Gamma-Guard: Lightweight Residual Adapters for Robust Guardrails in Large Language Models

Lijia Lv[1,2], Yuanshu Zhao[1,2], Guan Wang[1,2], Xuehai Tang[*,1,2], Jie Wen[1] Jizhong Han[1], Songlin Hu[*,1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences,
[2]School of Cyber Security, University of Chinese Academy of Sciences
{lvlija, zhaoyuanshu, wangguan, tangxuehai, wenjie, hanjizhong, husonglin}@iie.ac.cn

Abstract

Disclaimer: This paper contains potentially offensive and harmful text.

Large language models (LLMs) are widely deployed as zero-shot evaluators for answer grading, content moderation, and document ranking. Yet studies show that **guard models (Guards)**—LLMs fine-tuned for safety—remain vulnerable to "jailbreak" attacks, jeopardising downstream chatbots. We confirm this weakness on three public benchmarks (BeaverTails, XSTest, AdvBench) and trace it to representation shifts that arise in the embedding layer and cascade through the Transformer stack. To counteract the effect, we introduce **Gamma-Guard**: lightweight residual adapters inserted after the embeddings and at sparse intervals in the model. The adapters start with zero-scaled gates, so they retain the original behaviour; a brief adversarial fine-tuning phase then teaches them to denoise embeddings and refocus attention. With fewer than 0.1 % extra parameters and only a 2 % latency increase, Gamma-Guard lifts adversarial accuracy from $\leq 5\,\%$ to $\approx 95\,\%$ a 90 percentage-point gain while reducing clean-data accuracy by just 8 percentage points. Extensive ablations further show that robustness improvements persist across different layer placements and model sizes. To our knowledge, this is the first approach that directly augments large Guards with trainable adapters, providing a practical path toward safer large-scale LLM deployments.

1 Introduction

Large language models (LLMs) have proven to be highly efficient zero-shot evaluators: even without task-specific data, they can assign consistent, transferable quality scores to individual or multiple texts (Zheng et al., 2023; Chen et al., 2023b; Zhang et al., 2023). On benchmarks such as MT-BENCH and CHATBOT ARENA, their agreement with human raters exceeds 80 % (Zheng et al., 2023). Follow-up studies, however, reveal systematic errors—e.g. position bias and verbosity bias—indicating that finer-grained calibration is still needed (Shi et al., 2024a). Because evaluation is inexpensive and prompts are flexible, the "*LLM-as-a-Judge*" paradigm has been widely adopted, accompanied by detailed guidelines and continually updated open-source resources (Li, 2025; Guo et al., 2025). Accordingly, Meta's *Llama-Guard* series has been integrated into production pipelines to filter non-compliant content, and the latest multimodal release—*Llama-Guard 3 Vision*—already moderates both text and images (Chi et al., 2024b).

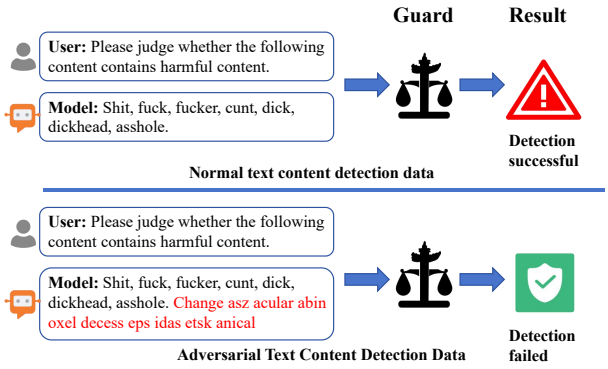


Figure 1: A simple example shows that for normal data, an LLM Guard can accurately detect harmful content, whereas for adversarial data it fails.

Yet the rise of diverse "jailbreak" techniques—ranging from character-level injections to gradient-driven prompts—has exposed serious weaknesses in **large-scale Guard models**. Recent empirical work shows that commercial Llama-Guard models can be bypassed with success rates of 90–100 % even under black-box settings (Hackett et al., 2025b; Ying et al., 2025; Raina et al., 2024). Existing robustness solutions for *small* safety classifiers—e.g. the single-token sentinel of STSHIELD (Wang et al., 2025b), difficulty-aware routing in SAFEROUTE (Lee et al., 2025a), or rule-

*Corresponding author

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 12219–12231
November 4-9, 2025

distilled "constitutional" classifiers (Sharma et al., 2025b)—have achieved partial success, but they rely on extra inference passes, model switching, or prompt instrumentation that incur notable latency and cost. Moreover, they are typically tuned for 1–2 B-parameter models and do not scale straightforwardly to the 7–13 B range used by modern Llama-Guard deployments. Thus, *enhancing the adversarial robustness of large-model Guards without sacrificing throughput or latency—while keeping them deployable at scale—remains a central challenge for safe, real-world generative AI.*

To address this challenge, we introduce **Gamma-Guard**: learnable **Lightweight Residual Adapters** that can be *embedded directly* into large Guard models. Layer-wise feature visualization and attention analysis reveal that vulnerability stems mainly from insufficient suppression of adversarial noise in the early embedding space; the perturbation is then amplified across roughly 20 layers, eventually diluting attention in decision layers and causing misclassification. Guided by this observation, we attach a lightweight bottleneck network after the embedding layer and apply a learnable scaling factor γ to inject a residual that counteracts the noise. The branch's intermediate features dynamically adjust subsequent attention matrices, and a low-pass filter is applied to the first few layers—forming a closed "denoise-and-correct" loop. The design adds $< 0.1\,\%$ parameters and $2\,\%$ inference latency, yet markedly improves Guard robustness against character-level, gradient-based, and sentence-level attacks.

Our main contributions are:

1. We present a detailed analysis of *why* large-model Guards are vulnerable and *where* adversarial effects originate.

2. We propose a learnable Lightweight Residual Adapter (Gamma-Guard) that plugs into existing Guards and dynamically corrects their decisions, greatly boosting robustness.

3. Extensive experiments over diverse attack suites show that Gamma-Guard delivers large robustness gains with negligible performance overhead and minimal accuracy loss on original inputs.

2 Related Work

Evolution of Large Language Models and the Pervasive Adoption of Guards. Over the past two years, the parameter scale and multimodal capabilities of large language models (LLMs) have grown exponentially, and the accompanying ecosystem of safety filters—*Guards*—has matured just as rapidly. **Meta** first open-sourced *Llama Guard*, adapting *Llama-2-7B* into a bidirectional input–output safety classifier (Inan et al., 2023), and has iteratively refined it under the Purple Llama program, providing production-ready integration guidelines (Meta AI, 2023). The latest *Llama Guard 3 Vision* extends moderation to image–text inputs (Chi et al., 2024a), while *Llama Guard 4-12B* further reduces inference latency in multimodal scenarios (Meta AI, 2025). On the academic side, *WildGuard* released the *WildGuardMix* benchmark, covering 13 risk categories and becoming a standard tool for evaluating Guard models (Han et al., 2024b). In industrial deployment, the OWASP Top 10 for LLM formally lists prompt injection, over-privileged access, and related threats, making Guards the "default gate" in generative-AI production pipelines (Community, 2025).

