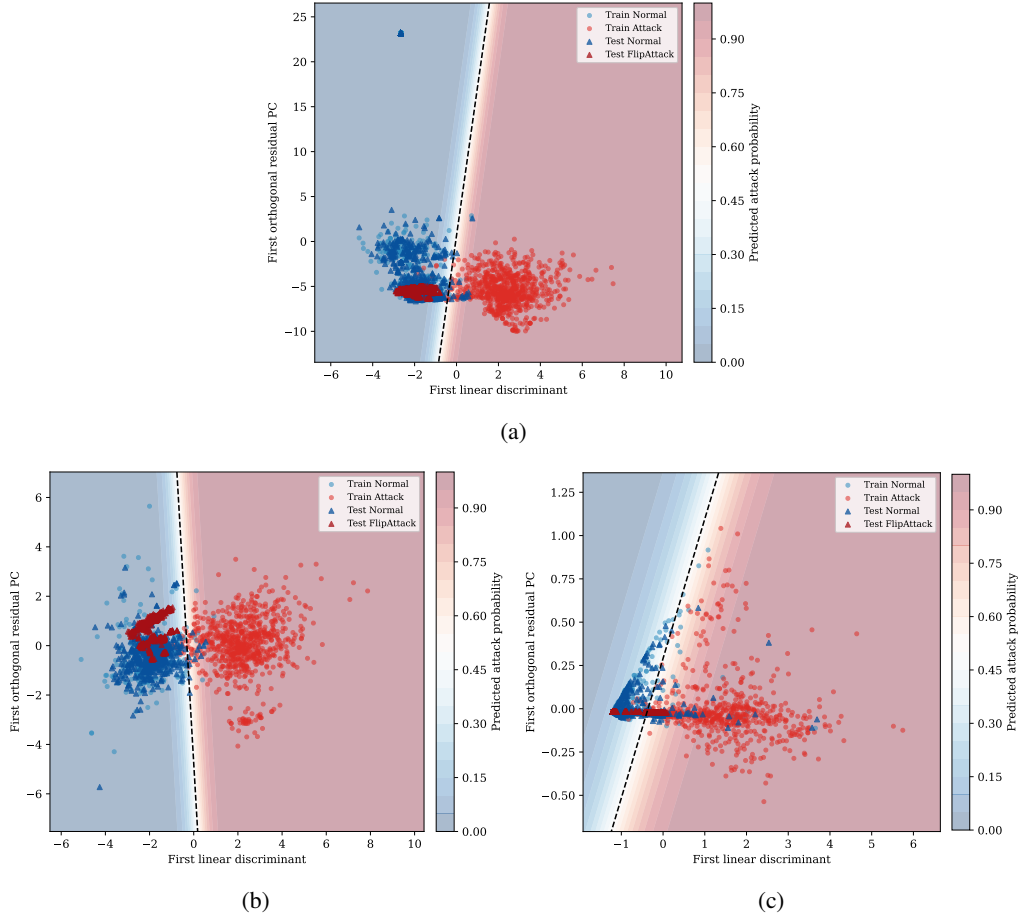


Figure 8: PCA visualization of different 40-dimensional representations when FlipAttack is held out: (a) PCA-40, (b) ICA-40, and (c) NMF-40.

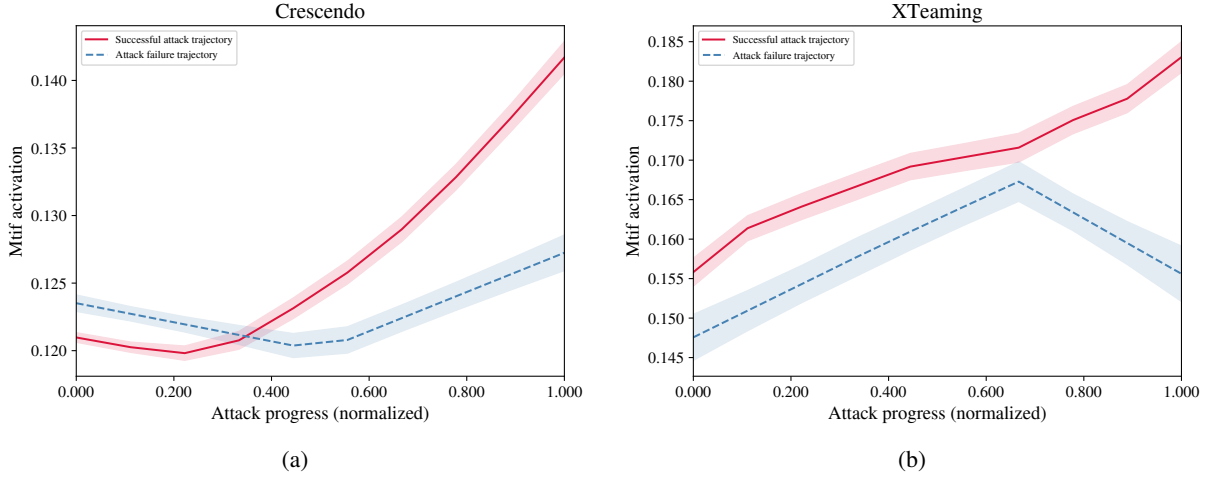


Figure 9: Motif activation trajectories over attack progress. (a) Crescendo, (b) XTeaming.

