NeuronGuard: Robust LLM Safety Alignment via Ablation-Aware Safety Signal Redistribution

Anjun Gao[1] **Yueyang Quan**[2] **Yufei Xia**[1*] **Zhuqing Liu**[2] **Minghong Fang**[1]
[1]University of Louisville, [2]University of North Texas

Abstract

Safety alignment in large language models (LLMs) remains brittle against a growing spectrum of attacks. Jailbreak attacks bypass safety mechanisms through crafted prompts, while neuron-level attacks directly prune safety-critical neurons post-deployment. Both exploit a common weakness: safety-relevant information concentrates in a sparse neuron subset. We present `NeuronGuard`, a fine-tuning-stage defense that simultaneously hardens LLMs against both attack classes by redistributing safety signals across a broader set of neurons. `NeuronGuard` dynamically identifies safety-critical neurons via periodically refreshed per-layer linear classifiers, forces refusal behavior under deliberate neuron ablation, and applies KL-divergence regularization for distributional consistency. A randomized gradient projection strategy preserves downstream task utility by resolving conflicts between the defense and task objectives. We provide a formal guarantee that `NeuronGuard` strictly reduces the attack success rate (ASR) upper bound, and experiments across three LLMs, six state-of-the-art attack strategies, and multimodal settings confirm near-zero ASR while maintaining task accuracy, including against white-box adaptive adversaries.

1 Introduction

Large language models (LLMs) (Team, 2024a; Grattafiori et al., 2024; Team, 2024b; Team et al., 2024; Abdin et al., 2024; Brown et al., 2020; Team, 2026; Blakeman et al., 2025) have demonstrated remarkable capabilities across diverse applications, from question answering and code generation to complex reasoning and planning. Beyond these foundational tasks, LLMs have been increasingly deployed as general-purpose intelligence components powering autonomous agents, dialogue systems, and tool-augmented workflows. This broad and rapid adoption has made the safety and robustness of LLMs a critical concern, particularly given the potential consequences of deploying misaligned or manipulable models in high-stakes domains.

To ensure safe deployment, these models undergo extensive alignment procedures, including supervised fine-tuning (Ouyang et al., 2022), reinforcement learning from human feedback (Bai et al., 2022), and preference optimization (Rafailov et al., 2023) to instill refusal behavior against harmful requests. However, this alignment has proven fragile across a growing spectrum of post-alignment attacks. Jailbreak attacks (Chao et al., 2025; Mehrotra et al., 2024; Chang et al., 2024; Zou et al., 2023; Liu et al., 2024) bypass safety mechanisms through crafted prompts, adversarial suffixes, or role-playing scenarios without modifying model parameters. At a deeper level, neuron-level attacks (Wu et al., 2026) strike at the internal structure of aligned models: prior work has revealed that safety alignment tends to concentrate in a sparse set of specialized neurons, and NeuroStrike (Wu et al., 2026) exploits this by identifying and pruning safety-critical neurons at inference time, disabling safety mechanisms without any adversarial prompt. Both attack classes share a common root cause, namely that safety-relevant information concentrates in a sparse neuron subset, yet they exploit it from orthogonal directions: one through the input and one through the model's parameters.

Existing defenses are inadequate against this dual threat. Prompt-level defenses such as perplexity filtering (Alon and Kamfonas, 2023), SmoothLLM (Robey et al., 2023), and GradSafe (Xie et al., 2024) are entirely ineffective against attacks that operate at the parameter level. Existing model-level defenses, including CAT (Xhonneux et al., 2024) and LED (Zhao et al., 2024), offer more direct protection but remain vulnerable to direct neuron manipulation. SafeNeuron (Wang et al., 2026) comes closest by freezing safety-critical neurons during

*Yufei Xia performed this research when he was under the supervision of Minghong Fang.

optimization, yet its *static* identification strategy cannot account for representational shifts as model weights evolve. More fundamentally, none of these approaches addresses the root cause: safety information remains concentrated in a sparse neuron subset, leaving the model vulnerable to any adversary who can identify and suppress it.

Our contributions: To address these challenges, we present NeuronGuard, a fine-tuning-stage defense framework that hardens LLMs against both jailbreak and neuron-level attacks by redistributing safety signals across a broader set of neurons. Our method builds on a key observation: the brittleness of current alignment stems from over-concentration of safety-relevant information in a sparse neuron subset, and robustness can be improved by training the model to maintain safe behavior even when primary safety neurons are suppressed. To identify target neurons, NeuronGuard fits lightweight per-layer linear classifiers that separate harmful from benign representations at each fine-tuning step; these classifiers are periodically refreshed to track the *current* safety-critical neuron set as weights evolve. Building on this, NeuronGuard constructs a safety objective that deliberately ablates the identified neurons and requires the model to refuse harmful queries in their absence, forcing gradient updates into the remaining neurons and progressively broadcasting safety representations across the network. A KL-divergence regularizer constrains the ablated model's output to match the standard forward pass, ensuring behavioral consistency. As safety signals spread to new neurons, those neurons are identified and ablated in turn, continuously expanding representational redundancy. Finally, since safety and task objectives frequently produce conflicting gradients, NeuronGuard applies a randomized projection that removes the opposing component from one gradient without systematically favoring either objective, preserving utility without sacrificing robustness.

We conduct evaluations across diverse attack settings, demonstrating that NeuronGuard effectively defends against both jailbreak attacks and neuron-level attacks while maintaining model utility on downstream fine-tuning tasks. We evaluate against six representative attack strategies, assess robustness in multimodal settings, and include challenging adaptive adversaries with full knowledge of our defense. Across all settings, NeuronGuard consistently reduces the attack success rate (ASR) to near zero while preserving task accuracy, with modest computational overhead suitable for practical fine-tuning pipelines.

Our contributions are summarized as follows:

- We propose NeuronGuard, a novel fine-tuning-stage defense framework that dynamically identifies safety-critical neurons via periodically refreshed per-layer linear classifiers, and broadcasts safety signals across a broader set of neurons through ablation-robust optimization and randomized gradient projection, providing simultaneous resistance to both jailbreak attacks and neuron-level attacks.
- We provide a formal theoretical guarantee showing that NeuronGuard strictly reduces the ASR upper bound relative to an undefended model under neuron-level attack, with the reduction quantified in terms of the ablation coverage ratio, per-neuron attack success probability, and the KL-divergence regularization strength.
- Comprehensive evaluations across diverse attack strategies and deployment scenarios demonstrate that NeuronGuard effectively hardens safety alignment while preserving model utility, including under adaptive adversaries explicitly designed to evade our defense.

