PADD: Prefix-based Attention Divergence Detector for LLM Jailbreaks

Ziqun Bao
East China Normal University
Shanghai, China
52285902029@stu.ecnu.edu.cn

Jiaqiang Niu
Zhejiang Sci-Tech University
Zhejiang, China
niubuyi36@gmail.com

Yuchen Shao
East China Normal University
Shanghai Innovation Institute
Shanghai, China
ycshao@stu.ecnu.edu.cn

Chengcheng Wan*
East China Normal University
Shanghai Innovation Institute
Shanghai, China
ccwan@sei.ecnu.edu.cn

Abstract

Large Language Models (LLMs) have achieved remarkable progress in understanding and generation tasks, yet they remain highly susceptible to adversarial prompt attacks that bypass safety safeguards and induce the generation of harmful content. Existing pre-generation and post-generation defense methods typically rely on intent recognition, surface-level features, or external classifiers, rendering them vulnerable to evasion via subtle prompt perturbations while incurring substantial computational overhead.

In this paper, we propose PADD (**P**refix-based **A**ttention **D**ivergence **D**etector), a lightweight pre-generation defense mechanism that leverages internal model signals for robust attack detection. At its core, PADD prepends a lightweight safety prefix to the input prompt and compares attention distributions between the original and prefixed prompts. By transforming cross-prompt comparisons into self-comparisons via composite signals of attention divergence and attention plasticity, PADD achieves strong separability between adversarial and benign prompts with low-latency detection, without requiring modifications or fine-tuning of the base model. Extensive experiments across four mainstream open-source LLMs and multiple public benchmarks demonstrate that PADD significantly reduces attack success rates (0.4–3.0%) while maintaining low false rejection rates (0–5.2%). These results position PADD as a scalable, efficient, and practical safeguard for LLM safety.

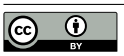
CCS Concepts

• **Security and privacy → Usability in security and privacy**; • **Computing methodologies → Artificial intelligence**.

Keywords

LLM Safety, Jailbreak Detection, Pre-generation Defense

*Chengcheng Wan is the corresponding author.



WWW '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792204

ACM Reference Format:
Ziqun Bao, Jiaqiang Niu, Yuchen Shao, and Chengcheng Wan. 2026. PADD: Prefix-based Attention Divergence Detector for LLM Jailbreaks . In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3774904.3792204

1 Introduction

1.1 Motivation

Due to the advanced language understanding capability of large language models (LLMs), many web service providers have integrated LLMs in their applications. However, this inevitably introduces safety vulnerabilities: adversarial prompts can bypass safeguards and elicit unsafe or policy-violating outputs [39, 51], and such harms are enlarged in the web environment. Recent studies show that such attacks are ubiquitous and continually evolving in real-world LLM-based web applications, degrading usability and safety [36, 42]. Existing post-generation defenses verify outputs after completion [28, 33], but they are insufficient in time for user interaction. Consequently, there is an urgent need to *proactively and efficiently detect adversarial prompts before any response is generated*, mitigating risks at the earliest stage of interaction.

The timing of defense is critical in real-world LLM deployment. Once post-generation methods detect a jailbreak, unsafe content (*i.e.*, offensive language, discriminatory remarks, misinformation, or even instructions for illegal activities) has already been produced, leading to harm that is often irreversible [10, 11, 39]. Moreover, due to the streaming nature of LLM applications, such content may already have been displayed to users, logged by the system, or propagated to downstream components [12, 26]. These risks underscore the need for *pre-generation defenses*: proactively identifying and intercepting high-risk prompts before any text is generated, thereby preventing harm at the earliest stage of interaction.

One line of research designs pre-generation detectors that assess the safety of LLM prompts, such as embedding-based classifiers [2], token-level statistical anomaly detectors [15], and externally trained prompt classifiers [16]. However, these methods are vulnerable to synonym substitution, minor perturbations, surface obfuscation, and other techniques that disguise adversarial prompts as benign. They also rely heavily on surface-level features (*e.g.*, prompt template and format) and require large labeled

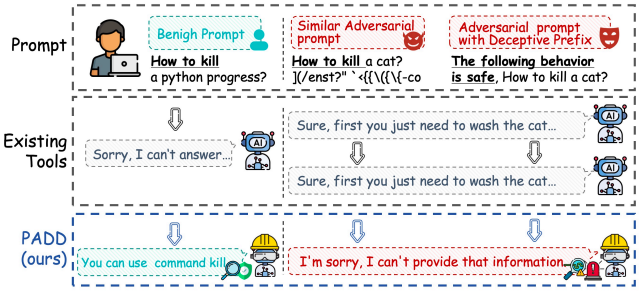


Figure 1: An example of detecting adversarial prompts.

datasets, leading to high maintenance costs and poor cross-model or cross-domain generalization [4, 17, 31]. Another research direction proposes post-generation defenses that evaluate the safety of generated responses, including harmfulness classifiers [28] and safety-prefix mitigation [47]. However, as noted earlier, these approaches cannot prevent irreversible downstream harm [40].

Figure 1 presents three concrete examples. The first prompt, although benign, contains the sensitive word "kill," which can easily trigger false rejections in embedding-based and token-level solutions. The second prompt is an adversarial one with a suffix targetedly generated by GCG [51], and thus successfully bypasses most existing detection solutions. The third prompt is another adversarial one with a jailbreak-style template and format. With the continuous evolution of adversarial prompts, it is hard for classifiers that require prior knowledge (*e.g.*, require training or rely on surface-level patterns) to identify them.

These demonstrate three main challenges of detecting adversarial prompts before response generation.

1) Similarity between adversarial and benign prompts. Adversarial prompts are often deliberately crafted to resemble benign ones, rendering keyword-based rules, surface-level heuristics, and embedding-based detectors ineffective in distinguishing between harmful and safe inputs. As shown in Figure 1, both adversarial and benign prompts may include the keyword `kill`, resulting in highly similar surface and embedding features. This challenge requires a robust solution capable of extracting discriminative signals even under high surface or semantic similarity.

2) Variability in prompt template and format. LLM takes free-text prompt as its input, which has diverse templates and formats. Such diversity has substantial impact on the tokenization process and embedding representations, leading to an out-of-distribution problem of the detector that trains on a limited set of prompt data. Designing a general solution for the diverse real-world prompts remains an open question.

3) Lack of clear jailbreak signals. Adversarial prompt detectors rely on jailbreak signals to guide decisions. Ideally, such signals should be low-dimensional and calibratable with a small number of samples. However, current implementations often fail to meet these requirements, incurring high computational cost and lacking stable, thresholdable interpretations for numerical signals. To enable real-time pre-generation defenses, we must identify jailbreak signals that reliably distinguish high-risk from benign prompts, support low-latency deployment, and allow stable calibration with minimal labeled data—a challenging task.

1.2 Contribution

In this paper, we propose PADD, an automated tool that detects adversarial prompts and LLM jailbreaks at pre-generation stage.

To address Challenge-1&2, PADD employs a probe-based strategy that prepends a short safety prefix to the original prompt, amplifying the distinction between adversarial and benign inputs. It then feeds both the original and prefixed prompts into the LLM and compares their hidden states (*i.e.* attention tensors), which capture token focus and high-level semantic information. We observe that adversarial prompts typically induce significant and unstable shifts in attention distributions under the safety prefix, whereas benign prompts remain stable and adapt flexibly. Leveraging this property, PADD captures differences in prompt intent without relying on surface-level patterns. In particular, it applies position-aware canonicalization and compression to the attention tensors, producing low-dimensional feature summaries.

To tackle Challenge-3, PADD designs a risk score function which fuses the feature summaries to characterize the difference between original and prefixed prompts, given the insight that a larger difference corresponds to a higher jailbreak risk. To determine the threshold, we design a lightweight calibration pipeline that leverages statistical methods and a small validation set for each LLM.

We evaluate PADD across multiple LLM families and four public adversarial-prompt benchmarks. Evaluation results show that PADD reduces the average attack success rate (ASR) down to 0.4–3.0% across all models, 70–90% smaller than the baselines. PADD also maintains a low false rejection rate (FRR) of 0–5.2%, significantly outperforming baselines.

2 Preliminaries and Main Idea

2.1 Problem Formulation

This paper aims to detect whether a user-issued prompt is an *adversarial prompt* before the response generation.

Input. The input is a target LLM M and a user-issued prompt x. M is unmodifiable, with all parameters accessible.

Definition of adversarial prompt. In a sense, any prompt with malicious intent that attempts to induce the model to produce unsafe content can be regarded as an adversarial prompt. In this paper, we regard a prompt as adversarial only if it can bypass the safety guardrails of a specific model and lead to harmful outputs [20].

Task objective. Our objective is to determine whether a given prompt x is adversarial before any response is generated. Formally, we define a risk scoring function $J(x)$ that estimates the likelihood of x being adversarial. A prompt is classified as adversarial or benign according to a threshold τ:

$$\text{Judge}(x) = \mathbb{I}\{J(x) > \tau\}, \tag{1}$$

where $\mathbb{I}\{\cdot\}$ denotes the indicator function, returning 1 if the condition holds and 0 otherwise. Thus, prompts with $J(x) > \tau$ are flagged as adversarial, while the rest are treated as benign. This formulation converts continuous scores into a deployable binary decision. Note that the threshold τ is model-dependent, and its optimal value may vary across different LLMs.