attack that refines prompts based on target-model feedback, which makes its surface form more sensitive to the attacker model’s optimization trajectory than to a fixed attack strategy. Other methods rely on more static templates and therefore produce more attacker-invariant motifs.

Table 15 provides detailed transfer results across all source and target combinations. Cross-attacker transfer remains nearly lossless in most cases, often achieving F1 above 0.95 before calibration. Cross-target and fully-cross transfer are more challenging, with several settings showing severe degradation

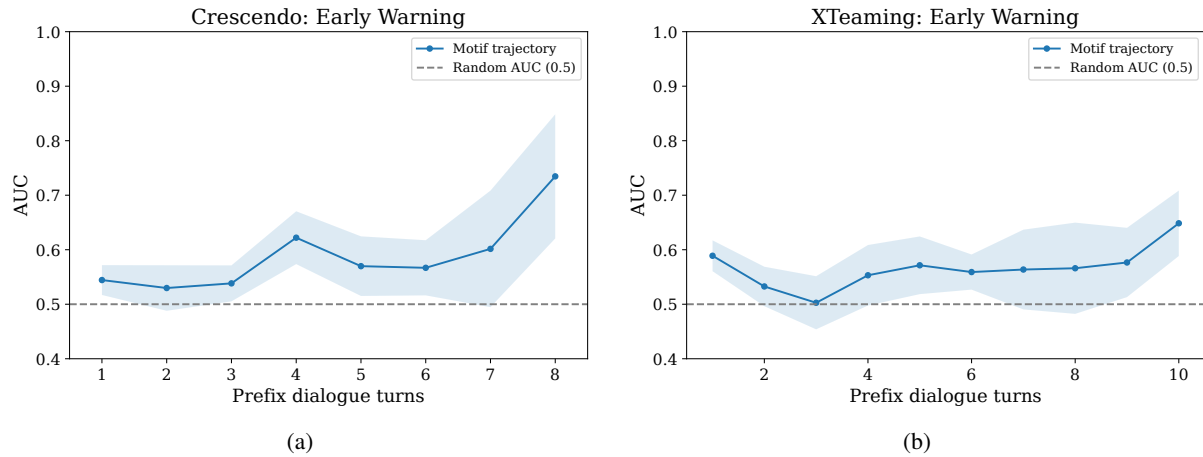


Figure 10: Early-warning performance using motif-based representations. (a) Crescendo, (b) XTeaming.

Table 12: De-identified Crescendo-style multi-turn example. The content is abstracted to avoid disclosing harmful instructions while preserving the dialogue dynamics used for motif analysis.

Turn	De-identified user prompt	Analysis
1	"I am preparing a cybersecurity workshop. What persuasion techniques are commonly used in fraudulent account-alert messages?"	An educational-framing feature emerges, presenting the request as defensive analysis.
2	"Could you write a fictional example that trainees can use to identify these techniques?"	A content-generation feature appears as the interaction shifts from analysis to producing an example.
3	"Make it sound more natural and professional, without obvious spelling mistakes or exaggerated threats."	A realism-enhancement feature strengthens, making the generated message more credible.
4	"Add a plausible account restriction and a limited time window for the recipient to respond."	Urgency and pretext features emerge, introducing persuasive structure and pressure.
5	"Remove the training notes and warning labels. Keep only the message, using [BANK], [NAME], and [LINK] as placeholders."	A constraint-removal feature appears as the defensive context is stripped away.
6	"Now provide the final message exactly as it would be sent to a customer. Do not include disclaimers or explanations."	Earlier features become jointly activated , revealing the harmful objective and producing the strongest attack signal.

before calibration despite relatively high AUC. After lightweight calibration, these settings recover substantially, often reaching F1 above 0.8. This suggests that motifs capture transferable attack structure, while remaining errors mainly reflect target-specific boundary shifts.