2 Related work

Post-alignment attacks on LLMs: Post-alignment attacks are inference-time adversarial strategies that bypass safety alignment in deployed LLMs. We focus on two representative families: jailbreak attacks and neuron-level attacks. Jailbreak attacks subvert safety alignment through crafted inputs without modifying model parameters. Generation-based methods such as PAIR (Chao et al., 2025), TAP (Mehrotra et al., 2024), and Puzzler (Chang et al., 2024) synthesize adversarial prompts via iterative querying or puzzle-like reformulations. Optimization-based methods take a more direct route: GCG (Zou et al., 2023) searches over adversarial token suffixes via gradient guidance, while AutoDAN (Liu et al., 2024) combines gradient signals with fluent prompt generation. Rather than manipulating inputs, neuron-level attacks exploit the internal structure of aligned models directly. Safety alignment is concentrated in a sparse set of specialized neurons (Wu et al., 2026), making it both localizable and fragile. NeuroStrike (Wu et al., 2026) compromises an aligned model by locating neurons responsible for refusal behavior and suppressing their activations during inference, thereby

inducing unsafe outputs without requiring specially crafted inputs.

Defenses against post-alignment attacks: Existing defenses fall into two categories, each with fundamental limitations. *Prompt-level defenses* intercept adversarial inputs at inference time without altering the model: Perplexity filtering (Alon and Kamfonas, 2023; Cheng et al., 2025) flags anomalous inputs, SmoothLLM (Robey et al., 2023) aggregates responses over randomly perturbed inputs via majority voting, and GradSafe (Xie et al., 2024) detects jailbreak attempts through gradient patterns on safety-critical parameters. However, these defenses are inherently ineffective against neuron-level attacks, which operate at the parameter level and require no adversarial prompt. *Model-level defenses* modify the model's parameters or training procedure, offering stronger but still insufficient protection. CAT (Xhonneux et al., 2024) performs adversarial training in the continuous embedding space, and LED (Zhao et al., 2024) edits safety-critical layers via hidden-state activation analysis; both target prompt-based threats and offer little resistance to direct neuron manipulation. SafeNeuron (Wang et al., 2026) freezes safety-critical neurons during direct preference optimization to encourage redundant safety pathways, but its static identification strategy cannot capture representation shifts during weight updates. Fundamentally, none of these approaches trains the model to distribute safety signals broadly, which is the root cause of their fragility.

Note that a complementary line of research has investigated post-attack forensic attribution, which aims to determine how a successful attack was carried out and identify the source or root cause responsible for its success (Jia et al., 2025; Gao et al., 2026b; Zhang et al., 2025, 2026; Gao et al., 2026a). These techniques provide valuable information for diagnosing compromised systems and understanding the origins of successful attacks, thereby complementing conventional defense mechanisms. Nevertheless, post-attack forensics addresses a different stage of the attack lifecycle from the mitigation strategies considered in this work. Our study instead focuses on preventing and mitigating attacks, while forensic investigation and attribution after a successful attack are beyond the scope of this paper.

3 Threat model

Attacker's goal and knowledge: We consider post-alignment attacks that aim to bypass a model's safety alignment and elicit harmful responses. This framing is consistent with both jailbreak attacks (Chao et al., 2025; Mehrotra et al., 2024; Chang et al., 2024; Zou et al., 2023; Liu et al., 2024) and neuron-level attacks (Wu et al., 2026), which are inherently post-hoc. We further assume a worst-case setting where the attacker has full access to the post-finetuning weights and can manipulate internal parameters or neurons, a realistic assumption for open-weight LLMs whose parameters are publicly available.

Defender's goal and knowledge: The defender aims to maintain safety alignment under post-alignment attacks, ensuring consistent refusal of harmful queries. We assume the defender operates during fine-tuning with full access to the training pipeline, including data, weights, and objectives. This is natural: the developer who conducts fine-tuning is directly responsible for the safety properties of the released model. Critically, fine-tuning is the last stage under the developer's control before public deployment, as once released, the developer cannot control how weights are accessed or modified. Hardening the model at this stage therefore proactively reduces the attack surface, without requiring any intervention at inference time or any assumptions about the attacker's specific strategy. Following prior work (Wu et al., 2026; Wang et al., 2026), we assume the defender holds a small safety probe set of harmful and benign queries. This assumption is reasonable as any responsible model developer would already maintain a small set of safety-critical queries as part of standard quality control before deployment, and such a probe set requires no knowledge of the attacker's specific strategy.

4 Our method

`NeuronGuard` addresses the root cause of alignment fragility through three coupled stages, as illustrated in Fig. 2 (Appendix). The first stage fits per-layer linear classifiers to identify the sparse neuron subset dominating safety-gating decisions, refreshed periodically as weights evolve. The second stage deliberately suppresses these critical neurons during training, forcing safety representations to redistribute across a broader population so that no small subset serves as a single point of failure.

The third stage resolves conflicts between the defense and task objectives via randomized gradient projection, preserving utility without sacrificing robustness. The complete procedure is summarized in Algorithm 1 (Appendix).

4.1 Stage 1: Identifying safety-critical neurons

Safety signals must be redistributed away from the neurons that currently dominate the model's refusal behavior. The first procedure is therefore to identify those neurons precisely at each point in fine-tuning. We achieve this by fitting a lightweight linear classifier at each layer that separates harmful from benign representations, and then reading off which neurons contribute most to that separation.