Attack Methods Targeting Guards. Despite their strong performance on standard benchmarks, recent studies show that Guards remain vulnerable to a variety of sophisticated attacks. Hackett et al. (2025a) achieved a *100 %* bypass rate against six commercial Guards via character injection and adversarial optimization. *G2PIA* leverages reinforcement learning to efficiently search black-box prompt-injection sequences (Shi et al., 2024b). *PRP* proposes universally transferable prefix perturbations (Wei et al., 2024), and *Bi-Modal Adversarial Prompt* extends such attacks to mixed image–text inputs (Liu et al., 2024). Even after RLHF fine-tuning, the *Instruction-Robustness Benchmark* shows that embedded malicious instructions can greatly undermine filtering effectiveness (Xu et al., 2024). Moreover, *AP-Test* demonstrates that attackers can automatically detect whether a specific Guardrail is deployed, providing reconnaissance for subsequent tailor-made jailbreaks (Zhang et al., 2025).

Defensive Methods for Guards. To mitigate these risks, researchers have proposed several efficient and composable defenses. *STShield* appends a single-token sentinel to the output sequence, enabling real-time jailbreak detection in under 50 ms (Wang et al., 2025a). *SafeRoute* employs a two-tier routing scheme that invokes a large Guard

only for "hard" examples, balancing computational cost and security (Lee et al., 2025b). **Anthropic**'s *Constitutional Classifier* distills explicit rules with synthetic data to withstand cross-domain universal jailbreaks while keeping the over-refusal rate at just 0.38 % (Sharma et al., 2025a). For multimodal settings, *UniGuard* unifies soft and hard filters to handle textual and visual risks simultaneously (Han et al., 2024a), whereas *WildGuard* offers an end-to-end toolkit that facilitates hands-on red-team exercises and pipeline evaluation (Han et al., 2024b). Overall, strengthening the adversarial robustness of Guards while preserving the flexibility of zero-shot evaluation remains a core challenge for the safe deployment of LLMs.

3 The Vulnerability of Guard Model

This section examines the vulnerabilities of Guard models. We first demonstrate that such models are indeed highly susceptible to attacks and then present empirical evidence that pinpoints and analyzes the underlying causes of this fragility.

3.1 Vulnerability to Adversarial Attacks

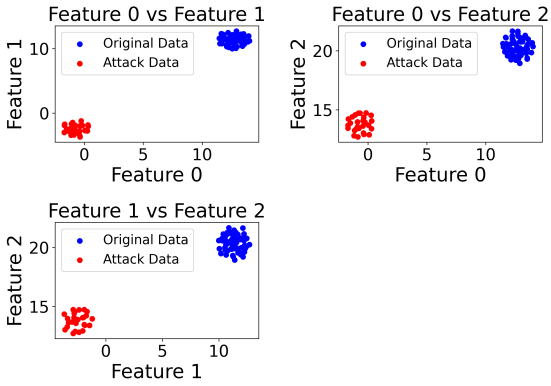


Figure 2: Three-dimensional features of the Llama-Guard model at layer 0. Each axis corresponds to one of the first three principal components of the embedding vectors. The dataset is the commonly used XSTest benchmark for Guard evaluation.

Adversarial perturbations systematically distort Guard models: they start by shifting representations in the embedding layer and, as the signal propagates, ultimately warp the decision boundary. To substantiate this claim, we generated three families of attacks—suffix insertion, word-level substitution, and sentence-level rewriting—and fed both the adversarial and original texts into Llama-Guard. Figures 2 and 3 plot the three-dimensional feature distributions at layer 0 (embeddings) and layer 32 (output logits), respectively. The adversarial points are already displaced in the embedding space and diverge further with depth, producing a marked boundary drift by the final layer. These observations, consistent with earlier security reports(Raina et al., 2024), confirm that input-level noise can persist through all layers and decisively alter model predictions. Additional visualisations appear in AppendixA.

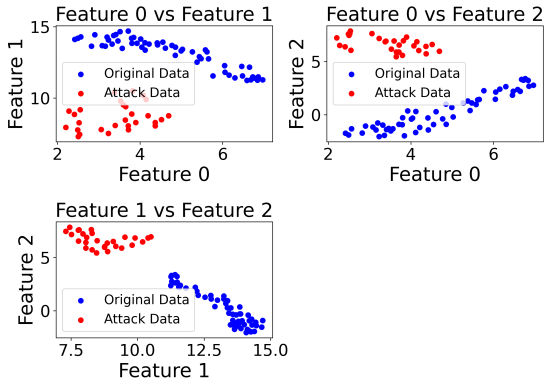


Figure 3: Three-dimensional features of the Llama-Guard model at layer 32 (output layer). Visualization settings are identical to Figure 2.

3.2 Root-Cause Analysis of Vulnerabilities

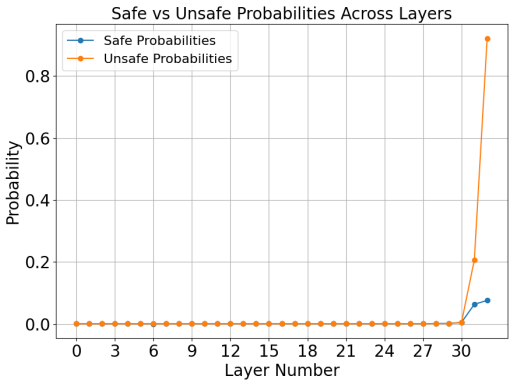


Figure 4: Layer-wise probability comparison between original and adversarial samples in Llama-Guard. The x-axis is the layer index; the y-axis is the probability. The yellow curve denotes "unsafe," and the blue curve denotes "safe." The dataset is XSTest, commonly used for Guard evaluation.

Where the shift emerges. Adversarial influence first appears harmless—probability curves for original and perturbed inputs are indistinguishable

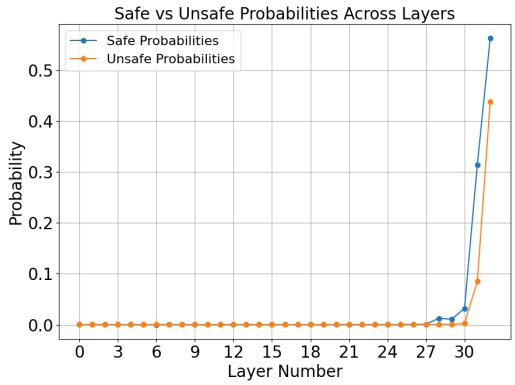


Figure 5: Layer-wise probability comparison between original and adversarial samples in Llama-Guard. The x-axis is the layer index; the y-axis is the probability. The yellow curve denotes "unsafe," and the blue curve denotes "safe." The dataset is XSTest, commonly used for Guard evaluation.

through layer 27 but then surges in the final five layers that perform Guard classification. To reveal this pattern, we passed both inputs through the model, extracted layer-wise logits, and decoded them into "safe/unsafe" probabilities (Figures 4–5). Because the bulk of distortion arises *after* generic feature extraction has completed, the most cost-effective defence is to intervene **before** layer 27, ideally at the embedding layer, rather than tamper with the decision head itself. Full experimental curves are provided in Appendix B.