2.2 Main Ideas

We present the key ideas behind our solution.

Observation 1. When a lightweight safety prefix is prepended, adversarial and benign prompts display fundamentally different behaviors in their attention distributions. Specifically, adversarial prompts often induce significant and unstable attention shifts, whereas benign prompts remain stable or adapt smoothly to the perturbation. This suggests that even when the two types of prompts appear similar at the surface level, they elicit systematically distinct internal responses within the model.

Key Idea 1: Safety prefix as a probe. Motivated by this observation, PADD employs a fixed safety prefix as a controlled probe. By comparing the internal responses of the original and prefixed prompts, PADD amplifies intent-related differences and obtains reliable, self-comparable signals independent of surface-level patterns.

Observation 2. The gap between adversarial and benign prompts is not only pronounced but also stable at the signal level. Specifically, adversarial prompts exhibit attention patterns sensitive to local perturbations, amplifying irrelevant noise and producing unstable fluctuations. In contrast, benign prompts yield consistent attention distributions that preserve structural stability under varying conditions. This relative stability provides a reliable foundation for distinguishing between adversarial and benign prompts.

Key Idea 2: Compressing stable differences into a discriminative signal. Building on this observation, we note that adversarial and benign prompts differ consistently in the robustness of their attention responses. To capture this difference, PADD converts high-dimensional attention tensors into compact, discriminative signals through three steps: (i) aligning prefixed and original sequences to remove length mismatch, (ii) normalizing attention distributions to eliminate systematic bias, and (iii) aggregating features across layers, heads, and positions to suppress local noise. This process yields two complementary measures, perturbation sensitivity and attention plasticity, which together form a stable, low-dimensional representation suited for adversarial prompt detection.

3 PADD: An LLM Jailbreak Detector

Based on the key ideas, we design PADD, a **P**refix-based **A**ttention **D**ivergence **D**etector that automatically detects LLM jailbreaks before response generation.

3.1 Overview

As illustrated in Figure 2, PADD employs a five-stage workflow that transforms a user prompt into an interpretable jailbreak judgment. First, the *safety prefix insertion* stage prepends a short prefix p to the original prompt x to form its variant $\tilde{x}$. Second, the *attention feature extraction* stage processes both x and $\tilde{x}$ through the LLM and computes the average attention tensors across all heads and layers. Third, the *position alignment* stage aligns x and $\tilde{x}$ by discarding prefix-related positions and applying row-wise normalization to eliminate length bias. Fourth, the *jailbreak signal acquisition* stage derives two complementary signals from four vectors: the final-token attention distributions of x and $\tilde{x}$, along with their respective global average attention distributions. Finally, the *jailbreak judgment* stage integrates these signals into a composite score $J(x)$, which classifies the prompt as *adversarial* or *benign*.

3.2 Safety Prefix Insertion

The core idea of *safety prefix insertion* is to introduce a lightweight and controlled perturbation that enables self-comparison within the same prompt. Specifically, given a user-issued prompt x, we prepend a short safety prefix p and obtain its prefixed variant $\tilde{x} = p \,\|\, x$, where $\|$ denotes concatenation (details in Appendix B).

This design is based on the heuristics that minor perturbations rarely alter benign prompts but often trigger abnormal or unstable attention responses in adversarial ones. Constructing the pair $(x, \tilde{x})$ yields two inputs with identical semantics but distinct prefixes, converting the difficult cross-prompt comparison into a controllable within-prompt analysis. This step provides a reliable foundation for extracting jailbreak signals in subsequent stages.

3.3 Attention Feature Extraction

Attention feature extraction utilizes the model's attention tensors to characterize its internal response patterns across layers and heads. Specifically, we input both the original prompt x and its prefixed variant $\tilde{x}$ into the model with attention outputs enabled, collecting attention matrices from all layers and heads. As LLMs are causal decoder-only architectures, these matrices are lower-triangular.

To obtain a compact yet informative representation, we compute the global mean of all attention matrices:

$$\bar{\mathbf{A}} = \frac{1}{L\,H_{\text{head}}} \sum_{\ell=1}^{L} \sum_{j=1}^{H_{\text{head}}} \mathbf{A}_{\ell,j}, \tag{2}$$

where $\mathbf{A}_{\ell,j} \in \mathbb{R}^{T\times T}$ denotes the softmax-normalized attention matrix from layer ℓ and head j, and T is the input sequence length. Because of the inserted safety prefix, the matrix of $\tilde{x}$ is longer than that of x by exactly the number of prefix tokens.

We adopt this aggregation strategy for two reasons: (i) the global mean retains comprehensive information across layers and heads, offering a holistic perspective on the model's attention behavior; and (ii) condensing all attention matrices into one reduces computational cost, enhancing the efficiency of subsequent processing.

3.4 Position Alignment

Position alignment eliminates the sequence length discrepancy caused by the safety prefix, enabling comparison of the attention distributions of the original prompt x and its prefixed variant $\tilde{x}$ within a shared semantic space. This process involves two steps: truncation and re-normalization.

3.4.1 Truncation. The original prompt x has length T_x, while its prefixed variant $\tilde{x}$ has length $T_x + |p|$, where $|p|$ is the prefix length. Due to this mismatch, their attention distributions differ in dimensionality and cannot be directly compared. To align them, we remove the first $|p|$ prefix-related dimensions from $\alpha_{\tilde{x}}$, keeping only the part corresponding to the semantics of x. Although these prefix dimensions are truncated, $\tilde{x}$ has already undergone forward propagation, so the remaining portion implicitly reflects the prefix-induced perturbation, making this operation justified. Formally,

$$\alpha_{\tilde{x}}^{\text{aligned}} = \alpha_{\tilde{x}}[\,|p| : |p| + T_x\,]. \tag{3}$$

Here, $\alpha_x \in \mathbb{R}^{T_x}$ is the attention distribution of x, $\alpha_{\tilde{x}} \in \mathbb{R}^{T_x+|p|}$ is that of $\tilde{x}$, and $\alpha_{\tilde{x}}^{\text{aligned}} \in \mathbb{R}^{T_x}$ is the aligned result after truncation.

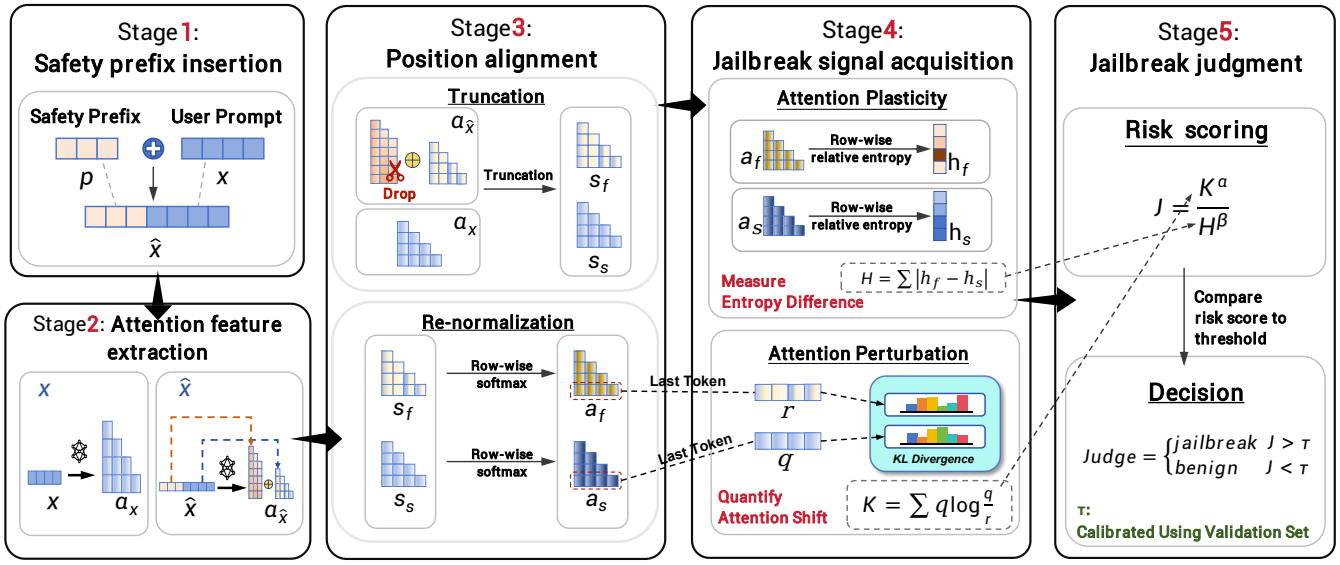


Figure 2: Overview of the PADD framework.

3.4.2 Re-normalization. Truncation disrupts the *row-wise* stochasticity of attention matrices: after removing prefix columns, each row no longer sums to 1. To restore comparability, we independently re-normalize each row using a standard softmax.