Concretely, the defender maintains a safety probe set consisting of two small subsets: a harmful query set D_{unsafe} and a benign query set D_{safe}. Each prompt $x \in D_{\text{safe}} \cup D_{\text{unsafe}}$ is assigned a binary label ($y = 1$ for harmful, $y = 0$ for benign). We then feed x through the current model and collect the activation vectors $h_l(x)$ at each layer l, and a per-layer logistic classifier is trained on these activations:

$$\hat{y}_l(x) = \sigma(W_l^\top h_l(x) + b_l), \quad \forall\, x \in D_{\text{safe}} \cup D_{\text{unsafe}}, \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function and W_l, b_l are the learnable classifier parameters. The weight vector W_l encodes neuron-level importance: a large positive weight $W_{l,i}$ indicates that neuron i is a strong predictor of the "unsafe" label, and hence a primary carrier of safety-relevant information. We therefore rank neurons by their corresponding classifier weights and select the top-ρ fraction as the safety-critical set for layer l:

$$\mathcal{I}_{\text{safety},l} = \text{TopFrac}(W_l,\ \rho)\,, \tag{2}$$

where $\text{TopFrac}(\cdot, \rho)$ returns the indices of the largest $\rho \cdot d_l$ elements, and d_l is the number of neurons in layer l. Because model representations shift as weights are updated, a fixed neuron set identified at initialization quickly becomes stale. We therefore refresh the classifiers, and recompute $\mathcal{I}_{\text{safety},l}$, every N optimization steps. This periodic re-identification ensures that Stage 2 always targets the *current* safety-critical neurons, not an outdated approximation, and is the mechanism that produces the iterative broadening effect described next.

4.2 Stage 2: Ablation-robust safety optimization

With the current safety-critical neuron sets $\{\mathcal{I}_{\text{safety},l}\}$ in hand, we construct a safety objective that drives the model to spread safety representations beyond those neurons. The objective has three terms, each enforcing a distinct aspect of robust safety.

Refusal loss under normal execution: The first term ensures the model refuses harmful queries during normal inference. Let $x_{\text{h}} \in D_{\text{unsafe}}$ denote a harmful input and y_{ref} a fixed refusal-template target (e.g., "I'm sorry, but I cannot assist with that request."). We apply standard cross-entropy supervision:

$$L_{\text{ref}} = \text{CE}(y_{\text{ref}},\ p_\theta(y \mid x_{\text{h}}))\,, \tag{3}$$

where $p_\theta(y \mid x_{\text{h}})$ is the model's output distribution under parameters θ. This term provides a baseline safety signal, but on its own, only reinforces the neurons that already encode refusal behavior. Safety information, therefore, remains concentrated in $\mathcal{I}_{\text{safety},l}$, leaving the model vulnerable whenever those neurons are suppressed. The next term addresses this limitation directly.

Refusal loss under safety-neuron ablation: To compel other neurons to take on safety responsibilities, we introduce a second supervision term that operates on a *modified* forward pass in which the safety-critical neurons are deliberately zeroed out. Specifically, for each layer l, we construct ablated activations $\tilde{h}_{i,l} = 0$ if $i \in \mathcal{I}_{\text{safety},l}$, and $\tilde{h}_{i,l} = h_{i,l}$ otherwise, then propagate them through the remaining layers to obtain the ablated output distribution $p_{\theta_{\text{abl}}}(y \mid x_{\text{h}})$. We then apply refusal supervision on this ablated forward pass:

$$L_{\text{ref}}^{\text{abl}} = \text{CE}(y_{\text{ref}},\ p_{\theta_{\text{abl}}}(y \mid x_{\text{h}}))\,. \tag{4}$$

Because the neurons in $\mathcal{I}_{\text{safety},l}$ are zeroed out, gradients from $L_{\text{ref}}^{\text{abl}}$ cannot flow back into them and must instead propagate to the remaining neurons, training them to carry safety-relevant information. As these neurons absorb safety information over successive steps, the refreshed classifiers identify them as part of the new safety-critical set and ablate them in turn, pushing the safety signal progressively outward to an ever-wider set of neurons.

Distributional consistency regularization: Optimizing L_{ref} and $L_{\text{ref}}^{\text{abl}}$ in isolation does not prevent the ablated model from producing correct refusal

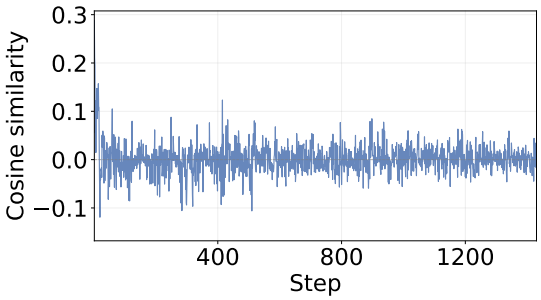

Figure 1: Cosine similarity between g_{user} and g_{safe} across fine-tuning steps on SST2 (Socher et al., 2013) with Llama.

tokens while diverging sharply on the remaining vocabulary, leading to unstable training dynamics. To prevent this, we add a KL-divergence regularizer between the two forward passes on the same harmful input:

$$L_{\text{cons}} = D_{\text{KL}}(p_{\theta_{\text{abl}}}(y \mid x_{\text{h}}) \parallel p_{\theta}(y \mid x_{\text{h}})). \quad (5)$$

This term enforces that ablating safety neurons preserves the model's overall output distribution, not just the refusal token, preventing degenerate collapse.

Full safety objective: Combining the three terms, the complete safety objective is:

$$L_{\text{safe}} = L_{\text{ref}} + L_{\text{ref}}^{\text{abl}} + L_{\text{cons}}. \quad (6)$$

L_{ref} ensures that the model refuses harmful queries in standard setting; $L_{\text{ref}}^{\text{abl}}$ forces safety representations to spread beyond the identified safety neurons; and L_{cons} maintains distributional consistency between the two settings. Together, these three terms promote distributed safety encoding across neurons, avoiding concentration in only a few neurons.

4.3 Stage 3: Resolving gradient conflicts via randomized projection

Beyond the safety objective, the model must simultaneously learn the user's downstream task. Let L_{user} denote the task loss computed on the user-provided fine-tuning dataset D_{user}. The overall fine-tuning objective is:

$$L_{\text{total}} = L_{\text{user}} + L_{\text{safe}}, \quad (7)$$

where L_{safe} is the safety objective defined in Eq. (6). However, naively minimizing L_{total} with a single gradient step is problematic in practice. Let $g_{\text{user}} = \nabla_{\theta} L_{\text{user}}$ and $g_{\text{safe}} = \nabla_{\theta} L_{\text{safe}}$ denote the gradients of the two objectives. As shown in Figure 1, in a large fraction of fine-tuning steps we observe $\langle g_{\text{user}}, g_{\text{safe}} \rangle < 0$, meaning the two gradients point in conflicting directions. Directly summing them would therefore cause destructive interference: progress on one objective would come at the expense of the other, ultimately harming task utility.