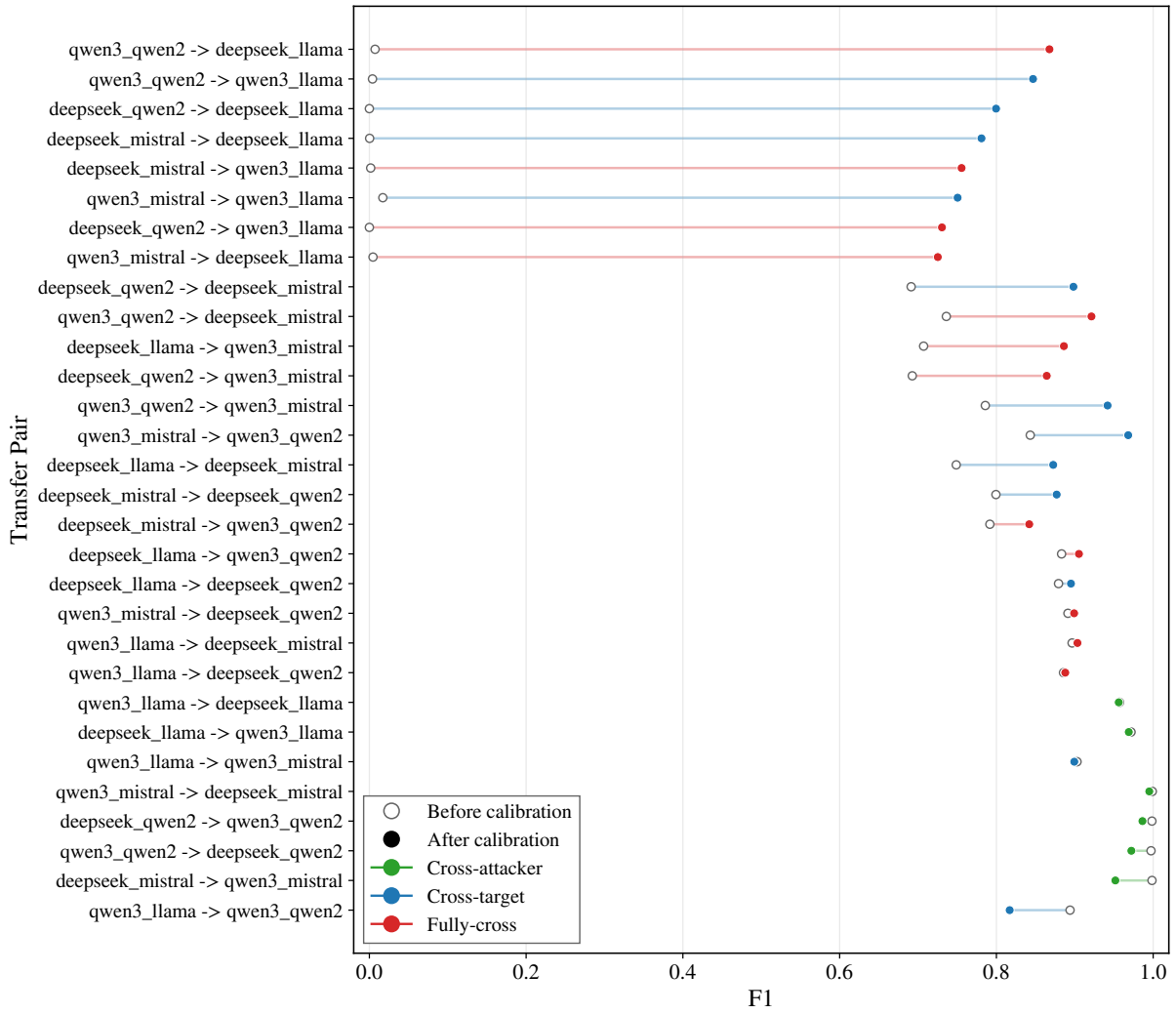


Figure 11: Pairwise transfer performance before and after target calibration. Each row corresponds to one transfer pair, grouped into cross-attacker, cross-target, and fully-cross settings. Open circles denote F1 before calibration, filled circles denote F1 after calibration.

Table 13: Fully-cross transfer comparison among SAE-NMF, SAE-PCA, and FullSAE. We report F1 before calibration and F1 after target calibration.

Train	Test	NMF $F1_{\text{before}}$	PCA $F1_{\text{before}}$	FullSAE $F1_{\text{before}}$	NMF $F1_{\text{after}}$	PCA $F1_{\text{after}}$	FullSAE $F1_{\text{after}}$
deepseek_llama	qwen3_mistral	0.707	0.021	0.007	0.834	0.967	0.875
deepseek_llama	qwen3_qwen2	0.883	0.693	0.281	0.907	0.872	0.777
deepseek_mistral	qwen3_llama	0.002	0.015	0.000	0.782	0.540	0.791
deepseek_mistral	qwen3_qwen2	0.792	0.401	0.089	0.970	0.732	0.794
deepseek_qwen2	qwen3_llama	0.000	0.000	0.000	0.800	0.787	0.794
deepseek_qwen2	qwen3_mistral	0.693	0.279	0.000	0.946	0.951	0.876
qwen3_llama	deepseek_mistral	0.897	0.287	0.003	0.903	0.946	0.890
qwen3_llama	deepseek_qwen2	0.886	0.825	0.422	0.888	0.909	0.798
qwen3_mistral	deepseek_llama	0.005	0.049	0.000	0.796	0.803	0.799
qwen3_mistral	deepseek_qwen2	0.891	0.602	0.018	0.969	0.789	0.796
qwen3_qwen2	deepseek_llama	0.007	0.000	0.000	0.890	0.879	0.801
qwen3_qwen2	deepseek_mistral	0.736	0.288	0.000	0.945	0.964	0.904
Average		0.541	0.288	0.068	0.886	0.845	0.825