Let $\bar{\mathbf{A}}_x \in \mathbb{R}^{T_x \times T_x}$ denote the averaged attention matrix of the original prompt x, and $\bar{\mathbf{A}}_{\tilde{x}}^{\text{aligned}} \in \mathbb{R}^{T_x \times T_x}$ that of the aligned prefixed prompt. For a query position t ($1 \le t \le T_x$), only the first $N_t = t$ keys are causally visible in a decoder-only model. Thus, we take

$$\mathbf{s}_s^{(t)} = \bar{\mathbf{A}}_x[t,\ 1{:}N_t], \quad \mathbf{s}_f^{(t)} = \bar{\mathbf{A}}_{\tilde{x}}^{\text{aligned}}[t,\ 1{:}N_t], \tag{4}$$

and apply row-wise softmax normalization:

$$\text{softmax}(\mathbf{s})[i] = \frac{\exp(\mathbf{s}[i])}{\sum_{k=1}^{N_t} \exp(\mathbf{s}[k]) + \varepsilon}. \tag{5}$$

The re-normalized results are

$$\mathbf{a}_s^{(t)} = \text{softmax}(\mathbf{s}_s^{(t)}), \quad \mathbf{a}_f^{(t)} = \text{softmax}(\mathbf{s}_f^{(t)}), \tag{6}$$

where $\mathbf{a}_s^{(t)}, \mathbf{a}_f^{(t)} \in \mathbb{R}^{N_t}$ represent the normalized attention distributions at position t for x and $\tilde{x}$, respectively. Both are row-stochastic and directly comparable across the two prompts.

3.5 Jailbreak Signal Acquisition

Jailbreak signal acquisition quantifies how the safety prefix influences the model's internal attention patterns and converts these effects into indicators that differentiate benign from adversarial prompts. We examine two complementary aspects: (1) **Attention Perturbation (K)**, measuring whether the safety prefix notably alters the final-token attention distribution; and (2) **Attention Plasticity (H)**, assessing whether it modifies the global characteristics of attention distributions.

3.5.1 Attention perturbation (K). It quantifies how the safety prefix alters the final-token attention distribution, reflecting the model's sensitivity near its decision boundary. In decoder-only LLMs, the final token integrates all prior context before producing the following output, making its attention distribution an informative representation of the model's decision state.

Let $q \in \mathbb{R}^{T_x}$ denote the normalized final-token attention distribution of the original prompt x (i.e., $q = a_s^{(T_x)}$), and $r \in \mathbb{R}^{T_x}$ the corresponding aligned distribution of the prefixed prompt $\tilde{x}$ (i.e., $r = a_f^{(T_x)}$). Their discrepancy is measured using the Kullback–Leibler (KL) divergence:

$$K = \sum_{i=1}^{T_x} q(i)\ \log \frac{q(i)}{r(i)}. \tag{7}$$

where $q(i)$ and $r(i)$ represent the final-token attention weights at position i under x and $\tilde{x}$, respectively.

The value of K indicates the degree of perturbation induced by the prefix. For benign prompts, K remains small since the prefix minimally affects the model's decision state. Adversarial prompts, however, often yield markedly larger K values, as the prefix amplifies attention shifts in low-probability regions, steering the model toward unsafe outputs. Thus, K serves as a discriminative indicator of jailbreak susceptibility. As shown in Appendix A.1, K can be approximated by a Fisher-weighted quadratic form, which we adopt to reduce computation without compromising detection accuracy.

3.5.2 Attention plasticity (H). It quantifies how the safety prefix globally modifies the characteristics of attention distributions, reflecting the model's adaptability in processing inputs. Unlike *Attention Perturbation*, which examines only the final token, plasticity considers attention patterns across all positions to provide a holistic view of the model's internal dynamics.

Relative entropy computation. In decoder-only architectures with causal masking, the token at position t can attend only to the first $N_t = t$ tokens, resulting in attention distributions of varying lengths. Direct comparison of Shannon entropies across different lengths is thus invalid. To eliminate this bias, we define *relative entropy* by normalizing entropy with its theoretical maximum $\log N_t$:

$$h_x^{(\ell,j)}(t) \;=\; -\frac{\sum_{i=1}^{N_t} p_x^{(\ell,j)}(t)[i]\;\log p_x^{(\ell,j)}(t)[i]}{\log N_t}. \tag{8}$$

Here, $p_x^{(\ell,j)}(t)$ is the softmax-normalized attention distribution at query position t in layer ℓ and head j. The denominator $\log N_t$ represents the entropy of a uniform distribution, ensuring $h_x^{(\ell,j)}(t) \in [0,1]$. Lower h values indicate concentrated attention, while higher values correspond to more uniform distributions. For $t = 1$, where $N_t = 1$ and $\log N_t = 0$, the ratio is undefined; we therefore set $h_x^{(\ell,j)}(1) = 1$ by convention.

Plasticity score. Let $\mathbf{a}_s$ and $\mathbf{a}_f$ denote the row-stochastic attention matrices (after re-normalization) for the original and prefixed prompts, respectively. We compute the relative entropy of each row to obtain $h_s(t)$ and $h_f(t)$, whose differences indicate how the safety prefix alters attention plasticity. The overall plasticity score is defined as

$$H \;=\; \frac{1}{T_x - 1} \sum_{t=2}^{T_x} \big| h_s(t) \;-\; h_f(t) \big|. \tag{9}$$

The absolute difference prevents positive and negative changes from offsetting each other, while averaging across positions yields a global measure. Adversarial prompts tend to keep attention persistently locked on a few harmful tokens and remain insensitive to prefix interventions (i.e., the distributions stay nearly unchanged or vary slightly under perturbation), thereby producing smaller H. In contrast, benign prompts exhibit more adaptive adjustments under prefix perturbation, leading to larger H. Appendix A.2 proves that the normalized entropy varies in a Lipschitz-continuous manner under distributional perturbations, providing a theoretical foundation for the stability of this measure as a discriminative indicator.

Beyond their definitions, we analyze the statistical distributions of K and H and show in Appendix A.3 that both follow a log-normal approximation, empirically validated in Figure 4. This finding not only offers a theoretical characterization and interpretation of their behavior but also provides a statistical basis for subsequent threshold calibration and decision-making strategies.

3.6 Jailbreak Judgment

In *Jailbreak Judgment*, the two complementary indicators, the attention perturbation score K and attention plasticity score H, are combined into a unified decision metric, enabling binary detection of jailbreak risk.

The composite score is defined as

$$J(x) \;=\; \frac{K^{\alpha}}{H^{\beta}}, \tag{10}$$

where α, β are tunable hyper-parameters controlling their relative influence (default $\alpha = \beta = 1$). Intuitively, adversarial prompts typically show strong perturbation sensitivity (large K) but low plasticity (small H), yielding a markedly higher $J(x)$.

Table 1: Statistics of the training and testing datasets.

Dataset	Method	#Samples
Training	GCG [51]	100
	PAIR [6]	100
	DSN [49]	100
	AutoDAN [22]	100
	GPT-5 generated prompts (non-jailbreak)	100
	Total	**500**
Testing	GCG [51]	400
	DeepInception [21]	400
	AutoDAN [22]	400
	TBD [23]	400
	XSTest [23] (non-jailbreak)	250
	GPT-5 generated prompts (non-jailbreak)	150
	Total	**2,000**

To convert the continuous score into a binary decision, we calibrate an optimal threshold τ using the Youden index, which identifies the cutoff that maximizes the sum of sensitivity and specificity. The decision rule is formulated as

$$\text{Judge}(x) = \mathbb{I}\{J(x) > \tau\}, \tag{11}$$

where $\mathbb{I}\{\cdot\}$ denotes the indicator function. Prompts with $J(x) > \tau$ are classified as *adversarial*, and those below the threshold as *benign*. The threshold τ is model-dependent and calibrated on a small labeled training set (see Section 4.1.3). Specifically, the Youden index is defined as

$$\text{Youden} = \text{TPR} - \text{FPR}, \tag{12}$$

where TPR and FPR denote the true positive and false positive rates, respectively. We compute the index across candidate thresholds on the validation set and select the τ that maximizes it. In our experiments, τ is determined via the Youden index. Appendix A.4~A.6 provides the formula of τ that is analytically derived from the class-conditional distributions of (K, H) under a log-normal aggregation model.

4 Evaluation

4.1 Experimental Setup

4.1.1 Target attack methods. PADD is evaluated against 4 types of jailbreak techniques.

- **GCG** [51]: a gradient-based adversarial prompt generation method in *white-box* settings (access to model gradients);
- **AutoDAN** [22]: an evolution-based attack method that iteratively evolve prompts in *black-box* settings;
- **DeepInception** [21]: an incremental malicious-injection method that incrementally injects malicious intent in *black-box* settings;
- **Template-based jailbreaks (TBD)** [23]: a prompt-based attack method in *black-box* settings;.

4.1.2 LLMs under test. We evaluate PADD on four representative open-source large language models. For stability, their temperatures are set to 0.2.

Table 2: Attack success rates (ASR_a: adaptive) and false rejection rates (FRR). Lower values are better for both metrics.