To address this, we introduce a randomized gradient correction that resolves conflicts at each fine-tuning step without systematically favoring either objective. At each step, we first compute $\langle g_{\text{user}}, g_{\text{safe}} \rangle$. If no conflict is detected, both gradients are used unchanged. If a conflict is detected, we randomly sample an ordering (g_1, g_2) of $(g_{\text{user}}, g_{\text{safe}})$, where g_1 is the gradient to be corrected and g_2 is kept unchanged. The corrected gradient is:

$$\hat{g}_1 = g_1 - \frac{\langle g_1, g_2 \rangle}{\|g_2\|_2^2} g_2, \qquad \hat{g}_2 = g_2, \quad (8)$$

which removes from g_1 precisely the component that opposes g_2, while leaving all non-conflicting components intact. The projection is applied to only one gradient rather than both, to avoid unnecessarily shrinking the effective update magnitude. Randomizing which gradient serves as g_1 ensures that neither objective is systematically deprioritized across fine-tuning. When no conflict is detected, we simply set $\hat{g}_1 = g_{\text{user}}$ and $\hat{g}_2 = g_{\text{safe}}$.

The final gradient update is then:

$$g_{\text{final}} = \hat{g}_1 + \hat{g}_2. \quad (9)$$

This ensures that the two objectives always make progress in mutually non-opposing directions, preserving both task utility and the safety robustness gained in Stage 2. The model parameters are then updated as $\theta \leftarrow \theta - \gamma \cdot g_{\text{final}}$, where γ is the learning rate.

5 Theoretical analysis

We provide theoretical guarantees for our proposed `NeuronGuard` under neuron-level attack (which is more powerful than jailbreak attacks based on our experimental observations), answering whether safety-neuron ablation provably reduces ASR and under what conditions. Guarantees are stated in terms of measurable quantities (classifier accuracy, ablation coverage, and KL divergence) estimable from fine-tuning logs. Let $p_{\theta}(\cdot \mid x)$ denote the model's output distribution and $h_{i,l}(x)$ the activation of neuron i at layer l. The ground-truth safety-neuron set at layer l is $S_l = \{ i \mid \mathbb{E}_{x \sim \mathcal{D}_{\text{unsafe}}}[h_{i,l}(x)] - \mathbb{E}_{x \sim \mathcal{D}_{\text{safe}}}[h_{i,l}(x)] > \tau, \ i \in$

$[1, d_l]\}$, where $\tau > 0$ and $|S_l| = s_l \ll d_l$. The ablation coverage ratio is defined as $\rho_l := |S_l \cap \mathcal{I}_{\text{safety},l}|/|S_l|$. We first provide the necessary assumptions.

Assumption 1 (Approximate linear separability and sufficient ablation coverage)**.** *There exists a linear classifier $\psi = (W_l, b_l)$ with $\Upsilon(h_l(x); \psi) = \mathbf{1}\{W_l^\top h_l(x) + b_l \geq 0\}$ achieving small misclassification rates $\delta_u, \delta_b \in (0, 1)$ on unsafe and benign queries respectively, where $\Upsilon(\cdot; \psi)$ is the binary decision rule of the linear classifier $\hat{y}_l(x)$ in Eq. (1). The ablation coverage satisfies $\rho_l \geq \rho_0$ for some constant $\rho_0 \in (0, 1)$.*

Assumption 2 (Bounded attack capability)**.** *The attacker perturbs only neurons in S_l, with $\|\delta\|_2 \leq \varepsilon$. After ablation, the effective perturbation support is restricted to $S_l \setminus \mathcal{I}_{safety,l}$.*

Assumption 3 (Local Lipschitz continuity of the output head)**.** *The subnetwork from layer l to the output, $z_\theta(\cdot)$, satisfies $\|z_\theta(h_l) - z_\theta(h_l')\|_2 \leq C_L \|h_l - h_l'\|_2$ in a neighborhood of $h_l(x)$, and all admissible perturbations remain within this neighborhood, where $C_L > 0$ is a constant.*

Assumption 4 (Bounded KL divergence between ablated and standard outputs)**.** *At convergence, $\mathbb{E}_{x \sim \mathcal{D}_{unsafe}}[D_{\text{KL}}(p_{\theta_{abl}}(y \mid x) \| p_\theta(y \mid x))] \leq \eta$ for a finite constant $\eta > 0$.*

Assumption 5 (Independent per-neuron exploitation)**.** *Each non-ablated safety neuron is independently exploitable with probability $\alpha_l \in [0, 1]$.*

Extended discussion and verifiability remarks are provided in Appendix A. We then state the main result showing that ablating safety neurons provably reduces the ASR upper bound relative to an undefended model, with the proof relying on three auxiliary lemmas in Appendix B.1: Lemma 1 shows ablation shrinks the adversarial subspace; Lemma 2 provides Lipschitz control of output perturbations; and Lemma 3 translates bounded KL divergence into bounded ℓ_1 distance via Pinsker's inequality.

Theorem 1 (ASR upper-bound reduction)**.** *Suppose Assumptions 1–5 hold. Let $s_l = |S_l|$ denote the number of safety neurons at layer l, $\rho_l \in (0, 1]$ the ablation coverage ratio, and $\alpha_l \in (0, 1)$ the per-neuron attack success probability. Define the neuron-level exploitation probabilities as $P_{\text{nl}} = 1-(1-\alpha_l)^{s_l}$ and the exploitation probability under our defense as $P_{\text{ours}} = 1-(1-\alpha_l)^{(1-\rho_l)s_l}$. Let $\mathcal{B}(\text{ASR})$ denote the theoretical upper bound of the attack success rate. If $\eta \leq \left((1-\alpha_l)^{(1-\rho_l)s_l} - (1-\alpha_l)^{s_l}\right)^2/2$, then $\mathcal{B}(\text{ASR}_{\text{nl}}) \geq \mathcal{B}(\text{ASR}_{\text{ours}})$, i.e., `NeuronGuard` strictly reduces the ASR upper bound relative to the undefended model.*

The sufficient condition on η formalizes an intuitive trade-off: ablation reduces the attacker's exploitable subspace, but introduces a distributional shift between the ablated and standard models. When KL regularization is tight enough, the defense's ASR upper bound is strictly lower than that of the undefended model (proof in Appendix B.2).