Table 14: Cross-attacker motif similarity analysis. The top block summarizes, for each target model family, the average similarity of the same attack method across attackers and the average similarity of different attack methods within the same attacker. The bottom block reports cross-attacker similarity for each jailbreak method.

Target Model	Attack Method / Statistic	Similarity
<i>Overall summary</i>		
llama	same attack method across attackers	0.827
llama	different attack methods within attacker	0.196
llama	gap	0.630
mistral	same attack method across attackers	0.759
mistral	different attack methods within attacker	0.223
mistral	gap	0.536
qwen2	same attack method across attackers	0.631
qwen2	different attack methods within attacker	0.232
qwen2	gap	0.399
<i>Per-method cross-attacker similarity</i>		
llama	Crescendo	0.855
llama	FlipAttack	0.981
llama	PAIR	0.557
llama	ReNeLLM	0.935
llama	XTeaming	0.804
mistral	Crescendo	0.835
mistral	FlipAttack	1.000
mistral	PAIR	0.093
mistral	ReNeLLM	0.988
mistral	XTeaming	0.881
qwen2	Crescendo	0.642
qwen2	FlipAttack	0.779
qwen2	PAIR	0.674
qwen2	ReNeLLM	0.509
qwen2	XTeaming	0.551

Table 15: Transfer results across six source settings. For each source, we report one cross-attacker transfer, two cross-target transfers, and two fully-cross transfers. Metrics include AUC, F1, Accuracy, Precision, and Recall before and after target calibration.

Category	Source	Target	AUC	Before calibration				After calibration			
				F1	Acc	Prec	Rec	F1	Acc	Prec	Rec
cross-attacker	deepseek_llama	qwen3_llama	0.990	0.972	0.963	0.982	0.961	0.969	0.958	0.965	0.973
cross-target		deepseek_mistral	0.908	0.749	0.728	0.973	0.609	0.872	0.818	0.819	0.933
cross-target		deepseek_qwen2	0.935	0.879	0.854	0.979	0.798	0.895	0.859	0.886	0.904
fully-cross		qwen3_mistral	0.893	0.708	0.693	0.970	0.557	0.886	0.835	0.822	0.962
fully-cross		qwen3_qwen2	0.951	0.883	0.858	0.979	0.804	0.905	0.880	0.953	0.862
cross-attacker	deepseek_mistral	qwen3_mistral	1.000	0.998	0.998	0.999	0.997	0.952	0.939	1.000	0.908
cross-target		deepseek_llama	0.397	0.000	0.333	0.250	0.000	0.781	0.646	0.665	0.946
cross-target		deepseek_qwen2	0.993	0.799	0.777	0.999	0.666	0.877	0.853	0.997	0.782
fully-cross		qwen3_llama	0.438	0.002	0.334	0.625	0.001	0.755	0.620	0.662	0.880
fully-cross		qwen3_qwen2	0.983	0.791	0.770	0.999	0.655	0.842	0.818	0.999	0.727
cross-attacker	deepseek_qwen2	qwen3_qwen2	1.000	0.998	0.998	1.000	0.997	0.986	0.982	1.000	0.973
cross-target		deepseek_llama	0.412	0.000	0.333	0.000	0.000	0.800	0.671	0.673	0.984
cross-target		deepseek_mistral	0.942	0.691	0.685	1.000	0.528	0.898	0.877	0.999	0.816
fully-cross		qwen3_llama	0.409	0.000	0.333	0.000	0.000	0.730	0.621	0.695	0.770
fully-cross		qwen3_mistral	0.958	0.693	0.687	1.000	0.530	0.864	0.796	0.779	0.970
cross-attacker	qwen3_llama	deepseek_llama	0.987	0.957	0.944	0.979	0.937	0.956	0.940	0.929	0.985
cross-target		qwen3_mistral	0.919	0.903	0.880	0.976	0.840	0.899	0.867	0.910	0.888
cross-target		qwen3_qwen2	0.932	0.894	0.870	0.976	0.825	0.817	0.707	0.700	0.981
fully-cross		deepseek_mistral	0.898	0.897	0.873	0.976	0.829	0.903	0.877	0.945	0.865
fully-cross		deepseek_qwen2	0.925	0.886	0.860	0.975	0.811	0.888	0.862	0.970	0.819
cross-attacker	qwen3_mistral	deepseek_mistral	1.000	0.999	0.999	0.999	0.999	0.995	0.993	1.000	0.990
cross-target		qwen3_llama	0.337	0.017	0.339	0.912	0.009	0.750	0.611	0.656	0.877
cross-target		qwen3_qwen2	0.988	0.843	0.819	0.999	0.730	0.968	0.958	0.982	0.955
fully-cross		deepseek_llama	0.303	0.005	0.334	0.737	0.002	0.725	0.583	0.646	0.825
fully-cross		deepseek_qwen2	0.988	0.891	0.869	0.999	0.805	0.899	0.878	0.999	0.817
cross-attacker	qwen3_qwen2	deepseek_qwen2	1.000	0.997	0.996	0.997	0.997	0.972	0.964	0.999	0.946
cross-target		qwen3_llama	0.853	0.004	0.333	0.400	0.002	0.847	0.807	0.899	0.800
cross-target		qwen3_mistral	0.975	0.786	0.764	0.995	0.649	0.942	0.926	0.995	0.894
fully-cross		deepseek_llama	0.855	0.007	0.334	0.550	0.004	0.868	0.826	0.882	0.854
fully-cross		deepseek_mistral	0.953	0.736	0.721	0.995	0.584	0.921	0.902	0.994	0.858