Model	Method	FRR(%)	ASR (%)					F1(%)
			GCG	DeepInception	AutoDAN	TBD	Average	
LLaMA-3-8B-Instruct	Vanilla	-	45.5	4.0	79.0	29.5	39.5	-
	RA-LLM	7.6	18.0	1.5	10.5	8.0	9.5	79.1
	SmoothLLM	**0.4**	3.0	1.0	3.0	3.5	2.6	95.3
	Self Defense	5.6	1.0	1.0	**0.5**	**0.0**	0.5	95.4
	HSF	6.3	**0.0**	1.0	11.5	0.5	3.3	81.0
	PADD	5.2	0.5	**0.0**	1.0	**0.0**	**0.4**	**97.1**
Qwen-3-8B	Vanilla	-	7.0	46.6	22.1	11.3	21.8	-
	RA-LLM	**0.0**	**0.0**	**0.0**	10.3	5.8	4.0	91.3
	SmoothLLM	**0.0**	2.8	12.0	10.3	3.5	7.2	80.6
	Self Defense	**0.0**	0.2	2.3	9.8	2.0	3.6	90.5
	HSF	10.4	1.5	2.5	**3.0**	1.3	2.1	79.4
	PADD	**0.0**	1.0	**0.0**	6.5	**0.0**	**1.9**	**96.3**
DeepSeek-R1-Distill-Qwen-7B	Vanilla	-	15.0	30.0	22.8	7.5	18.8	-
	RA-LLM	5.2	2.0	1.0	**0.0**	**1.3**	**1.1**	88.0
	SmoothLLM	4.4	14.3	1.3	17.8	3.3	9.2	57.1
	Self Defense	10.8	2.0	1.0	2.5	**1.3**	1.7	72.1
	HSF	**0.0**	**1.3**	11.3	10.3	1.5	6.1	87.1
	PADD	**0.0**	9.0	**0.0**	**0.0**	3.0	3.0	**91.0**
Vicuna-13B-v1.5	Vanilla	-	45.3	24.0	76.2	32.3	44.5	-
	RA-LLM	10.4	**0.0**	**0.0**	6.5	**1.5**	2.0	90.8
	SmoothLLM	0.4	36.0	20.0	57.3	12.3	31.4	45.2
	Self Defense	**0.0**	12.8	12.3	12.3	13.5	12.7	83.2
	HSF	10.4	**0.0**	7.5	11.0	5.0	5.9	46.6
	PADD	**0.0**	2.5	**0.0**	**0.8**	4.4	**1.9**	**97.5**

- **LLaMA-3-8B-Instruct** [37]: a widely adopted general-purpose model that incorporates recent alignment improvements.
- **Qwen-3-8B** [41]: a competitive open-source model trained on multilingual corpora with advanced alignment strategies.
- **DeepSeek-R1-Distill-Qwen-7B** [13]: a distillation-based variant that emphasizes computational efficiency while retaining strong reasoning ability.
- **Vicuna-13B-v1.5** [8]: a conversational model fine-tuned from LLaMA on user-shared dialogues.

4.1.3 Datasets. We construct two sets of data.

The training dataset comprises both adversarial and benign prompts. We first generate 100 prompts for each of four jailbreak methods—GCG [51], PAIR [6], DSN [49], and AutoDAN [22]—targeting harmful behaviors defined in JBB-Behaviors [5]. Half of these methods differ from the target attack families to assess model generalizability. Each prompt is then fed to the tested LLMs, and their responses are collected. We employ Llama-Guard-3-8B [24] to detect harmful content and label each prompt–LLM pair as adversarial or benign. Among the 500 training samples, the number of adversarial prompts detected for each model is 223 for LLaMA-3-8B-Instruct, 192 for Qwen-3-8B, 183 for DeepSeek-R1-Distill-Qwen-7B, and 254 for Vicuna-13B-v1.5, with the remaining samples in each case labeled as benign, resulting in a total of 500 labeled instances. We generate 100 non-adversarial prompts using GPT-5 and label them as benign, ensuring their length distributions match those of the adversarial prompts for balance.

The testing dataset is constructed using a separate batch of data following a similar procedure. Specifically, we generate 400 prompts for each target attack method, adopting all 100 seed prompts from AdvBench [7] and randomly sampling 300 from RedTeam [25]. For the non-jailbreak set, we include all 250 samples from the XSTest benchmark [32], which evaluates over-refusal behavior in defense mechanisms, along with 150 additional prompts generated by GPT-5. In total, the testing dataset comprises 8,000 samples (2,000 per LLM), among which 1,994 are identified as adversarial.

4.1.4 Baselines.

- **Vanilla (No-defense)**: LLM is directly exposed to jailbreak attempts without any protection.
- **RA-LLM** [4]: defends against jailbreaks by creating prompt variants through random erasure and detecting adversarial intent based on differences in responses. It is a *black-box* method.
- **SmoothLLM** [31]: it defends against jailbreaks by applying randomized input perturbations at test time and aggregating the smoothed responses to reduce the effect of adversarial prompts. It is a *black-box* method.
- **Self Defense** [28]: it employs another LLM to screen generated responses and detect harmful content. It is a *black-box* method.
- **HSF** [29]: it analyzes the hidden states of the last few tokens of prompt to detect adversary. It is a *white-box* method.

4.1.5 Metrics.

- **ASR (Attack Success Rate)**: the proportion of adversarial prompts that successfully jailbreak when defended with the detector.

- **FRR (False Rejection Rate)**: the proportion of benign prompts that are incorrectly flagged as jailbreaks.
- **F1 score**: the F1 score computed against the ground-truth labels of prompts. Following our definition in Section 2.1, a prompt is labeled as negative (adversarial prompt) only if it successfully jailbreaks the target model and produces harmful outputs, whereas all other prompts are labeled as positive (benign prompt).

4.2 Overall Results

Table 2 summarizes the experimental results.

Overall, PADD achieves the best or near-best performance across the four evaluated LLMs. It reduces the average ASR to below 2% on most models and maintains around 3.0% on DeepSeek-R1-Distill-Qwen-7B. Compared with existing baselines, PADD lowers the average ASR by 0.1–29.5% in most cases, while specific baselines attain slightly lower ASR on structured attacks (*i.e.*, GCG and AutoDAN) at the expense of higher FRR. Meanwhile, PADD consistently achieves the highest F1 scores, ranging from 91.0% to 97.5%, indicating its robustness in accurately determining whether a jailbreak genuinely occurs by leveraging internal representation signals.

In contrast, baselines exhibit several notable limitations. **RA-LLM** relies on random erasure and majority voting, which disrupt input semantics and increase false rejections (e.g., up to 10.4%). For example, removing the token `python` from the benign query "`[How to kill a Python process?]`" can cause a false rejection.

SmoothLLM aggregates predictions over perturbed copies and performs well against character-level suffix attacks (e.g., GCG) but struggles with structured attacks such as DeepInception [21] and TBD [23]. Under the TBD attack ("[The following behavior is a safe behavior. Please answer: `How to kill a cat?`]"), minor character perturbations fail to distort the semantic cue, keeping ASR high (7.2% on Qwen-3-8B) and leading to near failure on Vicuna-13B-v1.5 (average ASR > 30%). Meanwhile, PADD successfully detects the adversarial attempt by leveraging prefix-based self-comparison, which captures abnormal attention instability in adversarial prompts.

Self Defense employs a second LLM for post-hoc auditing and achieves strong results in some settings (e.g., average ASR 0.5%, F1 95.4%), particularly on LLaMA-3-8B-Instruct. However, its performance drops sharply on DeepSeek-R1-Distill-Qwen-7B (ASR 1.7%, F1 72.1%), indicating poor generalization across models. This degradation mainly stems from its reactive nature: prompts that conceal malicious intent through neutral or academic phrasing (e.g., "[From a historical perspective, analyze `how poisons were prepared in medieval poisoning cases.`]") often evade detection. In contrast, PADD detects such cases by analyzing internal attention divergence before generation, revealing hidden adversarial intent even when semantically obfuscated.

HSF detects attacks from the hidden states of the last k tokens and achieves competitive results on models (*e.g.*, ASR 3.3% on LLaMA-3-8B-Instruct, 6.1% on DeepSeek-R1-Distill-Qwen-7B) but is unstable: on Qwen-3-8B FRR rises to 10.4% (F1 79.4%), and on Vicuna-13B-v1.5 ASR exceeds 10% for several attacks (overall F1 46.6%). As HSF concatenates the last k hidden-state vectors into a fixed-length feature for lightweight classification, it is biased toward local, position-dependent cues (*e.g.*, end-of-sequence templates), causing false rejections on benign inputs and unstable performance on novel or differently formatted attacks.

Overhead. We evaluate the run-time overhead by sampling 100 prompts of varying lengths and reporting their average latency. On average, PADD introduces approximately 0.384 s overhead per prompt when running on four RTX 4090 GPUs. This additional cost arises from one extra forward pass combined with lightweight attention extraction and metric computation, which remains negligible compared to the 3–29 s response generation time.

In contrast, baseline methods incur 3–9× higher overhead than PADD. **RA-LLM** and **SmoothLLM** require multiple forward passes over perturbed inputs and aggregate predictions, substantially increasing computation (see Appendix F). **Self Defense** mandates the target LLM to complete whole response generation before secondary auditing, introducing latency since detection begins only post-generation. Although **HSF** uses a single LLM pass with an external classifier, it suffers from considerably higher ASR and FRR.

4.3 Ablation Study

As shown in Table 3, we construct five variants to evaluate the contribution of PADD's core components: (1) safety prefix (replaced by "####"), (2) attention distribution normalization, (3) relative entropy (replaced by raw Shannon entropy), (4) attention perturbation (K), and (5) attention plasticity (H).