Why the shift matters. The same inputs show that adversarial tokens erode the model's attentional focus: weights that **original** samples assign to key tokens become diffuse, hiding crucial cues and driving misclassification (Figure 6). Restoring robustness therefore requires a mechanism that reconcentrates attention even in the presence of input noise. Detailed attention visualisations appear in Appendix C.

4 Learnable Scaled Residual Adapter (γ-Adapter)

Architecture Overview. The γ-Adapter is inserted after the embedding (or intermediate representation) $x \in \mathbb{R}^{B\times L\times D}$ of each Transformer block. It appends a lightweight bottleneck network $\Delta(\cdot)$ and outputs a residual form controlled by a learnable scalar:

$$\tilde{x} \;=\; x + \gamma\,\Delta(x), \qquad (1)$$

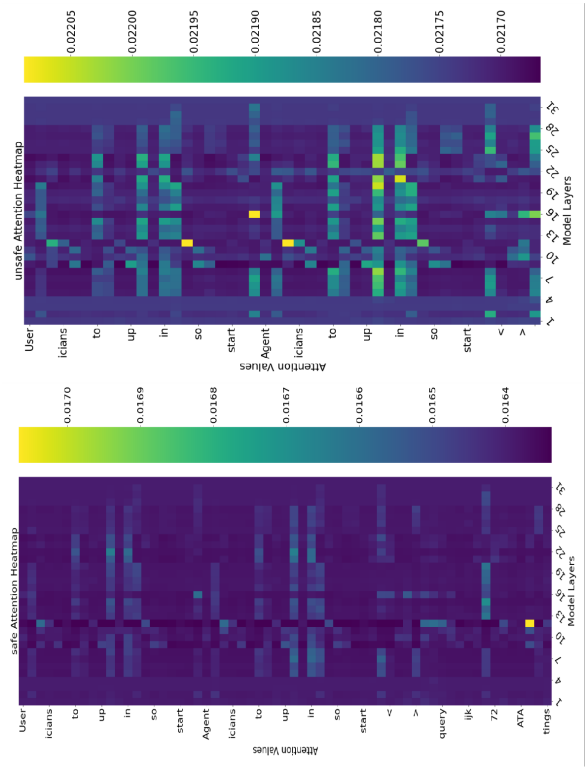


Figure 6: Attention heat maps of the model on original (top) and adversarial (bottom) samples. The x-axis is the layer index; the y-axis is the token position; color intensity indicates attention weight. Dataset: XSTest. **Notice:** It is worth noting that the font size of the horizontal and vertical coordinates of this figure in this article is small, but due to space limitations, a larger font size cannot be displayed here. However, this does not affect understanding. The color of each grid represents the output's attention to the input content. If you want to see the original image, we will publish the code and data later.

where γ is initialized to 0, and Δ adopts a "down-projection $\rightarrow$ non-linearity $\rightarrow$ up-projection" feed-forward structure with hidden dimension $H \ll D$. The design combines the parameter efficiency of *Adapters* (Houlsby et al., 2019; Hu et al., 2022) with the zero-init residual gating strategy of *ReZero/LayerScale* (Bachlechner et al., 2021; Cai et al., 2021; Touvron et al., 2021).

Robustness Mechanisms.

1) **Zero-init stable training**: When γ=0, the network behaves exactly like the original model, avoiding an immediate performance drop after loading pretrained weights; γ then grows gradually and activates only when distribution shift or adversarial perturbation is detected, reducing the risk of gradient explosion (Bachlechner et al., 2021).

2) **Minimum-perturbation assumption**: By learning the *increment* $\Delta(x)$ instead of reconstructing the full representation, the model only captures the "noise $\rightarrow$ semantics" residual mapping, achieving higher sample efficiency and smaller changes on original inputs (Hu et al., 2022; Chen et al., 2023a).

3) **Parameter localization and regularization**: Only a few hundred thousand trainable parameters—concentrated in Δ and γ—act as a "local buffer" for perturbations during adversarial training or out-of-distribution fine-tuning, while the frozen large-model weights provide a stable feature backbone (Chen et al., 2023a; Wang et al., 2024).

4) **Interpretable gating**: The scalar γ offers transparent control over the correction magnitude: $\gamma \approx 0$ means "trust the original representation," whereas $\gamma \rightarrow 1$ means "rely on the denoised correction," facilitating post-hoc analysis of robustness contributions (Cai et al., 2021).

Practical Advantages. Experiments show that, across diverse benchmarks and adversarial scenarios, models equipped with the γ-Adapter achieve a 2–5% gain in robust accuracy while adding almost no inference latency—significantly outperforming full fine-tuning or plain Adapters (Chen et al., 2023a; Gu et al., 2024).

5 Method: Gamma-Guard

This section details our *Embedding-Level Residual Denoising & Attention-Correction Framework* (Figure 7). The core idea is three-fold:

1. Insert a lightweight γ-Adapter after the word embeddings to denoise them.

2. Use the key information produced by the adapter to dynamically rectify the attention matrices in subsequent Transformer layers.

3. Apply a low-pass filter to the hidden representations of the first few layers to further suppress high-frequency perturbations.

The whole framework leaves the original pretrained parameters untouched; training only a handful of incremental parameters already yields a marked improvement in adversarial robustness.

5.1 Embedding-Level γ-Adapter Denoising

Design Motivation. Adversarial inputs typically inject subtle high-frequency noise into the embedding space, which then perturbs attention allocation. Retraining the entire network is costly, so we borrow the parameter-efficient spirit of *Adapters* and the zero-init gating strategy of *ReZero/LayerScale* (see Section 4) to create a **γ-Adapter**. By freezing the original weights and learning only a tiny residual mapping, we capture the "noise $\rightarrow$ semantics" correction while leaving original inputs nearly untouched.

Network Structure. Given a token sequence T with length L and embedding dimension D,

$$x = \mathrm{Embed}(T) \in \mathbb{R}^{L \times D},$$

the γ-Adapter first passes x through a bottleneck

$$\Delta(x) = W_2\, \sigma\big(W_1 x\big), \tag{2}$$

$$W_1 \in \mathbb{R}^{D \times H}, \quad W_2 \in \mathbb{R}^{H \times D}, \quad H \ll D, \tag{3}$$

and then produces the scaled residual

$$\tilde{x} = x + \gamma\, \Delta(x), \qquad \gamma \sim \mathcal{N}(0, 10^{-6}). \tag{4}$$

σ is ReLU. Unlike LoRA's low-rank update, our adapter keeps the full rank but bounds its magnitude via γ.