6 Experiments

6.1 Experimental setup

Models, datasets, and evaluation metrics: We conduct experiments on three pre-trained LLMs: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct (Team, 2024a), and Falcon3-7B-Instruct (Team, 2024b), and fine-tune LLMs on four datasets: SST2 (Socher et al., 2013), AG-News (Zhang et al., 2015), CoLA (Warstadt et al., 2019), and GSM8K (Cobbe et al., 2021). We assess each defense using two complementary metrics. **Utility accuracy (ACC)** measures task performance after fine-tuning, where a higher ACC indicates better preservation of model utility. **Attack success rate (ASR)** measures the fraction of unsafe responses produced under harmful queries, where a lower ASR indicates more effective safety preservation. Following prior work (Inan et al., 2023; Wu et al., 2026), we employ Llama-Guard-3-8B (Grattafiori et al., 2024) as the safety judge. We report the license of all models and datasets we used in this paper in Appendix C.

Baselines and attacks: We compare our method against six defense baselines. Three are prompt-level defenses: Perplexity Filter (Alon and Kamfonas, 2023), SmoothLLM (Robey et al., 2023), and GradSafe (Xie et al., 2024). Three are model-level defenses: CAT (Xhonneux et al., 2024),

Model	Task	No defense	Perplexity	SmoothLLM	GradSafe	CAT	LED	SafeNeuron	NeuronGuard
Llama	SST2	0.93	0.93	0.83	0.91	0.86	0.88	0.87	0.92
	AGNews	0.91	0.91	0.82	0.88	0.84	0.87	0.82	0.90
	CoLA	0.82	0.82	0.74	0.80	0.76	0.75	0.76	0.79
	GSM8K	0.65	0.65	0.55	0.63	0.58	0.61	0.60	0.62
Qwen	SST2	0.92	0.92	0.85	0.88	0.85	0.86	0.86	0.89
	AGNews	0.90	0.90	0.84	0.87	0.83	0.85	0.81	0.88
	CoLA	0.82	0.82	0.74	0.80	0.76	0.73	0.76	0.78
	GSM8K	0.65	0.65	0.46	0.62	0.58	0.55	0.60	0.58
Falcon	SST2	0.94	0.94	0.87	0.92	0.87	0.88	0.88	0.93
	AGNews	0.91	0.91	0.84	0.90	0.84	0.86	0.82	0.88
	CoLA	0.83	0.83	0.76	0.77	0.77	0.75	0.77	0.78
	GSM8K	0.68	0.68	0.57	0.61	0.61	0.62	0.63	0.66

Table 1: ACC↑ of different methods across three models and four fine-tuning tasks.

Model	Attack	No defense	Perplexity	SmoothLLM	GradSafe	CAT	LED	SafeNeuron	NeuronGuard
Llama	PAIR	0.17	0.14	0.15	0.09	0.10	0.07	0.04	0.01
	TAP	0.15	0.13	0.13	0.08	0.12	0.09	0.05	0.00
	Puzzler	0.29	0.25	0.23	0.12	0.17	0.13	0.07	0.01
	GCG	0.35	0.16	0.27	0.15	0.09	0.05	0.03	0.01
	AD	0.31	0.28	0.24	0.11	0.13	0.09	0.05	0.01
	NS	0.89	0.64	0.83	0.72	0.58	0.49	0.22	0.04
Qwen	PAIR	0.23	0.18	0.19	0.14	0.18	0.14	0.06	0.00
	TAP	0.25	0.21	0.10	0.16	0.17	0.13	0.07	0.01
	Puzzler	0.41	0.37	0.39	0.25	0.26	0.20	0.10	0.01
	GCG	0.40	0.18	0.33	0.17	0.13	0.08	0.04	0.01
	AD	0.36	0.36	0.23	0.18	0.20	0.15	0.07	0.01
	NS	0.83	0.57	0.68	0.68	0.53	0.42	0.21	0.02
Falcon	PAIR	0.15	0.12	0.12	0.07	0.05	0.02	0.03	0.01
	TAP	0.16	0.15	0.15	0.08	0.07	0.04	0.02	0.00
	Puzzler	0.30	0.25	0.27	0.09	0.09	0.05	0.03	0.01
	GCG	0.28	0.11	0.24	0.13	0.16	0.12	0.06	0.01
	AD	0.22	0.20	0.16	0.11	0.13	0.09	0.05	0.01
	NS	0.81	0.71	0.74	0.59	0.66	0.57	0.28	0.02

Table 2: ASR↓ of different methods under six attack types across three models on the SST2 task.

LED (Zhao et al., 2024), and SafeNeuron (Wang et al., 2026). Detailed descriptions are provided in Appendix D. We evaluate all defenses against six representative attacks spanning both attack families discussed in Section 3. PAIR (Chao et al., 2025), TAP (Mehrotra et al., 2024), and Puzzler (Chang et al., 2024) are generation-based jailbreak attacks that synthesize adversarial prompts to bypass safety mechanisms. GCG (Zou et al., 2023) and AutoDAN (Liu et al., 2024) are optimization-based jailbreak attacks that use gradient-based methods to craft token sequences that maximize the likelihood of unsafe responses. NeuroStrike (Wu et al., 2026) is a neuron-level attack that selectively prunes safety-critical neurons to disable safety mechanisms.

Parameter settings: We adopt LoRA (Hu et al., 2022) for fine-tuning, with rank $r = 32$ and scaling factor $\alpha = 64$, fine-tuning for 10 epochs on four utility tasks each containing 2,000 samples. The learning rate γ across all experiments is set to 10^{-5}. For all baselines and attacks, we use the default hyperparameter settings reported in the original works. For our method, the fraction of masked neurons ρ is set to 0.05, and the refresh interval N for the linear classifier is set to 200 steps. The safety probe set consists of 750 harmful queries from D_{unsafe} and 750 benign queries from D_{safe}, both sampled from BeaverTails (Ji et al., 2023). We use StrongREJECT (Souly et al., 2024) as the test dataset to compute ASR. All experimental results are reported as averages over 10 different random seeds, using two NVIDIA H100 GPUs, each with 94GB of memory.