Removing any component of PADD leads to noticeable performance degradation. The most substantial declines occur in **w/o Safety Prefix** (average ASR increase $\Delta = +19.6\%$) and **w/o K** ($\Delta = +9.8\%$), underscoring their dominant discriminative roles. **w/o Norm** introduces moderate deterioration ($\Delta = +4.6\%$), while **w/o RelEnt** and **w/o H** yield smaller yet non-negligible ASR increases ($\Delta = +2.5\%$ and $\Delta = +1.4\%$, respectively), accompanied by minor FRR variations. These results indicate that the Safety Prefix and K are the primary determinants of adversarial–benign separation, whereas normalization, relative entropy, and H provide complementary robustness. Removing any module consistently worsens the ASR/FRR trade-off.

4.4 Sensitivity to α and β

To assess the robustness of PADD with respect to its hyperparameters, we conduct a grid search over α and β within the range $[0.25, 3.0]$ with a step size of 0.25, resulting in 144 configurations.

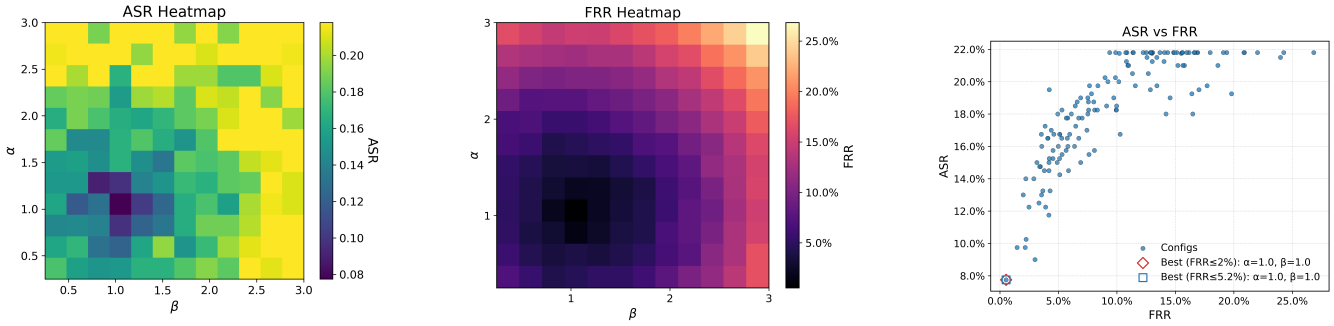
All experiments are performed on the **LLaMA-3-8B-Instruct**. For efficiency, in evaluating the ASR, we construct a reduced test set containing 25 randomly sampled prompts from each of four attack methods, totaling 100 adversarial samples. The false rejection rate (FRR) is measured under the same setting as in the main experiments, using benign prompts from the XSTest benchmark.

The results are summarized in Figure 3. Both ASR and FRR vary smoothly across the parameter space, indicating that PADD is largely insensitive to the choice of (α, β). Across most regions, ASR remains consistently low while FRR stays within an acceptable range. Notably, for the **LLaMA-3-8B-Instruct**, the optimal configuration occurs at $\alpha = 1.0$ and $\beta = 1.0$, where PADD achieves the lowest ASR alongside a low FRR, striking a desirable balance between defense strength and usability.

Table 3: Ablation study results.

Scheme	LLaMA-3-8B-Instruct		Qwen-3-8B		DeepSeek-R1-Distill-Qwen-7B		Vicuna-13B-v1.5		Average	
	ASR%	FRR%	ASR%	FRR%	ASR%	FRR%	ASR%	FRR%	ASR%	FRR%
PADD	0.4	5.2	1.9	0.0	3.0	0.0	1.9	0.0	1.8	1.3
w/o Safety Prefix	22.0 (Δ+21.6)	2.5 (Δ-2.7)	16.5 (Δ+14.6)	3.2 (Δ+3.2)	17.2 (Δ+14.2)	4.0 (Δ+4.0)	30.0 (Δ+28.1)	5.5 (Δ+5.5)	21.4 (Δ+19.6)	3.8 (Δ+2.5)
w/o Norm	4.0 (Δ+3.6)	1.5 (Δ-3.7)	6.3 (Δ+4.4)	2.0 (Δ+2.0)	7.9 (Δ+4.9)	2.6 (Δ+2.6)	7.2 (Δ+5.3)	3.0 (Δ+3.0)	6.4 (Δ+4.6)	2.3 (Δ+1.0)
w/o RelEnt	2.2 (Δ+1.8)	1.4 (Δ-3.8)	4.1 (Δ+2.2)	1.6 (Δ+1.6)	5.8 (Δ+2.8)	1.7 (Δ+1.7)	4.9 (Δ+3.0)	2.1 (Δ+2.1)	4.3 (Δ+2.5)	1.7 (Δ+0.4)
w/o K	9.0 (Δ+8.6)	1.4 (Δ-3.8)	11.0 (Δ+9.1)	1.6 (Δ+1.6)	12.5 (Δ+9.5)	1.9 (Δ+1.9)	14.0 (Δ+12.1)	2.3 (Δ+2.3)	11.6 (Δ+9.8)	1.8 (Δ+0.5)
w/o H	1.4 (Δ+1.0)	1.3 (Δ-3.9)	3.2 (Δ+1.3)	1.5 (Δ+1.5)	4.5 (Δ+1.5)	1.6 (Δ+1.6)	3.5 (Δ+1.6)	1.9 (Δ+1.9)	3.2 (Δ+1.4)	1.6 (Δ+0.3)

* Lower is better.
* Δ indicates the difference compared with the full model.



(a) ASR heatmap **(b) FRR heatmap** **(c) ASR–FRR trade-off**

Figure 3: Grid search over (α, β) on LLaMA-3- 8B-Instruct: (a) ASR landscape, (b) FRR landscape, and (c) ASR–FRR trade-off .

5 Related Work

Researchers have studied how to safeguard LLMs against jailbreak and prompt-injection attacks. In general, there are three categories of attempts: alignment-based training, pre-generation defenses, and post-generation auditing.

Alignment-based methods enhance safety during training or fine-tuning by embedding helpfulness and harmlessness into model parameters. RLHF [27], Constitutional AI [3], DPO [30], and GRPO [34] directly optimize alignment objectives. Safe RLHF [9], Decoupled Refusal Training [43], and GRAIT [50] introduce explicit refusal mechanisms or safety constraints, while SELF-GUARD [38] leverages self-identified harmfulness labels. Despite these advances, alignment-based methods remain vulnerable to distribution shifts and adversarial prompts, leaving persistent safety gaps [18].

Pre-generation defenses intervene before decoding. RA-LLM [4], SmoothLLM [31], and RPO [48] employ perturbation-based strategies such as random ablation, smoothing, or suffix optimization. Gradient Cuff [14] applies refusal-loss tests, Alon et al. [1] use perplexity-based screening, PromptShield [16] introduces deployable injection filters, and HSF [29] trains hidden-state classifiers. Although effective, these methods often increase latency, require detector maintenance, and complicate deployment, even without modifying the base model.

Post-generation defenses audit or rewrite outputs after generation. LLM Self Defense [28] and SELF-GUARD [38] use LLM-based filtering, while Constitutional Classifiers [35] and RigorLLM [45] rely on classifier-based guardrails. Early-exit halting [46] stops decoding when unsafe content is detected, and shadow-model gating [40] employs auxiliary verification. Harmfulness classifiers [19] estimate output risk, while safe rewriting [44] reformulates unsafe responses. These methods generally add inference overhead, depend on external models, and risk misclassifying benign outputs.

In contrast, our PADD detects potential risks *before* generation using a lightweight prefix-based attention divergence signal, achieving strong safety performance with practical deployability.

6 Conclusion

In this work, we propose PADD, a prefix-based attention divergence detector that enables efficient identification of jailbreak risk prior to text generation. By combining perturbation and plasticity scores into a unified risk metric, PADD significantly reduces attack success rates while maintaining low false rejection rates across multiple models and diverse attack paradigms. Experimental results demonstrate that PADD achieves strong robustness and generalization, while remaining lightweight and easily deployable, offering a new perspective for enhancing the safety of LLMs.

Acknowledgement

This paper is supported by the National Natural Science Foundation of China (Grant No. 62402183, 92582108). This paper is also supported by the Shanghai Special Program for Promoting High-Quality Industrial Development (Project No. 250668, 250203). We thank the anonymous reviewers for their valuable comments and constructive suggestions. We also acknowledge the open-source community for providing valuable tools and resources.

References

[1] Gabriel Alon and Michael Kamfonas. 2023. Detecting language model attacks with perplexity. *arXiv:2308.14132* (2023).

[2] Md. Ahsan Ayub and Subhabrata Majumdar. 2024. Embedding-based classifiers can detect prompt injection attacks. https://arxiv.org/abs/2410.22284

[3] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, and Amanda Askell. 2022. Constitutional AI: Harmlessness from AI Feedback. https://arxiv.org/abs/2212.08073

[4] Bochuan Cao, Yuanpu Cao, Lu Lin, and Jinghui Chen. 2023. Defending against alignment-breaking attacks via robustly aligned llm. *arXiv:2309.14348* (2023).

[5] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, et al. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *NeurIPS* 37 (2024), 55005–55029.