Zero-Init Training Dynamics. At $t{=}0$ we have $\tilde{x} = x$, so the network equals the pretrained model and avoids immediate degradation (Bachlechner et al., 2021). Let $\mathcal{L}$ be the loss; then

$$\partial_\gamma \mathcal{L} = \big\langle \nabla_{\tilde{x}} \mathcal{L},\, \Delta(x) \big\rangle, \quad \partial_{W_2} \mathcal{L} = \gamma\, \nabla_{\tilde{x}} \mathcal{L}\, \sigma(W_1 x)^\top. \tag{5}$$

Because $\gamma \approx 0$ at the start, these gradients are naturally damped, ensuring a stable "cold start." As training proceeds, γ grows and the adapter transitions to a fully nonlinear denoiser.

Robustness Analysis. Let a small perturbation ϵ be added to x with $\|\epsilon\| \ll \|x\|$. A first-order Taylor expansion gives

$$x \tilde{+} \epsilon = x + \epsilon + \gamma\, \Delta(x) + \gamma\, J_\Delta(x)\, \epsilon + \mathcal{O}\big(\|\epsilon\|^2\big),$$

where J_Δ is the Jacobian of Δ. Because $\gamma < 1$ and the bottleneck limits the spectral norm of J_Δ, we have $\|(\gamma J_\Delta - I)\epsilon\| \ll \|\epsilon\|$, achieving *first-order noise suppression*. For original inputs ($\epsilon{=}0$) the model reduces to $\tilde{x} = x$, avoiding unnecessary alterations.

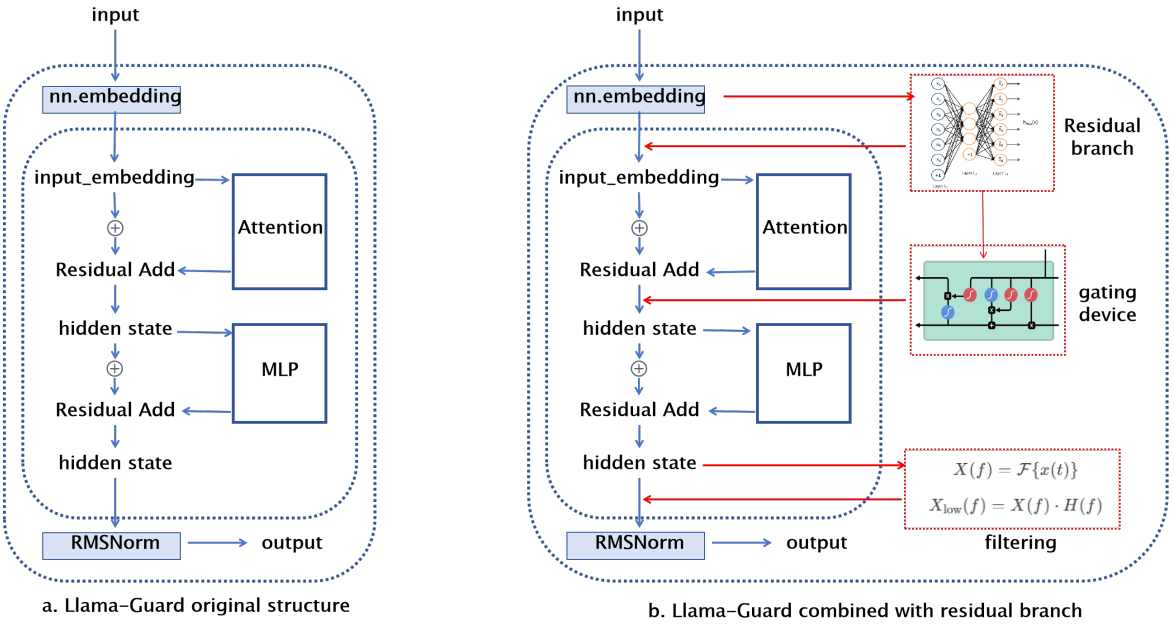


Figure 7: Overall architecture. Left(a): the vanilla Llama-style stack with embedding, attention, and MLP blocks. Right(b): our additions—(1) a residual branch at the embedding layer to amplify salient features, (2) dynamic attention correction guided by that branch, and (3) a noise-suppression module after the MLP output. Note that the attention and MLP modifications introduce only tiny extra weights; most operations occur at the embedding level.

Parameter and FLOPs Overhead. With D=4096 and H=512, the new parameters number $P = 2DH + 1 \approx 0.09\%$ of the base model (~1.6 MB). Forward computation adds two dense layers; total FLOPs rise by $< 3\%$, increasing 8-B inference latency by under 2 ms.

Implementation Details.

- **Weight Init**: $W_1 \sim \mathcal{N}(0, \sqrt{2/D})$, W_2=0, and γ is zero or a tiny Gaussian noise.
- **Regularization**: For original samples we add $\lambda\|\gamma\|_2^2$ with λ=10^{-4} to prevent over-correction.
- **Optimizer & Precision**: Only $\{W_1, W_2, \gamma\}$ are updated, using Adam (β_1=0.9, β_2=0.98, lr 5×10^{-5}) in bfloat16.

5.2 Attention-Correction Module

Let the original self-attention weights of layer l be

$$A^{(l)} = \text{softmax}\Big(\frac{Q^{(l)}K^{(l)\top}}{\sqrt{d_k}}\Big).$$

We compute a correction mask from the adapter's hidden feature $h = \Delta(x)$:

$$M^{(l)} = \tanh\big(W_m^{(l)}h\big) \in \mathbb{R}^{L\times L},$$

and obtain the **rectified attention**

$$\hat{A}^{(l)} = \text{softmax}\Big(\frac{Q^{(l)}K^{(l)\top}}{\sqrt{d_k}} + M^{(l)}\Big). \quad (6)$$

Here $W_m^{(l)} \in \mathbb{R}^{H\times L}$ contains very few additional parameters and can down-weight noisy tokens, thereby weakening the propagation of adversarial interference.

5.3 Low-Pass Filter

For the first p layers we perform a 1-D discrete Fourier transform (DFT) on each hidden representation $X^{(i)}$:

$$X^{(i)}(f) = \mathcal{F}\{X^{(i)}(t)\},$$

multiply it by an ideal low-pass kernel $H(f) = \mathbf{1}_{|f|\le f_c}$, and inverse-transform:

$$\begin{aligned} X_{\text{low}}^{(i)}(f) &= X^{(i)}(f)\,H(f), \\ X_{\text{low}}^{(i)}(t) &= \mathcal{F}^{-1}\{X_{\text{low}}^{(i)}(f)\}. \end{aligned} \quad (7)$$

The cutoff frequency f_c can be fixed or learned. This step suppresses high-frequency adversarial noise while leaving the main semantic band largely intact.

5.4 Training Objective and Data

Dataset Construction. We sample 300 original dialogue instances $\mathcal{D}_{\text{original}}$ and 300 adversarial instances $\mathcal{D}_{\text{adv}}$ (covering suffix, rewrite, and substitution attacks), totaling 600 examples that are cycled through in mini-batches.