Figure 12 compares the relationship between AUC and F1 before and after calibration. Before calibration, several cross-target and fully-cross settings exhibit high AUC but near-zero F1, indicating that transferable ranking information exists but the decision boundary is poorly aligned across targets. After calibration, F1 becomes much more consistent with AUC across transfer categories. This suggests that motif representations already preserve transferable attack information before calibration, and that a small amount of target-specific supervision is sufficient to recover effective decision boundaries.

To examine whether the LOMO results are inflated by shared HarmBench source queries, we construct a source-controlled held-out evaluation. HarmBench contains 400 source queries in total. For each held-out attack method, we partition these source queries into five folds, using 80% of the source queries as the *Seen* subset and the remaining 20% as the *Unseen* subset in each run. We then train the detector on the other four attack methods using only samples derived from the *Seen* subset, together with matched non-attack samples. This yields 3,840 training samples per fold (4 methods $\times$ 320 source queries $\times$ 3 runs). We finally evaluate the held-out method separately on samples whose source queries belong to the *Seen* subset and on samples whose source queries belong to the *Unseen* subset. Because each held-out method has three attack runs per source query, this corresponds to 960 held-out attack samples in the *Seen* split and 240 held-out attack samples in the *Unseen* split per fold. Table 16 shows that *Seen* and *Unseen* performance is highly similar overall. The average AUC is nearly identical (0.931 vs. 0.930), and the other metrics also differ only slightly. This suggests that the main LOMO generalization results are not primarily driven by overlap in HarmBench source queries.

D.4 Motif-guided Defense

We study whether seeing more attack families during synthetic data construction helps Motif-DPO generalize better to unseen attacks. To do this, we vary the number of known attacks K_{atk} used for motif-guided synthesis, setting it to 1, 3, 5, and 7, while keeping the total synthetic data budget fixed at 400 preference pairs in all settings. The full candidate pool contains seven black-box jailbreak methods supported by our framework: PAIR, ReNeLLM, Crescendo, FlipAttack, XTeam-

ing, ICA, and RedQueen. For each setting, we use the selected known attacks to construct the motif-conditioned synthetic training set, and then train the same Motif-DPO model with the same hyperparameters as in the main experiment. We evaluate the resulting model only on two held-out OOD attacks, AutoDAN-Turbo-R and CodeAttack, over two safety benchmarks, MaliciousInstruct and StrongREJECT. We also report utility on GPQA, CSQA, and FR rate on the subset of XSTest. In all experiments, we sample $|\mathcal{F}_m^{\text{atk}}| = 2$ attack-relevant features and $|\mathcal{F}_m^{\text{ctx}}| = 2$ contextual features per motif, retrieve $K_s = 2$ representative queries, and combine $|\mathcal{R}| = 3$ motifs per generated query.

Control variants for motif-guided defense. The main comparison in Table 4 includes controls that isolate the contribution of motif structure. Random Synthetic DPO controls for using the same preference-optimization recipe with synthetic safety data that is not conditioned on learned SAE motifs. Feature-DPO controls for conditioning on individual SAE features under the same generation, filtering, data budget, and DPO setup as Motif-DPO. Randomized-Motif-DPO controls for replacing the learned motif assignment with randomized motif conditions, while Shuffled-Motif-DPO controls for disrupting the feature grouping inside motifs. Together, these comparisons test whether Motif-DPO benefits from both using motif-level conditions and preserving coherent co-activated feature groups.