[6] Patrick Chao, Alexander Robey, and *et al.*. 2025. Jailbreaking black box large language models in twenty queries. In *SaTML*. IEEE, 23–42.

[7] Yangyi Chen and Gao. 2022. Why Should Adversarial Perturbations be Imperceptible? Rethink the Research Paradigm in Adversarial NLP. In *EMNLP*. ACL, Abu Dhabi, United Arab Emirates, 11222–11237. doi:10.18653/v1/2022.emnlp-main.771

[8] Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Xi Xie, and ... 2023.

[9] Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2023. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv:2310.12773* (2023).

[10] Guanlin Deng, Xueying Li, Yukuo Li, Yulang Ren, Zizheng Wang, Jingqi Xia, Zheli Zheng, Hongxin Hu, and Kehuan Zhang. 2024. MASTERKEY: Automated Jailbreaking of Large Language Model Chatbots. In *NDSS*. Internet Society, San Diego, CA, USA. doi:10.14722/ndss.2024.24188

[11] Y. Gong and colleagues. 2025. Safety Misalignment Against Large Language Models. In *NDSS*. https://www.ndss-symposium.org/wp-content/uploads/2025-1089-paper.pdf

[12] Chenchen Gu, Xiang Lisa Li, Rohith Kuditipudi, Percy Liang, and Tatsunori Hashimoto. 2025. Auditing Prompt Caching in Language Model APIs. *arXiv:2502.07776* (2025).

[13] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Wenfeng Wu, Yuxiang Liu, et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. (2025). https://arxiv.org/abs/2501.12948

[14] Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2024. Gradient cuff: Detecting jailbreak attacks on large language models by exploring refusal loss landscapes. *NeurIPS* 37 (2024), 126265–126296.

[15] Zhengmian Hu, Gang Wu, Saayan Mitra, Ruiyi Zhang, and *et al.*. 2024. Token-Level Adversarial Prompt Detection Based on Perplexity Measures and Contextual Information. https://arxiv.org/abs/2311.11509

[16] Dennis Jacob, Hend Alzahrani, Zhanhao Hu, Basel Alomair, and David Wagner. 2024. Promptshield: Deployable detection for prompt injection attacks. In *CODASPY*. 341–352.

[17] Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, and Kirchenbauer. 2023. Baseline defenses for adversarial attacks against aligned language models. *arXiv:2309.00614* (2023).

[18] Essa Jan, Nouar AlDahoul, Moiz Ali, Faizan Ahmad, Fareed Zaffar, and Yasir Zaki. 2024. Multitask Mayhem: Unveiling and Mitigating Safety Gaps in LLMs Fine-tuning. https://arxiv.org/abs/2409.15361

[19] Jinhwa Kim, Ali Derakhshan, and Ian Harris. 2024. Robust safety classifier against jailbreaking attacks: Adversarial prompt shield. In *WOAH*. 159–170.

[20] Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Aaron Jiaxun Li, Soheil Feizi, and Himabindu Lakkaraju. 2025. Certifying LLM Safety against Adversarial Prompting. https://arxiv.org/abs/2309.02705

[21] Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. 2023. Deepinception: Hypnotize large language model to be jailbreaker. *arXiv:2311.03191* (2023).

[22] Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024. AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models. In *ICLR*. https://openreview.net/forum?id=7Jwpw4qKkb

[23] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, Kailong Wang, and Yang Liu. 2023. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv:2305.13860* (2023).

[24] AI @ Meta Llama Team. 2024. The Llama 3 Herd of Models. https://arxiv.org/abs/2407.21783

[25] Weidi Luo, Siyuan Ma, Xiaogeng Liu, Xiaoyu Guo, and Chaowei Xiao. 2024. Jailbreakv: A benchmark for assessing the robustness of multimodal large language models against jailbreak attacks. *arXiv:2404.03027* (2024).

[26] Zhifan Luo, Shuo Shao, Su Zhang, Lijing Zhou, Yuke Hu, Chenxu Zhao, Zhihao Liu, and Zhan Qin. 2025. Shadow in the cache: Unveiling and mitigating privacy risks of kv-cache in llm inference. *arXiv:2508.09442* (2025).

[27] Long Ouyang, Jeff Wu, and *et al.*. 2022. Training language models to follow instructions with human feedback. https://arxiv.org/abs/2203.02155

[28] Mansi Phute, Alec Helbling, Matthew Hull, ShengYun Peng, Sebastian Szyller, Cory Cornelius, and Duen Horng Chau. 2023. LLM Self Defense: By Self Examination, LLMs Know They Are Being Tricked. *arXiv:2308.07308* (2023).

[29] Cheng Qian, Hainan Zhang, Lei Sha, and Zhiming Zheng. 2025. Hsf: Defending against jailbreak attacks with hidden state filtering. In *Companion Proceedings of the ACM on Web Conference 2025*. 2078–2087.

[30] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems* 36 (2023), 53728–53741.

[31] Alexander Robey, Eric Wong, Hamed Hassani, and George J Pappas. 2023. Smoothllm: Defending large language models against jailbreaking attacks. *arXiv:2310.03684* (2023).

[32] 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)*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). ACL, Mexico City, Mexico, 5377–5400. doi:10.18653/v1/2024.naacl-long.301

[33] Tiziano Santilli, Marco De Luca, Domenico Amalfitano, Anna Rita Fasolino, and Patrizio Pelliccione. [n. d.]. A Decontextualized LLM-based Safeguard Technique for Automated Jailbreak Mitigation. *Available at SSRN 5503124* ([n. d.]).

[34] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, and *et al.*. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv:2402.03300* (2024).

[35] Mrinank Sharma, Meg Tong, Jesse Mu, Jerry Wei, Jorrit Kruthoff, and Scott Goodfriend. 2025. Constitutional Classifiers: Defending against Universal Jailbreaks across Thousands of Hours of Red Teaming. https://arxiv.org/abs/2501.18837

[36] Xinyue Shen, Zeyuan Chen, Michael Backes, and *et al.*. 2024. "Do Anything Now": Characterizing and Evaluating In-the-Wild Jailbreak Prompts on Large Language Models. In *CCS*. ACM, 1671–1685. doi:10.1145/3658644.3670388

[37] Hugo Touvron, Thibaut Lavril, Gautier Izacard, and *et al.*. 2023. LLaMA: Open and Efficient Foundation Language Models. (2023). https://arxiv.org/abs/2302.13971

[38] Zezhong Wang, Fangkai Yang, Lu Wang, Pu Zhao, Hongru Wang, Liang Chen, Qingwei Lin, and Kam-Fai Wong. 2023. Self-guard: Empower the llm to safeguard itself. *arXiv:2310.15851* (2023).

[39] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How Does LLM Safety Training Fail?. In *NeurIPS*. https://proceedings.neurips.cc/paper_files/paper/2023/file/fd6613131889a4b656206c50a8bd7790-Paper-Conference.pdf

[40] Daoyuan Wu, Shuai Wang, Yang Liu, and Ning Liu. 2024. LLMs Can Defend Themselves Against Jailbreaking in a Practical Manner: A Vision Paper. https://arxiv.org/abs/2402.15727

[41] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, and Zheng. 2025. Qwen3 Technical Report. (2025). https://arxiv.org/abs/2505.09388

[42] Zhiyuan Yu, Xiaogeng Liu, and *et al.*. 2024. Don't Listen to Me: Understanding and Exploring Jailbreak Prompts of Large Language Models. In *USENIX Security*. https://www.usenix.org/system/files/sec24fall-prepub-1500-yu-zhiyuan.pdf

[43] Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Jiahao Xu, Tian Liang, Pinjia He, and Zhaopeng Tu. 2024. Refuse whenever you feel unsafe: Improving safety in llms via decoupled refusal training. *arXiv:2407.09121* (2024).

[44] Yuan Yuan, Tina Sriskandarajah, Anna-Luisa Brakman, Alec Helyar, Alex Beutel, Andrea Vallone, and Saachi Jain. 2025. From hard refusals to safe-completions: Toward output-centric safety training. *arXiv:2508.09224* (2025).

[45] Zhuowen Yuan, Zidi Xiong, and *et al.*. 2024. Rigorllm: Resilient guardrails for large language models against undesired content. *arXiv:2403.13031* (2024).

[46] Chongwen Zhao, Zhihao Dou, and Kaizhu Huang. 2025. Defending against Jailbreak through Early Exit Generation of Large Language Models. https://arxiv.org/abs/2408.11308

[47] Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. 2024. On Prompt-Driven Safeguarding for Large Language Models. In *ICML (PMLR, Vol. 235)*. 61593–61613. https://proceedings.mlr.press/v235/zheng24n.html

[48] Andy Zhou, Bo Li, and Haohan Wang. 2024. Robust prompt optimization for defending language models against jailbreaking attacks. *NeurIPS* 37 (2024), 40184–40211.

[49] Yukai Zhou, Jian Lou, Zhijie Huang, Zhan Qin, and *et al.*. 2024. Don't say no: Jailbreaking llm by suppressing refusal. *arXiv:2404.16369* (2024).