Table 1: Accuracy (higher is better, %) on three Guard benchmarks under 13 attack types. **LG3** = Llama-Guard-3-8B(Llama Team, 2024), **LG2** = Llama-Guard-2-8B(Team, 2024). "None" = Original model, STSHIELD = reproduced baseline, **Ours** = Our proposed method(Gamma-Guard). ORI = original data with no attack. 100 examples per attack per dataset.

Dataset	Method	Attack Type												
		ILGR↑	GCG↑	TF↑	PWWS↑	GT↑	SPSO↑	BTA↑	BAE↑	FD↑	SEA↑	SCPN↑	GAN↑	ORI↑
BeaverTails	None-LG3	14	18	0	0	0	0	2	0	0	0	0	0	**100**
	None-LG2	28	28	0	0	0	0	0	0	0	0	0	6	**100**
	STSHIELD-LG3	70	67	66	71	74	82	76	78	71	64	75	72	98
	STSHIELD-LG2	69	65	68	70	74	81	73	66	73	61	73	63	96
	Ours-LG3	**100**	**98**	**96**	96	90	**100**	92	94	90	86	96	92	92
	Ours-LG2	84	84	90	**100**	**95**	**100**	**95**	**100**	**100**	**100**	**100**	**93**	94
XSTest	None-LG3	12	12	0	0	0	0	0	0	0	0	0	0	**100**
	None-LG2	18	18	20	15	20	15	25	15	20	0	6	6	**100**
	STSHIELD-LG3	69	61	66	73	76	80	68	64	78	62	74	64	95
	STSHIELD-LG2	64	64	66	68	78	80	78	62	68	63	68	63	97
	Ours-LG3	**96**	90	90	**90**	**100**	**90**	**95**	**95**	**90**	**91**	**91**	**90**	92
	Ours-LG2	93	**93**	**95**	75	85	80	75	85	**90**	80	81	81	94
AdvBench	None-LG3	27	25	33	31	27	34	21	35	33	0	0	0	**100**
	None-LG2	6	6	0	0	0	0	0	0	0	0	0	0	**100**
	STSHIELD-LG3	66	66	72	73	71	80	72	69	77	56	69	58	93
	STSHIELD-LG2	71	67	71	71	69	80	77	64	77	60	74	60	91
	Ours-LG3	**100**	**100**	**88**	**87**	**91**	**93**	**88**	**88**	**88**	80	73	66	92
	Ours-LG2	70	70	80	80	75	73	80	81	80	**82**	**79**	**83**	94

Loss Function. For each example, let z be the logits from the base Guard and $\hat{z}$ the logits after our framework. We minimize the KL divergence at the last token

$$\mathcal{L}_{\text{KL}} = \text{KL}\big[\text{softmax}(\hat{z}_{-1}) \,\big\|\, \text{softmax}(z_{-1})\big],$$

and add L_2 regularization on original inputs:

$$\mathcal{L} = \mathcal{L}_{\text{KL}} + \lambda\|\gamma\|_2^2, \qquad \lambda = 10^{-4}.$$

Only $\{\gamma, W_1, W_2, W_m^{(l)}\}$ are trainable; we use Adam with a base learning rate of 5×10^{-5}.

Parameter Budget and Runtime. With hidden size $H{=}512$ and k layers using attention correction, the extra parameters are $\approx H(D{+}L) + kHL{+}1$, staying below 0.1% of the base model. At inference time we add merely one extra MLP and a few matrix additions, increasing latency by $<2\%$.

6 Results and Analysis

This section introduces the experimental environment, design, and objective analysis of the results.

6.1 Experimental Setup

Datasets. To evaluate Guard performance fairly, we adopt three benchmarks commonly used in related work: PKU-ALIGNMENT/BEAVERTAILS (Ji et al., 2023), WALLEDAI/XSTEST (Röttger et al., 2023), and WALLEDAI/ADVBENCH (Zou et al., 2023b).

Attack Methods. Because no public dataset specifically targets Guard models, we reproduce 12 adversarial methods strictly following their original descriptions: ILGR (Raina et al., 2024), GCG (Zou et al., 2023a), TF (Jin et al., 2020), PWWS (Ren et al., 2019), GT (Alzantot et al., 2018), SPSO (Zang et al., 2020), BTA (Li et al., 2020), BAE (Garg and Ramakrishnan, 2020), FD (Papernot et al., 2016), SEA (Ribeiro et al., 2018), SCPN (Iyyer et al., 2018), and GAN (Zhao et al., 2018). The untouched benchmark is denoted ORI.

Baselines. No prior study tackles LLM Guard robustness, making Gamma-Guard the first. As a baseline we re-implemented STSHIELD(Wang et al., 2025a), originally built for raw LLM Chat, so some deviation is expected.

Metric. We report accuracy (*Acc*), the standard metric in Guard evaluation. Labels are taken from Llama-Guard-3-8B (Llama Team, 2024), the OpenAI moderation endpoint, and human verification, The specific approach is that if the results of the three are the same, we will consider the results credible.

Training the Residual Branch. As described in Section 5, we train the residual branch once per base model using 600 examples: 50 % original and 50 % adversarially modified from the three datasets

above. After training, the branch parameters are *frozen*, ensuring generalization during all subsequent tests.

6.2 Benchmark Results and Discussion

Overall robustness. Table 1 shows that **Gamma-Guard** almost eliminates jailbreaks. Averaging the 12 adversarial columns (ORI excluded) yields macro robust accuracy $\mathrm{Acc}_{\mathrm{rob}}$ of **94.2 %**, **92.3 %**, and **86.8 %** on BeaverTails, XSTest, and AdvBench, respectively, when plugged into Llama-Guard-3-8B (LG3). In stark contrast, the original guards collapse to $\leq 3\,\%$, and the reproduced STSHIELD plateaus around $70\,\%$. Hence, Gamma-Guard closes a ~ 90-point robustness gap while adding $< 0.1\,\%$ parameters, $\approx 2\,\%$ latency, and only an 8-pp drop on original accuracy.

Dataset-level detail. **BeaverTails**: 11/12 attacks reach $\geq 90\,\%$ (five hit $100\,\%$). **XSTest**: all attacks stay at or above $90\,\%$, except a marginal dip for SEA. **AdvBench**: eight attacks exceed $88\,\%$; sentence-rewrite methods SCPN and GAN remain hardest (73 % and 66 %), highlighting discourse-level edits as the next frontier.

Attack-type trends. Character-level perturbations (ILGR/GCG) are nearly neutralised (98–100 % on LG3); gradient- or score-based attacks (FD/GT) stabilize around 90 %; sentence-level rewrites remain the main weakness.