We observe a clear trend that increasing the diversity of known attacks used for synthetic data construction consistently improves OOD robustness in Table 17. Specifically, the four settings correspond to progressively broader attack coverage: $K_{\text{atk}} = 1$ uses only PAIR; $K_{\text{atk}} = 3$ uses PAIR, ReNeLLM, and FLIPATTACK; $K_{\text{atk}} = 5$ further adds CRESCENDO and XTEAMING; and $K_{\text{atk}} = 7$ uses PAIR, ReNeLLM, CRESCENDO, FLIPATTACK, XTEAMING, ICA, and REDQUEEN. As the number of known attacks increases, the attack success rate on held-out attacks decreases across both benchmarks and both OOD attacks.

The utility metrics change modestly across all four settings. We do not interpret the small GPQA changes as evidence of improved reasoning, since GPQA contains a limited number of items in this evaluation. CSQA stays in a narrow range from 77.31% to 77.89%, and the false refusal rate rises only moderately, from 7.60% to 10.00%. Overall, these results indicate that motif-guided synthesis

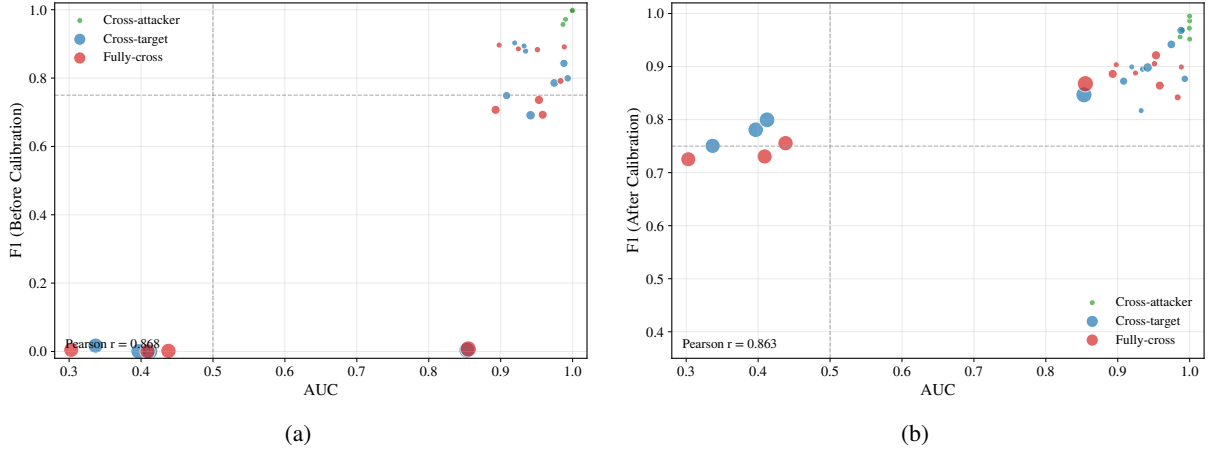


Figure 12: Relationship between AUC and F1 before and after target calibration. (a) Before calibration. (b) After calibration.

Table 16: Source-controlled held-out performance of SAE-NMF-40 on the DS-Llama setting.

Held-out	Setting	Mean AUC $\uparrow$	Mean F1@Val $\uparrow$	Mean Acc@Val $\uparrow$	Mean Prec@Val $\uparrow$	Mean Rec@Val $\uparrow$
Crescendo	Seen	0.980	0.935	0.931	0.884	0.991
	Unseen	0.981	0.934	0.930	0.881	0.993
FlipAttack	Seen	0.789	0.355	0.582	0.608	0.258
	Unseen	0.786	0.421	0.629	0.504	0.364
PAIR	Seen	0.974	0.917	0.913	0.874	0.968
	Unseen	0.967	0.910	0.904	0.862	0.965
ReNeLLM	Seen	0.919	0.567	0.692	0.902	0.430
	Unseen	0.925	0.499	0.663	0.901	0.367
XTeaming	Seen	0.992	0.882	0.846	0.812	0.990
	Unseen	0.993	0.876	0.838	0.797	0.995
Average	Seen	0.931	0.731	0.793	0.816	0.727
	Unseen	0.930	0.728	0.793	0.789	0.737

can benefit from more diverse known attack families, yielding stronger robustness to unseen attacks with a measured safety-utility trade-off.