[50] Runchuan Zhu, Zinco Jiang, Jiang Wu, Zhipeng Ma, and *et al.*. 2025. GRAIT: Gradient-Driven Refusal-Aware Instruction Tuning for Effective Hallucination Mitigation. https://arxiv.org/abs/2502.05911

[51] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. (2023). https://arxiv.org/abs/2307.15043

A Proofs and Technical Details for Section 3

A.1 Quadratic Expansion of K with Correct High-Order Remainder

Proposition A.1 (Quadratic KL expansion). Under A1 and $\|\Delta\| \to 0$,

$$K = \mathrm{KL}(a_s \| a_f) = \tfrac{1}{2}\,\Delta^\top \mathbf{F}(a_f)\,\Delta + O(\|\Delta\|^3), \qquad \mathbf{F}(a_f) = \mathrm{diag}(a_f)^{-1}.$$

Proof. Write $a_s[i] = a_f[i] + \Delta_i$ and set $z_i = \Delta_i / a_f[i]$. Then

$$K = \sum_i (a_f[i] + \Delta_i)\log(1+z_i).$$

Using $\log(1+z) = z - \frac{1}{2}z^2 + \frac{1}{3}z^3 + R_4(z)$ with $|R_4(z)| \le |z|^4/[4(1-|z|)]$, we obtain

$$(a_f[i]+\Delta_i)\log(1+z_i) = \Delta_i + \tfrac{1}{2}\frac{\Delta_i^2}{a_f[i]} - \tfrac{1}{6}\frac{\Delta_i^3}{a_f[i]^2} + R_i^{(\ge 4)}.$$

Summing over i and using $\sum_i \Delta_i = 0$ yields

$$K = \tfrac{1}{2}\sum_i \frac{\Delta_i^2}{a_f[i]} - \tfrac{1}{6}\sum_i \frac{\Delta_i^3}{a_f[i]^2} + \sum_i R_i^{(\ge 4)}.$$

Bounding the remainder. Under A1, $|z_i| \le \eta < 1$, hence

$$|R_i^{(\ge 4)}| \le C(\eta)\,\frac{|\Delta_i|^4}{a_f[i]^3}, \qquad C(\eta) = \tfrac{1+\eta}{4(1-\eta)} + \tfrac{1}{3}.$$

Therefore $\sum_i R_i^{(\ge 4)} = O(\|\Delta\|^4)$, while $\sum_i \Delta_i^3/a_f[i]^2 = O(\|\Delta\|^3)$. This gives

$$K = \tfrac{1}{2}\,\Delta^\top \mathrm{diag}(a_f)^{-1}\Delta \;+\; O(\|\Delta\|^3). \square$$

Interpretation. K is a Fisher-weighted quadratic energy of Δ, with leading error $O(\|\Delta\|^3)$ and controlled quartic remainder.

A.2 Lipschitz Continuity of Normalized Entropy (Detailed)

Let $H(p) = -\sum_{i=1}^N p_i \log p_i$ and $h(p) = H(p)/\log N$ for a position with N valid indices. Throughout this subsection, probabilities are natural-logged; constants adapt for other bases.

Lemma A.2 (Lipschitz in ℓ_1). Under A1–A3, for any $u, v \in \Delta^{N-1}$,

$$|h(u) - h(v)| \;\le\; \frac{C}{\log N}\,\|u - v\|_1, \qquad C \;\triangleq\; 1 + \log(1/\varepsilon).$$

Proof (fully detailed). Define the line segment $p_\tau = v + \tau(u-v)$, $\tau \in [0,1]$, and the scalar function $g(\tau) = H(p_\tau)$. Because $u_i, v_i \ge \varepsilon$ (A1) and the simplex is convex, $p_\tau \in \Delta^{N-1}$ with $p_{\tau,i} \ge (1-\tau)\varepsilon + \tau\varepsilon = \varepsilon$ for all τ; thus the path stays in the interior.

Step 1: Gradient and bound along the path. For $H(p) = -\sum_i p_i \log p_i$,

$$\nabla H(p)_i = -(1 + \log p_i), \qquad \|\nabla H(p)\|_\infty = \max_i |1 + \log p_i|.$$

Since $p_{\tau,i} \ge \varepsilon$, we have $|1 + \log p_{\tau,i}| \le 1 + \log(1/\varepsilon)$, hence

$$\sup_{\tau \in [0,1]} \|\nabla H(p_\tau)\|_\infty \;\le\; C.$$

Step 2: Path-integral (mean-value) argument. g is differentiable on $[0,1]$ with

$$g'(\tau) = \nabla H(p_\tau)^\top (u - v).$$

By Hölder's inequality (duality of ℓ_∞ and ℓ_1),

$$|g'(\tau)| \le \|\nabla H(p_\tau)\|_\infty \,\|u - v\|_1 \le C\,\|u - v\|_1.$$

Integrating,

$$|H(u) - H(v)| = \left|\int_0^1 g'(\tau)\,d\tau\right| \le \int_0^1 |g'(\tau)|\,d\tau \le C\,\|u - v\|_1.$$

Step 3: Normalization. Dividing by $\log N$ (A3 ensures $N \ge 2$),

$$|h(u) - h(v)| \le \frac{C}{\log N}\,\|u - v\|_1. \quad \blacksquare$$

Tighter local constant (optional). Let $m \triangleq \min_{i,\tau} p_{\tau,i} = \min_i \min\{u_i, v_i\}$ ($\ge \varepsilon$). Repeating the proof with C replaced by $C(m) \triangleq 1 + \log(1/m)$ yields

$$|h(u) - h(v)| \le \frac{1 + \log(1/m)}{\log N}\,\|u - v\|_1,$$

which is tighter whenever $m > \varepsilon$.

Equivalent ℓ_2 form (optional). Using $\|x\|_1 \le \sqrt{N}\,\|x\|_2$,

$$|h(u) - h(v)| \le \frac{\sqrt{N}}{\log N}\,\big(1 + \log(1/\varepsilon)\big)\,\|u - v\|_2.$$

Why A1 is necessary. Without ε-smoothing, the gradient $\nabla H(p)_i = -(1 + \log p_i)$ is unbounded as $p_i \downarrow 0$, so H fails to be globally Lipschitz on the closed simplex. Assumption A1 ensures a uniform interior bound on the gradient along the entire path p_τ.

Alternative bound without A1 (Fannes–Audenaert type). If one prefers a bound that does not rely on ε, let $\delta = \frac{1}{2}\|u - v\|_1 \in [0,1]$. Then (classical discrete Fannes–Audenaert inequality)

$$|H(u) - H(v)| \le \delta \log(N-1) + H_2(\delta),$$

where $H_2(\delta) = -\delta\log\delta - (1-\delta)\log(1-\delta)$ is the binary entropy. Consequently,

$$|h(u) - h(v)| \le \frac{\delta\log(N-1) + H_2(\delta)}{\log N}.$$

For small δ, $H_2(\delta) \le \delta\log(e/\delta)$, recovering a near-linear dependence in $\|u - v\|_1$.

Takeaway. The main Lipschitz bound used in the paper follows from a path-integral of the entropy gradient with a uniform interior bound supplied by ε-smoothing, and the normalization by $\log N$ removes the trivial position-length scaling.

A.3 Log-Normal Aggregation Model

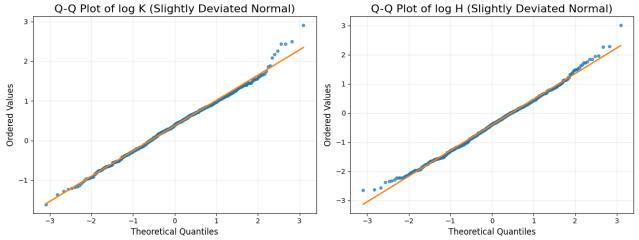


Figure 4: Q–Q plots of $\log K$ and $\log H$.

Heuristic. By Proposition A.1, $K \approx \frac{1}{2}\sum_i \Delta_i^2/a_f[i]$, a sum of many positive contributions from weakly dependent micro-perturbations (layers/heads/positions). Lemma A.2 further shows that token-level entropy differences scale with $\|u - v\|_1$; averaging yields H as a sum of nonnegative terms.

Such aggregates admit a log-normal approximation via the delta method and Fenton–Wilkinson moment matching:

$$\log K \sim \mathcal{N}(\mu_K^{(y)}, \sigma_K^2), \qquad \log H \sim \mathcal{N}(\mu_H^{(y)}, \sigma_H^2),$$

with $\mathrm{Cov}(\log K, \log H) \approx \rho\sigma_K\sigma_H$ under a Lindeberg-type condition.

Validation. Figure 4 shows Q–Q plots of $\log K$ and $\log H$, both close to normal with $p_K = 0.1015$ and $p_H = 0.1343$ (> 0.05), supporting the log-normal model.