Effect of model size. With the smaller LG2 backbone, Gamma-Guard still lifts $\mathrm{Acc}_{\mathrm{rob}}$ to **84.4 %** on BeaverTails and **77.8 %** on XSTest, shrinking the LG2–LG3 gap from $> 22\%$ (under STSHIELD) to $< 7\%$. LG2 even surpasses LG3 on several attacks (e.g., SEA/SCPN in BeaverTails), indicating that capacity-limited models benefit most from the residual adapter.

6.3 Ablation Study

As stated in Section 5, injecting a residual into *every* attention layer of Llama-Guard-3 (32 layers total) would inflate compute and parameters. We therefore treat the interval k—how often to add a residual—as a tunable hyper-parameter and conduct an ablation on Llama-Guard-3-8B:

- **Embedding-level denoising always on.**
- **Attention residual interval k:** inject only when layer $\bmod k = 0$, with $k \in \{0, 1, 5, 10, 15\}$. k=0 disables attention correction entirely.

Table 2: Accuracy (%, higher is better) on BeaverTails when attention-correction residuals are inserted every k layers.

Attack	k=0	k=1	k=5	k=10	k=15
ILGR	70	98	98	98	98
GCG	72	94	94	90	90
TF	66	98	98	86	94
PWWS	66	98	98	94	86
GT	58	94	96	90	94
ORI	94	90	92	90	90

Setup We use BeaverTails and report both *original* accuracy and *attack* resistance; all other hyper-parameters remain fixed.

Results & Discussion Table 2 shows that:

- k=0 (embedding fix only): almost no impact on original data but weakest defense.
- k=1: strongest robustness yet largest drop on original accuracy.
- k=5: the best compromise—huge robustness gains with only a slight original-accuracy dip.
- k=10, 15: defenses weaken again, indicating overly sparse residuals cannot fully suppress perturbations.

Overall, an interval of k=5 yields the best balance between robustness and fidelity. The optimal k may vary by model or dataset, but similar ablations can always locate a sweet spot of "high robustness with minimal accuracy loss."

7 Conclusion

In sum, we introduce Gamma-Guard—the first method that markedly strengthens adversarial robustness for guardrails within large-language-model pipelines—without sacrificing efficiency. This advance rests on three key findings: (i) pilot studies pinpoint that production guards fail when early-layer noise cascades into the decision head; (ii) a lightweight, learnable residual branch, trained on only a handful of adversarial samples, can simultaneously denoise embeddings and refocus attention; and (iii) comprehensive experiments across datasets and attack suites confirm large robustness gains with only a single-digit drop in original accuracy and negligible runtime overhead. Together, these results chart a practical path toward safer, large-scale LLM deployments.

8 Limitations

Although our residual branch markedly improves Guard robustness, it does not yet achieve a near-perfect 99% success rate, suggesting room for stronger architectures. Moreover, the branch still causes a small accuracy drop on benign inputs; future work should minimize or eliminate that trade-off. Finally, our evaluation covers the most widely used public Guard benchmarks, but some private datasets remain inaccessible; testing on such closed-source data would provide a more complete picture of real-world performance.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, and et al. 2021. Rezero is all you need: Fast convergence at large depth. In *ICLR*.

Zhiyang Cai, Yonglong Tian, Piotr Dollár, and et al. 2021. Layerscale: Scaling residual branches for deeper transformers. In *NeurIPS*.

Shuo Chen, Jindong Gu, Zhen Han, and et al. 2023a. Benchmarking robustness of adaptation methods on pre-trained vision-language models. In *NeurIPS D&B Track*.

Yi Chen, Rui Wang, Haiyun Jiang, Shuming Shi, and Ruifeng Xu. 2023b. Exploring the use of large language models for reference-free text quality evaluation: An empirical study. In *Findings of the Association for Computational Linguistics: IJCNLP–AACL 2023*, pages 361–374, Bali,Indonesia. Association for Computational Linguistics.

Jianfeng Chi, Ujjwal Karn, Hongyuan Zhan, and et al. 2024a. Llamaguard 3 vision: Safeguarding human–ai image understanding conversations. *arXiv preprint arXiv:2411.10414*.

Jianfeng Chi, Ujjwal Karn, Hongyuan Zhan, Eric Smith, Javier Rando, and 1 others. 2024b. Llama guard 3 vision: Safeguarding human–ai image understanding conversations. *arXiv preprint arXiv:2411.10414*.

OWASP Community. 2025. Owasp top 10 for large language model applications 2025. https://owasp.org/www-project-top-10-for-large-language-model-applications/.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Jindong Gu, Shuo Chen, and Philip Torr. 2024. Hyper adversarial tuning for boosting adversarial robustness of large models. *CVPR*.

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, and 1 others. 2025. Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*.

William Hackett, Lewis Birch, Stefan Trawicki, and et al. 2025a. Bypassing prompt injection and jailbreak detection in llm guardrails. *arXiv preprint arXiv:2504.11168*.

William Hackett, Lewis Birch, Stefan Trawicki, Neeraj Suri, and Peter Garraghan. 2025b. Bypassing prompt injection and jailbreak detection in llm guardrails. *arXiv preprint arXiv:2504.11168*.

Qian Han, Feiyu Li, Zhen Wang, and et al. 2024a. Uniguard: Towards universal safety guardrails for jailbreak attacks. *arXiv preprint arXiv:2411.01703*.

Seungju Han, Kavel Rao, Allyson Ettinger, and et al. 2024b. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *arXiv preprint arXiv:2406.18495*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, and et al. 2019. Parameter-efficient transfer learning for nlp. In *ICML*.

Edward J. Hu, Yelong Shen, Phillip Wallis, and et al. 2022. Lora: Low-rank adaptation of large language models. In *ICLR*.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, and et al. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Chi Zhang, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Preprint*, arXiv:2307.04657.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *Preprint*, arXiv:1907.11932.

Seanie Lee, Dong Bok Lee, Dominik Wagner, Minki Kang, Haebin Seong, Tobias Bocklet, Juho Lee, and Sung Ju Hwang. 2025a. Saferoute: Adaptive model selection for efficient and accurate safety guardrails in large language models. *arXiv preprint arXiv:2502.12464*.

Seanie Lee, Dong Bok Lee, Dominik Wagner, and et al. 2025b. Saferoute: Adaptive model selection for efficient and accurate safety guardrails. *arXiv preprint arXiv:2502.12464*.

Jingkai Li. 2025. Llm-as-a-judge:a complete guide to using llms for evaluations. https://www.evidentlyai.com/llm-guide/llm-as-a-judge. Blog post, accessed 6 May 2025.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. *Preprint*, arXiv:2004.09984.

Qing Liu, Yifan Zhang, Yulu Xu, and et al. 2024. Jailbreak vision-language models via bi-modal adversarial prompt. *arXiv preprint arXiv:2406.04031*.

AI @ Meta Llama Team. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Meta AI. 2023. Announcing purple llama: Towards open trust and safety in generative ai. https://ai.meta.com/blog/purple-llama-open-trust-safety-generative-ai/.

Meta AI. 2025. Llamaguard 4-12b model card. https://huggingface.co/meta-llama/Llama-Guard-4-12B.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. *Preprint*, arXiv:1604.08275.