6.2 Experimental results

NeuronGuard outperforms baselines: Table 1 reports the ACC of fine-tuned models on four tasks under various defenses. Our proposed NeuronGuard preserves high ACC and closely matches the "No defense" setting, indicating strong utility without performance degradation. For in-

stance, on Falcon (SST2), NeuronGuard achieves 0.93 ACC versus 0.94 under No defense, and on Llama (AGNews), it attains 0.90 versus 0.91. Table 2 reports the ASR under six attacks on SST2 (AutoDAN and NeuroStrike are abbreviated as AD and NS); Table 3 to Table 5 (Appendix) report the ASR for AGNews, CoLA, and GSM8K respectively. Existing baselines fail to provide effective defense under adversarial conditions. In contrast, NeuronGuard reduces ASR to nearly zero across all attacks, achieving 0.00 under PAIR on Qwen and 0.04 under NeuroStrike on Llama. Overall, NeuronGuard effectively defends against diverse attacks while maintaining high ACC on benign tasks.

Impact of ρ: We investigate how the selection ratio ρ in the top-ρ criterion, which controls the fraction of ablated safety-critical neurons, affects NeuronGuard. Table 6 (Appendix) reports ACC on SST2 and ASR under different attacks for different ρ. A larger ρ zeros out more safety neurons, compelling the remaining neurons to learn safety-relevant representations more intensively. Overall, NeuronGuard exhibits stable and robust performance even at low ρ, demonstrating the effectiveness of our approach in broadcasting safety signals.

Impact of N: We investigate the effect of the refresh interval N, which controls how frequently safety-critical neurons are re-identified. Table 7 (Appendix) reports ACC on SST2, ASR, and fine-tuning time for varying N. A smaller N keeps neuron estimates accurate but incurs higher overhead, while a larger N improves efficiency but may rely on outdated estimates. Overall, NeuronGuard maintains robust performance across a wide range of N, demonstrating that periodic re-identification effectively balances efficiency and accuracy.

Impact of the size of D_{safe} and D_{unsafe}: We investigate the impact of the sizes of D_{safe} and D_{unsafe}, used to identify safety-critical neurons and train refusal responses. Table 8 (Appendix) reports ASR and ACC for dataset sizes ranging from 500 to 2,500 samples (set equally). Defense performance marginally improves as size increases, eventually reducing ASR to near zero. Notably, NeuronGuard achieves highly effective defense with as few as 500 samples, demonstrating that our method is data-efficient and requires only limited safety data to robustly broadcast safety signals.

Computation cost of NeuronGuard: Figure 3 (Appendix) reports the running time of each method on SST2 under NeuroStrike, while test-time methods (e.g., Perplexity) report only inference time. The overhead of NeuronGuard remains on the same order of magnitude as "No defense". Since NeuronGuard operates during fine-tuning, it incurs no additional cost at inference time, unlike test-time defenses that require extra operations per input, making NeuronGuard particularly well suited to efficiency-critical deployment scenarios.

7 Discussion

Different variants of NeuronGuard: To evaluate each component's contribution, we compare NeuronGuard against four variants: **Variant I** uses only refusal training L_{ref} without neuron ablation; **Variant II** includes neuron ablation but omits the consistency regularizer L_{cons}; **Variant III** minimizes $L_{\text{user}} + L_{\text{safe}}$ without gradient projection; and **Variant IV** projects two gradients simultaneously in each step rather than using randomized projection. Table 9 (Appendix) reports ASR and ACC on SST2 across different attacks. Removing any component leads to either higher ASR or lower ACC, while the full NeuronGuard achieves the lowest ASR with high task accuracy, demonstrating that all components are indispensable.

Adaptive attacks: To evaluate worst-case resilience, we construct two adaptive attacks where the adversary has full knowledge of NeuronGuard. The first, iterative pruning (IP), repeatedly identifies and removes safety-related neurons on the progressively pruned model to eliminate newly emerging safety mechanisms. The second, nonlinear evasion (NE), leverages nonlinear models to uncover and remove safety-relevant neurons beyond those detectable by linear methods. More details are provided in Appendix E. Table 10 (Appendix) reports ASR under these attacks on SST2 (ACC remains identical to Table 1 on Llama). While existing defenses are highly vulnerable, NeuronGuard remains robust. Further analysis of NeuronGuard's robustness under adaptive attacks is presented in Appendix F.

NeuronGuard is effective for the multimodal scenario: Safety is also critical for vision language models (VLMs). To assess NeuronGuard in multimodal settings, we evaluate ACC and ASR under image inputs on Qwen2.5-VL-7B-Instruct (Team, 2025). For the fine-tuning task, we use SST2 in text-to-image (T2I) format, with both D_{safe} and D_{unsafe} constructed in T2I format from Beaver-

Tails (Ji et al., 2023). We consider two attack settings: (i) harmful queries from StrongREJECT (Souly et al., 2024) converted to T2I format, and (ii) NSFW images from the NSFW Detection dataset (deepghs, 2023), all resized to 224×224. We focus on NeuroStrike, as it naturally extends to VLMs and is the strongest attack considered. Table 11 (Appendix) reports ACC on SST2 and ASR under both settings (SR denotes StrongREJECT). NeuronGuard matches no-defense ACC and outperforms all baselines in ASR, demonstrating strong robustness in multimodal scenarios.

Effectiveness of dynamic neuron identification in NeuronGuard: A key design choice of NeuronGuard is to refresh the safety-critical neuron set periodically rather than fix it once in advance. To isolate its contribution, we replace NeuronGuard's dynamic identification with the static identification of SafeNeuron (Wang et al., 2026), keeping all other components unchanged. Table 12 (Appendix) reports ACC and ASR on SST2 with Llama. The ASR of static variant under NeuroStrike attack rises from 0.04 to 0.29, suggesting that a one-shot estimate becomes stale as safety signals redistribute during fine-tuning.