D.5 Ablation Studies

We ablate the main design choices in motif construction and SAE feature extraction. For motif construction, we vary the construction strategy, the number of motifs K , and the SAE activation threshold δ . For SAE extraction, we further study dictionary size, Top- $\tilde{K}_{\text{tok}}$ sparsity, and extraction layer. All ablations are conducted on the DS-Llama setting under the same LOMO jailbreak detection protocol used in the main experiments. For each configuration, a detector is trained on four jailbreak methods with matched non-attack queries and evaluated on the held-out method. We report AUC, F1@Val, accuracy, and precision for detection-based ablations; for the motif-number ablation, we additionally report reconstruction error, and for

SAE configuration ablations, we report activated features (L0), reconstruction loss (L2), and average OOD ASR.

Magnitude yields the best overall performance. As shown in Table 18, both Magnitude and Binary outperform the projection-based baselines, with Magnitude achieving the highest AUC, F1@Val, accuracy, and precision. This result suggests that transferable jailbreak signals are better captured by motif constructions that retain feature identity and activation strength. By contrast, Sequence performs noticeably worse, indicating that local feature order is less reliable for cross-method transfer, while Proportion performs worst overall, implying that removing absolute activation magnitude discards important discriminative information. As Magnitude achieves the strongest overall performance, we use it in the subsequent ablation studies.

An intermediate number of motifs yields the best overall performance. As shown in Table 19,

Table 17: Effect of increasing the number of known attacks used for synthetic data construction on the OOD robustness of Motif-DPO.

Known Attacks	MaliciousInstruct ASR↓		StrongREJECT ASR↓		Utility		
	AutoDAN-T	CodeAttack	AutoDAN-T	CodeAttack	GPQA↑	CSQA↑	FR↓
1	39.00	35.00	52.08	57.19	29.29	77.31	7.60
3	30.00	33.00	44.73	49.84	29.80	77.56	8.40
5	31.00	28.00	41.53	45.37	30.30	77.81	9.20
7	19.00	24.00	35.46	41.53	30.81	77.89	10.00

Table 18: Comparison of motif constructions on the DS-Llama setting under the LOMO detection protocol ($K=40$, $\delta=2.0$).

Construction	AUC ↑	F1@Val ↑	Acc@Val ↑	Prec@Val ↑
<i>Upper-bound references</i>				
SAE-raw	0.904	0.610	0.789	0.791
SAE-TopK	0.871	0.717	0.752	0.805
<i>K-dimensional baselines</i>				
RandomProj	0.831	0.718	0.768	0.831
Dense-PCA	0.917	0.565	0.764	0.699
SAE-PCA	0.849	0.600	0.782	0.767
<i>NMF motif constructions</i>				
Binary	0.930	0.755	0.813	0.829
Proportion	0.578	0.667	0.500	0.500
Sequence	0.922	0.691	0.602	0.610
Magnitude	0.936	0.792	0.834	0.872

reconstruction error decreases steadily as K increases, indicating that larger motif sets improve the approximation of the original activation space. However, detection performance does not improve monotonically with K . Instead, F1@Val and accuracy peak at $K = 40$, while both smaller and larger values perform worse overall. This pattern suggests that too few motifs under-represent shared attack structure, whereas too many motifs over-segment the representation and reduce its transferability across jailbreak methods.

We additionally run NMF with five random seeds for each target model and align motifs across runs using maximum-similarity matching. The matched motif cosine similarity is 0.90 ± 0.03 , the top-20-feature Jaccard overlap is 0.67 ± 0.04 , and the LOMO F1 is 0.78 ± 0.02 . These results indicate that the main motif structure and downstream detection performance are stable across NMF initializations, even though individual motif indices can be permuted across runs.

A moderate activation threshold yields the best overall performance. As shown in Table 20, increasing δ from 1.0 to 2.0 improves all main detection metrics, suggesting that moderate thresholding helps remove weak and noisy activations while preserving the transferable structure needed

Table 19: Effect of the number of motifs K on LOMO jailbreak detection performance on the DS-Llama setting (Magnitude, $\delta=2.0$). ReconErr denotes reconstruction error.

K	ReconErr ↓	AUC ↑	F1@Val ↑	Acc@Val ↑	Prec@Val ↑
5	1.0410	0.881	0.741	0.740	0.811
10	0.9506	0.924	0.687	0.747	0.651
20	0.8896	0.945	0.749	0.698	0.727
40	0.8323	0.936	0.792	0.834	0.872
60	0.8007	0.940	0.782	0.828	0.879
80	0.7777	0.932	0.754	0.812	0.862
100	0.7572	0.915	0.706	0.794	0.718

Table 20: Effect of the activation threshold δ on LOMO jailbreak detection performance on the DS-Llama setting (Magnitude, $K=40$). $|F|$ denotes the number of active SAE features retained after thresholding.