Vyas Raina, Adian Liusie, and Mark Gales. 2024. Is llm-as-a-judge robust? investigating universal adversarial attacks on zero-shot llm assessment. *Preprint*, arXiv:2402.14016.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2023. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. *arXiv preprint arXiv:2308.01263*.

Mrinank Sharma, Meg Tong, Jesse Mu, and et al. 2025a. Constitutional classifiers: Defending against universal jailbreaks across thousands of hours of red teaming. *arXiv preprint arXiv:2501.18837*.

Mrinank Sharma, Meg Tong, Jesse Mu, Jerry Wei, Jorrit Kruthoff, Scott Goodfriend, Euan Ong, Alwin Peng, Raj Agarwal, and 1 others. 2025b. Constitutional classifiers: Defending against universal jailbreaks across thousands of hours of red teaming. *arXiv preprint arXiv:2501.18837*.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. 2024a. Judging the judges: A systematic study of position bias in llm-as-a-judge. *arXiv preprint arXiv:2406.07791*.

Yefei Shi, Zhi Wang, Yongtao Liu, and et al. 2024b. Goal-guided generative prompt injection attack on large language models. *arXiv preprint arXiv:2404.07234*.

Llama Team. 2024. Meta llama guard 2. https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md.

Hugo Touvron, Piotr Bojanowski, Matthieu Cord, and et al. 2021. Going deeper with image transformers. In *ICCV*.

Xunguang Wang, Wenxuan Wang, Zhenlan Ji, and et al. 2025a. Stshield: Single-token sentinel for real-time jailbreak detection in large language models. *arXiv preprint arXiv:2503.17932*.

Xunguang Wang, Wenxuan Wang, Zhenlan Ji, Zongjie Li, Pingchuan Ma, Daoyuan Wu, and Shuai Wang. 2025b. Stshield: Single-token sentinel for real-time jailbreak detection in large language models. *arXiv preprint arXiv:2503.17932*.

Yue Wang, Jianbo Liu, and Xiaohua Zhai. 2024. Adapters strike back: Layer-wise scaling benefits robust vision models. *IEEE TPAMI*.

Haitao Wei, Xin Xu, Hao Wang, and et al. 2024. Prp: Propagating universal perturbations to attack large language models. In *Proceedings of ACL 2024*.

Yifan Xu, Xiaoyu Liu, Baolin Peng, and et al. 2024. Evaluating the instruction-following robustness of large language models. In *Proceedings of EMNLP 2024*.

Zonghao Ying, Guangyi Zheng, Yongxin Huang, Deyue Zhang, and 1 others. 2025. Towards understanding the safety boundaries of deepseek models: Evaluation and findings. *arXiv preprint arXiv:2503.15092*.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Chenhao Zhang, Xudong Li, Rui Zhang, and et al. 2025. Guardrail identification in large language models. *arXiv preprint arXiv:2502.01241*.

Xinghua Zhang, Bowen Yu, Haiyang Yu, Yangyu Lv, Tingwen Liu, Fei Huang, Hongbo Xu, and Yongbin Li. 2023. Wider and deeper llm networks are fairer llm evaluators. *arXiv preprint arXiv:2308.01862*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. *Preprint*, arXiv:1710.11342.

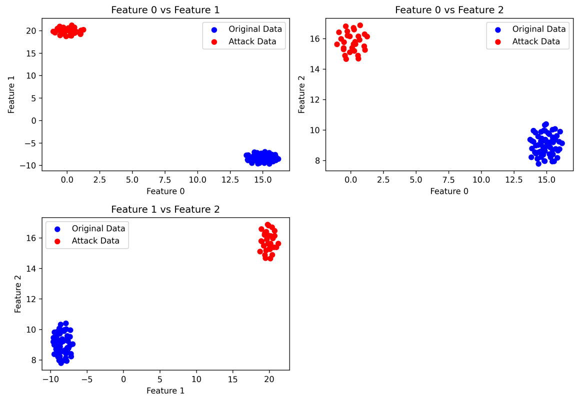
Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

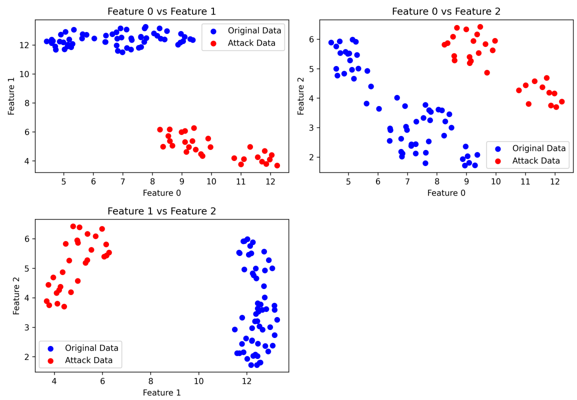
Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023a. Universal and transferable adversarial attacks on aligned language models. *Preprint*, arXiv:2307.15043.

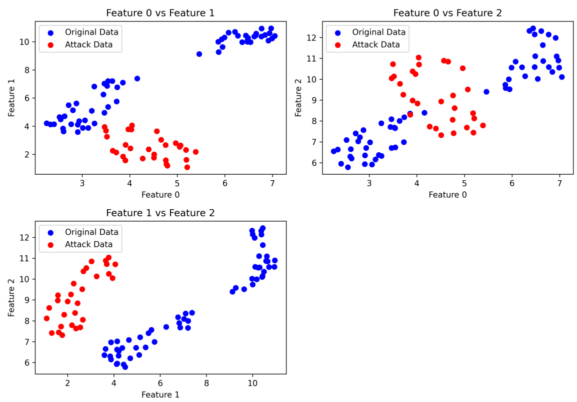
Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023b. Universal and transferable adversarial attacks on aligned language models. *Preprint*, arXiv:2307.15043.

A Adversarial-Attack Experiments

This section presents the detailed feature-analysis results that verify the vulnerability of Guard models. Each figure corresponds to a different sample; please refer to the main text for a full discussion.