Sensitivity of NeuronGuard to the loss weighting: In Eq. (7), L_{user} and L_{safe} both carry a coefficient of 1. To check that this choice is not fragile, we introduce a weight λ_{safe} on L_{safe} purely for this analysis and vary it to 0.5 and 2. Table 13 (Appendix) reports ASR and ACC on SST2 with Llama. Results are stable across the range, and NeuronGuard keeps ASR at most 0.08 under all six attacks regardless of the weighting. A larger λ_{safe} trades accuracy for marginal ASR gains, so the default $\lambda_{\text{safe}} = 1$ already sits at the balance point.

8 Conclusion

We presented NeuronGuard, a fine-tuning-stage defense that redistributes safety signals across a broader set of neurons to address the root cause of alignment fragility. Through ablation-robust optimization, KL-divergence regularization, and randomized gradient projection, NeuronGuard hardens LLMs against both jailbreak and neuron-level attacks while preserving task utility, achieving near-zero ASR across diverse attack settings, multimodal scenarios, and white-box adaptive adversaries. These findings highlight the importance of improving safety robustness at the neuron level. We hope NeuronGuard can inspire future defenses that make safety alignment more resilient.

9 Limitations

NeuronGuard operates during the fine-tuning stage and therefore assumes that the defender has full control over the training pipeline, including the training data, model weights, and optimization objectives. This assumption may not hold in deployment scenarios where the model is released without any further fine-tuning, where the defender lacks the computational resources required to perform parameter updates, or where only black-box access to the model is available through an API. In such settings, complementary inference-time defenses would be needed to fill the gap.

10 Ethical considerations

The sole purpose of this paper is to advance the safety and robustness of LLMs against post-alignment attacks, with the goal of promoting the responsible deployment of AI systems in real-world applications. We do not encourage, facilitate, or endorse any malicious use of the attack methods discussed in this work. All jailbreak and neuron-level attacks examined in our experiments are drawn from prior publicly available research and are evaluated solely in controlled settings to benchmark the effectiveness of our defense framework, NeuronGuard. In our evaluation pipeline, we employ an LLM to assess attack success rates by determining whether a given model response violates policy. Although our work focuses on defensive alignment, analyzing safety-critical neurons and adaptive attack strategies could potentially inform stronger future attacks. We therefore encourage responsible disclosure and careful deployment of neuron-level safety analyses. The jailbreak prompts and evaluation data used in this work are obtained from publicly available benchmarks and prior research artifacts. We do not collect or use personally identifying information, and all experiments are conducted in controlled research settings involving potentially unsafe or offensive content solely for safety evaluation purposes.

Acknowledgments

We thank the reviewers for their constructive comments. This work was supported by the National Artificial Intelligence Research Resource (NAIRR) Pilot under Award Nos. 250513 and 260142.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

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

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, and 1 others. 2025. Nemotron 3 nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2512.20848*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Zhiyuan Chang, Mingyang Li, Yi Liu, Junjie Wang, Qing Wang, and Yang Liu. 2024. Play guessing game with llm: Indirect jailbreak attack with implicit clues. In *ACL*.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2025. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 23–42. IEEE.

Zirui Cheng, Jikai Sun, Anjun Gao, Yueyang Quan, Zhuqing Liu, Xiaohua Hu, and Minghong Fang. 2025. Secure retrieval-augmented generation against poisoning attacks. In *IEEE International Conference on Big Data*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

deepghs. 2023. Nsfw detection dataset. https://huggingface.co/datasets/deepghs/nsfw_detect. Multi-class image classification dataset for NSFW content detection with five categories: drawing, hentai, neutral, porn, and sexy.

Anjun Gao, Yueyang Quan, Zhuqing Liu, and Minghong Fang. 2026a. Beware what you autocomplete: Forensic attribution of backdoored code completions. In *Conference on Language Modeling (COLM)*.

Anjun Gao, Yueyang Quan, Yufei Xia, Zhuqing Liu, and Minghong Fang. 2026b. Patcher: Post-hoc patching of backdoored large language models. In *USENIX Security Symposium*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. In *ICLR*.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and 1 others. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36:24678–24704.

Yuqi Jia, Minghong Fang, Hongbin Liu, Jinghuai Zhang, and Neil Zhenqiang Gong. 2025. Tracing back the malicious clients in poisoning attacks to federated learning. In *NeurIPS*.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *ICLR*.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2024. Tree of attacks: Jailbreaking black-box llms automatically. *Advances in Neural Information Processing Systems*, 37:61065–61105.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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*, 36:53728–53741.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and 1 others. 2024. A strongreject for empty jailbreaks. *Advances in Neural Information Processing Systems*, 37:125416–125440.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Qwen Team. 2024a. Qwen2.5: A party of foundation models.

Qwen Team. 2025. Qwen2.5-vl.

Qwen Team. 2026. Qwen3.5: Accelerating productivity with native multimodal agents.

TII Team. 2024b. The falcon 3 family of open models.

Rachel SY Teo, Laziz U Abdullaev, and Tan M Nguyen. 2025. The blessing and curse of dimensionality in safety alignment. *arXiv preprint arXiv:2507.20333*.

Zhaoxin Wang, Jiaming Liang, Fengbin Zhu, Weixiang Zhao, Junfeng Fang, Jiayi Ji, Handing Wang, and Tat-Seng Chua. 2026. Safeneuron: Neuron-level safety alignment for large language models. *arXiv preprint arXiv:2602.12158*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Lichao Wu, Sasha Behrouzi, Mohamadreza Rostami, Maximilian Thang, Stjepan Picek, and Ahmad-Reza Sadeghi. 2026. Neurostrike: Neuron-level attacks on aligned llms. In *NDSS*.

Sophie Xhonneux, Alessandro Sordoni, Stephan Günnemann, Gauthier Gidel, and Leo Schwinn. 2024. Efficient adversarial training in llms with continuous attacks. *Advances in Neural Information Processing Systems*, 37:1502–1530.

Yueqi Xie, Minghong Fang, Renjie Pi, and Neil Gong. 2024. Gradsafe: Detecting jailbreak prompts for llms via safety-critical gradient analysis. In *ACL*.

Baolei Zhang, Haoran Xin, Yuxi Chen, Zhuqing Liu, Biao Yi, Tong Li, Lihai Nie, Zheli Liu, and Minghong Fang. 2026. Who taught the lie? responsibility attribution for poisoned knowledge in retrieval-augmented generation. In *IEEE Symposium on Security and Privacy*.