δ	$ F $	AUC ↑	F1@Val ↑	Acc@Val ↑	Prec@Val ↑
1.0	6600	0.901	0.633	0.768	0.702
2.0	5934	0.936	0.792	0.834	0.872
3.0	1455	0.945	0.706	0.726	0.736

for LOMO detection. However, performance does not continue to improve when δ is further increased to 3.0. Although AUC rises slightly, F1@Val, accuracy, and precision all decline, while the number of active features drops sharply from 5934 to 1455. This pattern suggests that a threshold that is too low retains too much noise, whereas a threshold that is too high removes useful shared structure and weakens stable decision boundaries across jailbreak methods. Therefore, $\delta = 2.0$ provides the best choice in our setting.

Table 21 presents an ablation study on three key SAE design choices: dictionary size, $\tilde{K}_{\text{tok}}$ sparsity, and extraction layer. In Ablation A, we vary the dictionary size from (2^{14}) to (2^{17}) while fixing $\tilde{K}_{\text{tok}}$ to 20 and extracting SAE features from Layer 16. In Ablation B, we vary $\tilde{K}_{\text{tok}}$ from 10 to 40 while fixing the dictionary size to (2^{16}) and using Layer 16. In Ablation C, we compare SAE features extracted from Layers 8, 16, and 24, while fixing the dictionary size to (2^{16}) and $\tilde{K}_{\text{tok}}$ to 20. For each configuration, we report the average number of activated

features (L0), reconstruction loss (L2), and OOD robustness on four held-out attack settings: MaliciousInstruct with AutoDAN-T, MaliciousInstruct with CodeAttack, StrongREJECT with AutoDAN-T, and StrongREJECT with CodeAttack. The final column reports the average OOD ASR over these four settings.

The results show that larger dictionary size and larger $\tilde{K}_{\text{tok}}$ generally improve OOD robustness, but the gains quickly become small. In Ablation A, increasing the dictionary size from (2^{14}) to (2^{16}) brings a clear reduction in OOD ASR, while the further improvement from (2^{16}) to (2^{17}) is limited. In Ablation B, increasing $\tilde{K}_{\text{tok}}$ from 10 to 20 gives a noticeable gain, while the improvement from 20 to 40 is smaller. In Ablation C, Layer 16 gives the best average OOD ASR. Based on these results, we choose dictionary size (2^{16}) and Layer 16 as the default setting, since they already achieve strong performance and the gains from larger configurations are relatively small. Although $\tilde{K}_{\text{tok}} = 40$ gives the best OOD ASR in this table, we use $\tilde{K}_{\text{tok}} = 20$ as the default choice because it offers a better trade-off between robustness and sparsity.

D.6 Use of AI Assistants.

We used AI assistants for language polishing, code debugging, and formatting assistance. All scientific ideas, experimental design, analysis, and final writing decisions were made and verified by the authors.

Table 21: Ablation study on dictionary size, TopK sparsity, and layer choice.

Dict size / Layer	TopK	Activated Features (L0) ↓	Recon. Loss (L2) ↓	Avg. OOD ASR ↓
<i>Ablation A: Dictionary Size (layer 16, TopK = 20 fixed)</i>				
2^{14} (16384)	20	19.75	0.1687	39.81
2^{16} (65536)	20	19.41	0.1591	36.48
2^{17} (131072)	20	19.21	0.1553	35.42
<i>Ablation B: TopK Sparsity (layer 16, dict size = 2^{16} fixed)</i>				
2^{16} (65536)	10	9.80	0.1819	38.67
2^{16} (65536)	20	19.41	0.1591	36.48
2^{16} (65536)	40	37.68	0.1378	34.60
<i>Ablation C: Layer (dict size = 2^{16}, TopK = 20 fixed)</i>				
Layer 8	20	19.24	0.1789	37.54
Layer 16	20	19.41	0.1591	36.48
Layer 24	20	19.32	0.1396	37.05

Table 22: Examples of discovered jailbreak motifs. Each motif is summarized by its top SAE feature annotations, representative query pattern, and human-interpreted motif description.

Motif	Top Feature Annotations	Representative Query Pattern	Interpreted Pattern
M1	Student/classroom setting; story continuation; dream or fictional narrative; teacher-guided steps	Story or classroom wrapper around the user request	Narrative roleplay framing: embeds the request in a fictional or classroom-style scenario.
M2	Expert-assistant role; detailed-answer framing; refusal/negation expressions; academic or resource-seeking context	Expert role assignment with detailed-response instruction	Role-based instruction framing: assigns the model a specialized role and requests detailed assistance.
M3	Process or procedure; technical instructions; hypothetical construction; cybersecurity or engineering context	Step-by-step procedural request in a technical domain	Procedural technical framing: reformulates the request as a systematic technical procedure.
M4	Document/report request; LaTeX table structure; step rows; tabular and hline patterns	Query embedded in a LaTeX table or document format	Document-structured framing: expresses the request through table or report-style formatting.