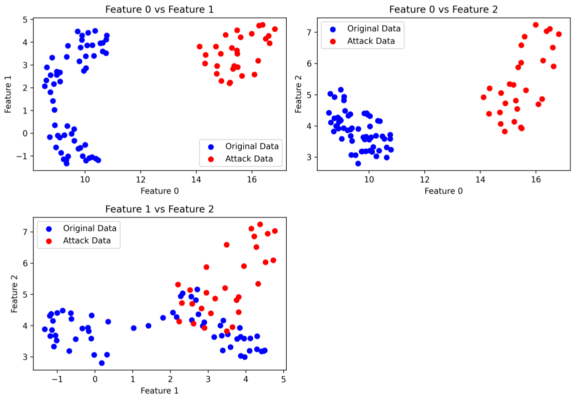


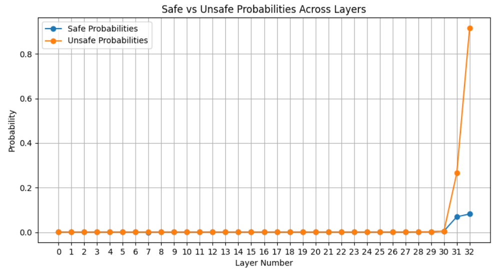


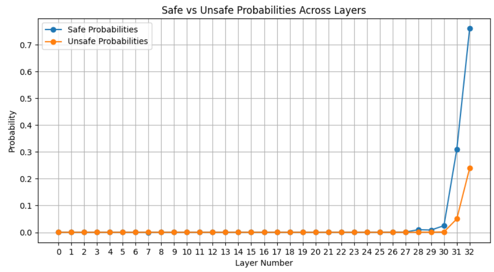


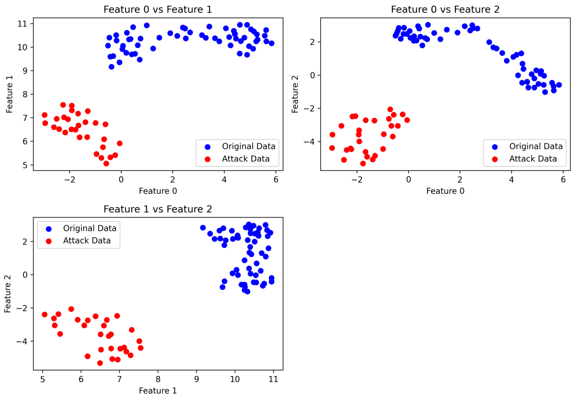
B Layer-wise Location Experiments

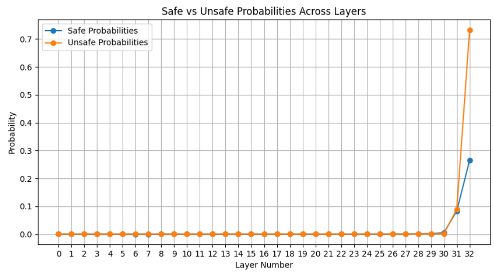
This section provides the full results used to pinpoint *where* adversarial shifts occur within the model. For each question ID we show the original (top) and successfully attacked (bottom) inputs.

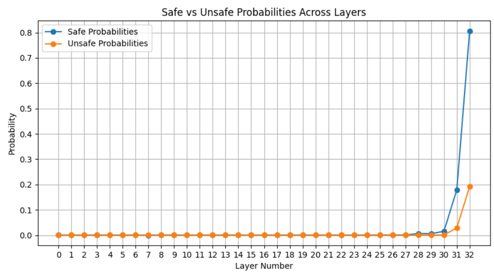

Feature 0 vs Feature 1
Feature 0 vs Feature 2
Feature 1 vs Feature 2
Original Data
Attack Data
Feature 0
Feature 1
Feature 2

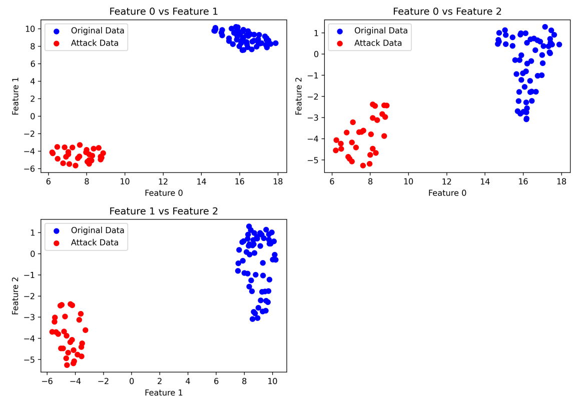

Safe vs Unsafe Probabilities Across Layers
Safe Probabilities
Unsafe Probabilities
Probability
Layer Number

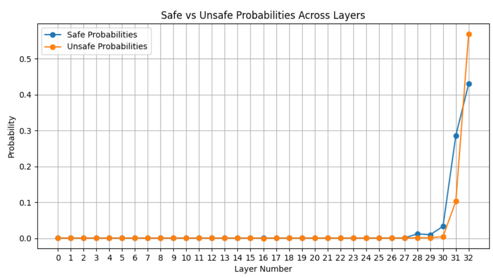

Safe vs Unsafe Probabilities Across Layers
Safe Probabilities
Unsafe Probabilities
Probability
Layer Number

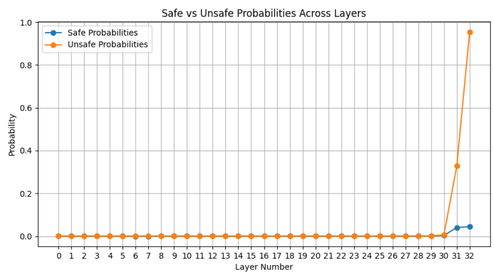

Feature 0 vs Feature 1
Feature 0 vs Feature 2
Feature 1 vs Feature 2
Original Data
Attack Data
Feature 0
Feature 1
Feature 2

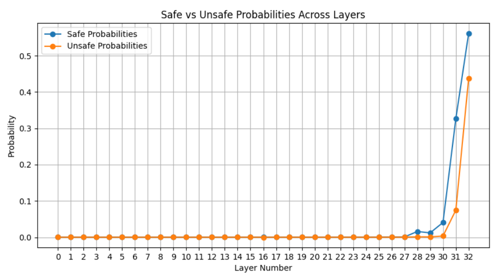

Safe vs Unsafe Probabilities Across Layers
Safe Probabilities
Unsafe Probabilities
Probability
Layer Number

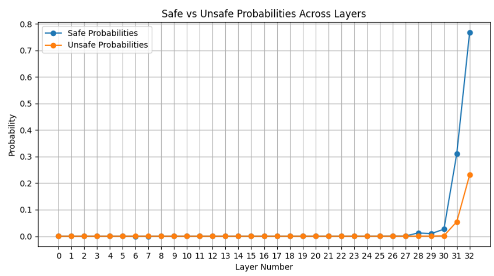

Safe vs Unsafe Probabilities Across Layers
Safe Probabilities
Unsafe Probabilities
Probability
Layer Number


Feature 0 vs Feature 1
Feature 0 vs Feature 2
Feature 1 vs Feature 2
Original Data
Attack Data
Feature 0
Feature 1
Feature 2


Safe vs Unsafe Probabilities Across Layers
Safe Probabilities
Unsafe Probabilities
Probability
Layer Number


Safe vs Unsafe Probabilities Across Layers
Safe Probabilities
Unsafe Probabilities
Probability
Layer Number


Safe vs Unsafe Probabilities Across Layers
Safe Probabilities
Unsafe Probabilities
Probability
Layer Number


Safe vs Unsafe Probabilities Across Layers
Safe Probabilities
Unsafe Probabilities
Probability
Layer Number

C Attention Analysis

This section presents the attention-heat-map analysis underlying our error diagnosis. See the main text for interpretation of token-level focus differences between original and adversarial inputs.