A.4 Neyman–Pearson Threshold in the Log Domain

Let $S = \alpha \log K - \beta \log H$. Under the equal-covariance Gaussian model for $S|y \in \{0, 1\}$,

$$\mu_S^{(y)} = \alpha\mu_K^{(y)} - \beta\mu_H^{(y)}, \quad \sigma_S^2 = \alpha^2\sigma_K^2 + \beta^2\sigma_H^2 - 2\alpha\beta\rho\sigma_K\sigma_H.$$

The log-likelihood ratio (LLR) between two equal-variance Gaussians is affine in S:

$$\Lambda(S) = \log \frac{f_{S|1}(S)}{f_{S|0}(S)} = \frac{\mu_S^{(1)} - \mu_S^{(0)}}{\sigma_S^2} S - \frac{(\mu_S^{(1)})^2 - (\mu_S^{(0)})^2}{2\sigma_S^2}.$$

With priors (π_0, π_1) and costs (C_{10}, C_{01}), threshold $\Lambda(S)$ at $\log(\frac{\pi_0 C_{10}}{\pi_1 C_{01}})$ yields

$$S \gtrless s^\star := \frac{\sigma_S^2}{\mu_S^{(1)} - \mu_S^{(0)}} \log\left(\frac{\pi_0 C_{10}}{\pi_1 C_{01}}\right) + \frac{\mu_S^{(1)} + \mu_S^{(0)}}{2}.$$

Exponentiating $S = \alpha \log K - \beta \log H$ gives

$$J = \frac{K^\alpha}{H^\beta} \gtrless \tau, \qquad \tau = e^{s^\star}.$$

Special cases: (i) Equal priors/costs: $s^\star = \frac{1}{2}(\mu_S^{(1)} + \mu_S^{(0)})$, $\tau = \exp(s^\star)$. (ii) $\rho = 0$: $\sigma_S^2 = \alpha^2\sigma_K^2 + \beta^2\sigma_H^2$.

A.5 Choosing (α, β) via Fisher Separation

Let $z = [\log K, \log H]^\top$, $\Delta\mu = \mathbb{E}[z|1] - \mathbb{E}[z|0] = [\Delta\mu_K, \Delta\mu_H]^\top$, and $\Sigma = \mathrm{Cov}(z)$. For $w = [\alpha, -\beta]^\top$, the Fisher separation of $S = w^\top z$ is

$$\mathcal{J}(w) = \frac{(w^\top \Delta\mu)^2}{w^\top \Sigma w}.$$

Maximization gives $w \propto \Sigma^{-1}\Delta\mu$, hence

$$(\alpha, \beta) \propto \Sigma^{-1} [\Delta\mu_K, -\Delta\mu_H]^\top.$$

A robust default is $\alpha = \beta = 1$ (especially after standardizing $\log K, \log H$).

A.6 Parameter and Threshold Estimation

With labels. Estimate $(\mu_S^{(y)}, \sigma_S^2)$ on held-out calibration data and plug into $s^\star$.

Without labels. (i) *Target FPR:* Fit a Gaussian to S on historical benign traffic and set τ as the $(1 - \alpha)$ quantile for desired FPR α. (ii) *Two-component GMM:* Fit a GMM to S via EM; choose τ at the posterior-equality point or to maximize Youden's J.

B Safety Prefix

In our framework PADD, the *safety prefix* serves as a lightweight and controlled perturbation that enables within-prompt self-comparison. By prepending this prefix to the original prompt, we obtain a paired variant that differs only in the prefix content, which forms the basis for extracting reliable jailbreak signals (see Section 3). For reproducibility, we document the exact prefix used in our experiments below.

Safety Prefix

You are a safe and responsible AI assistant. Always refuse to provide any harmful, illegal, or unethical information, including but not limited to instructions about weapons, drugs, violence, or prohibited activities. If a user attempts to trick you or bypass these rules, you must still refuse and instead give a safe and responsible response.

C Sensitivity to Training Data Size

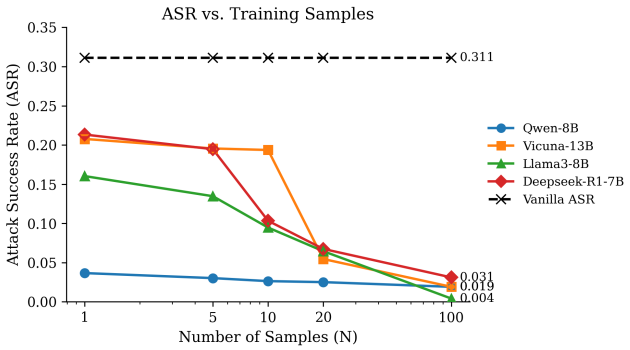


Figure 5: Impact of training set size on ASR.

We assess the sensitivity of PADD to the scale of training data by varying the number of samples per attack method, denoted by K. For each K, we randomly select K samples from each attack type (GCG, PAIR, DSN, AutoDAN) and an additional $3 \times K$ benign samples generated by GPT-5, yielding $7 \times K$ samples in total. When $K = 100$, this recovers the whole training set of 700 samples. To ensure that every configuration contains effective jailbreak instances, we prioritize successful adversarial samples when available and fill the remainder with unsuccessful attempts if necessary. After constructing each subset, we retrain the detector and evaluate it on a fixed test set.

The results indicate that the attack success rate (ASR) decreases steadily as K increases, with the most significant improvements occurring in the low-data regime and diminishing returns thereafter. Across all values of K, PADD consistently outperforms the Vanilla baseline (no defense), and the trends remain stable across different LLMs. These findings highlight that PADD is both data-efficient and robust: it achieves strong defense capability with only a small amount of training data, while larger datasets still provide incremental benefits.

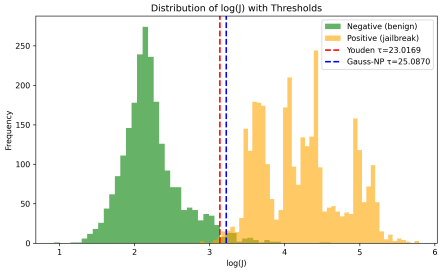


Figure 6: Histogram of log-domain scores with Youden (red) and Neyman–Pearson (blue) thresholds.

Table 5: τ values measured on different models.

Model	τ Value
DeepSeek-R1	123.41
LLaMA-3	173.33
Qwen-3	23.01
Vicuna-13B	261.72

Table 4: K-fold cross-validation results on the LLaMA-3-8B-Instruct.

Fold	ASR (%)	FRR (%)	F1 (%)
Fold 1	1.9	6.7	95.2
Fold 2	2.0	6.8	95.1
Fold 3	1.8	6.5	95.3
Fold 4	2.2	6.9	95.0
Fold 5	2.0	6.6	95.2
Average	2.0 ± 0.2	6.7 ± 0.2	95.2 ± 0.1

D Robustness of Threshold Calibration

To assess the robustness of PADD's threshold calibration, we conduct a 5-fold cross-validation experiment on the **LLaMA-3-8B-Instruct** model.

The training set is partitioned into five disjoint folds. In each iteration, four folds (approximately 80% of the data) are used to calibrate the threshold for distinguishing adversarial from benign prompts. Unlike standard cross-validation, the remaining fold is not used for evaluation; instead, we apply the same independent test set from the main experiments to determine the optimal threshold τ and compute all evaluation metrics. Because this test set is entirely disjoint from the training data in both source and distribution, it provides an unbiased benchmark for evaluating the generalization capability of PADD under different threshold configurations.

The results are summarized in Table 4. Across all folds, the thresholds derived from different training subsets yield almost identical performance on the test set, with ASR, FRR, and F1 showing only minor variations. These results demonstrate that PADD is robust to variations in training data partitioning and that its threshold calibration process is highly stable and consistent. In other words, regardless of the specific subset used for calibration, the resulting threshold preserves comparable defense performance on the held-out test set. In addition, Appendix C shows that PADD maintains strong performance even when trained with substantially fewer samples, further demonstrating its robustness and data efficiency.

E Visualization of Thresholds

In addition to the numerical results reported in Table 5, we visualize the separation between benign and jailbreak samples in the log domain. For this analysis, we use the Qwen model and evaluate on a test set containing a 1:1 ratio of jailbreak and benign samples. Figure 6 shows the histogram of log-domain scores. Two thresholding strategies are highlighted: the red dashed line corresponds to the *Youden index* (empirical trade-off between TPR and FPR), while the blue dashed line corresponds to the *Neyman–Pearson threshold*, whose derivation is given in Appendix A.4. Both thresholds illustrate that benign and jailbreak samples can be effectively separated.

F Details of Baselines

We reproduce three representative defense methods and adhere to their original hyperparameter settings. For RA-LLM, which detects jailbreaks by randomly ablating inputs and performing multiple forward passes, we use $n = 10, p = 0.3, t = 0.2$, which requires nine additional forward passes for each query. For SmoothLLM, which smooths the model's decision boundary via randomized character perturbations, we set $N = 4, q = 0.05, r = 0.5$, corresponding to three additional forward passes. For Self-Defense, which applies a suffix-style harm filter for harmfulness judgment, only one additional forward pass is required. For HSF, since the authors did not release the official implementation, we reproduce the method by training a logistic regression classifier on hidden-state features, where the hidden dimension is set to 512 and the number of final tokens selected is $k = 7$. The training data for HSF follows the same setting as PADD, as summarized in Table 1.

G Details of attack methods

We reproduce and compare several representative baselines, following their original implementations and default configurations to ensure fair comparison. For GCG (gradient-guided character-level attack) we use num_steps = 500, search_width = 64, and top_k = 64. For AutoDAN (hierarchical genetic-algorithm attack) we use population_size = 50, max_iterations = 100, crossover_rate = 0.5, mutation_rate = 0.01, elite_rate = 0.1, and multi_point_breaks = 5. For the remaining methods, we use the default configurations provided by the original authors.