Baolei Zhang, Haoran Xin, Minghong Fang, Zhuqing Liu, Biao Yi, Tong Li, and Zheli Liu. 2025. Traceback of poisoning attacks to retrieval-augmented generation. In *The Web Conference*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Wei Zhao, Zhe Li, Yige Li, Ye Zhang, and Jun Sun. 2024. Defending large language models against jailbreak attacks via layer-specific editing. *arXiv preprint arXiv:2405.18166*.

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

Algorithm 1 Our NeuronGuard

Require: User fine-tuning dataset D_{user}, benign query dataset D_{safe}, harmful query dataset D_{unsafe}, pre-trained model parameters θ_0, classifier refresh interval N, learning rate γ

Ensure: Hardened model $\widehat{\theta}$ with robust safety

Initialize $\theta \leftarrow \theta_0$
for each fine-tuning step $t = 1, 2, \ldots$ **do**
  // Stage 1: Identifying safety-critical neurons
  if $t \bmod N = 1$ **then**
    For each layer l, train classifier $\hat{y}_l(x)$ from $D_{\text{safe}} \cup D_{\text{unsafe}}$ based on Eq. (1)
    Identify $\mathcal{I}_{\text{safety},l}$ based on Eq. (2)
  end if
  // Stage 2: Ablation-robust safety optimization
  Compute L_{ref} on harmful batch based on Eq. (3)
  Compute $L_{\text{ref}}^{\text{abl}}$ under ablated forward pass based on Eq. (4)
  Compute L_{cons} between ablated and standard passes based on Eq. (5)
  Set $L_{\text{safe}} = L_{\text{ref}} + L_{\text{ref}}^{\text{abl}} + L_{\text{cons}}$ based on Eq. (6)
  // Stage 3: Resolving gradient conflicts via randomized projection
  Compute $g_{\text{user}} = \nabla_\theta L_{\text{user}}$, $g_{\text{safe}} = \nabla_\theta L_{\text{safe}}$
  if $\langle g_{\text{user}}, g_{\text{safe}} \rangle < 0$ **then**
    Sample ordering (g_1, g_2) of $(g_{\text{user}}, g_{\text{safe}})$ and derive $\hat{g}_1$, $\hat{g}_2$ based on Eq. (8)
  else
    Set $\hat{g}_1 = g_{\text{user}}$, $\hat{g}_2 = g_{\text{safe}}$
  end if
  Compute $g_{\text{final}} = \hat{g}_1 + \hat{g}_2$ based on Eq. (9)
  Update $\theta \leftarrow \theta - \gamma \cdot g_{\text{final}}$
end for
$\widehat{\theta} \leftarrow \theta$
return $\widehat{\theta}$

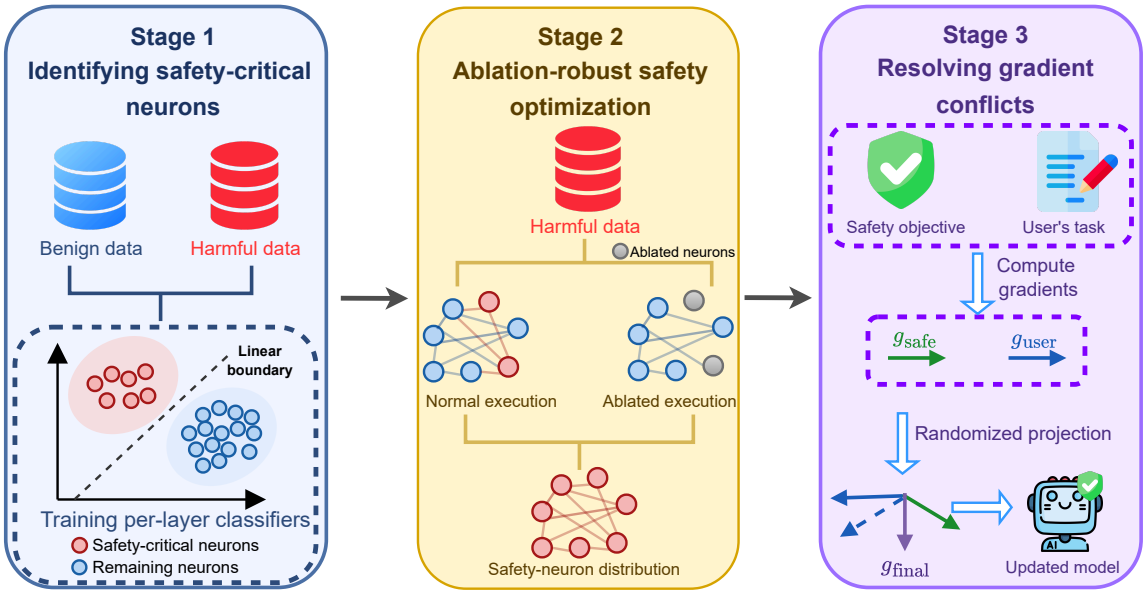

Figure 2: Overview of NeuronGuard, which hardens safety alignment against post-alignment attacks via three coupled stages. **Stage 1** trains per-layer linear classifiers on benign and harmful activations to identify the sparse safety-critical neuron subset. **Stage 2** deliberately ablates these neurons during training, forcing safety representations to redistribute across a broader neuron population via normal and ablated forward passes with distributional consistency regularization. **Stage 3** resolves conflicts between the safety gradient g_{safe} and the user task gradient g_{user} via randomized projection, producing a final update g_{final} that preserves both robustness and utility.

A Discussion of assumptions

We provide the extended discussion of the assumptions stated in Section 5.

On Assumption 1 (linear separability and ablation coverage): While the exact internal safety boundary and the population-level set S_l are not directly observable, prior work in mechanistic interpretability suggests that high-level behaviors in neural networks, including safety or refusal-related behaviors, often concentrate in a sparse subset of specialized neurons (Bau et al., 2017; Wu et al., 2026). Separately, empirical studies on activation-based analysis indicate that simple classifiers trained on intermediate representations can achieve non-trivial separation between benign and unsafe (or refusal-related) prompts, supporting approximate linear separability at certain layers (Teo et al., 2025). Assumption 1 is particularly suitable for safety-aligned models with stable refusal patterns or narrowly defined unsafe behaviors, where unsafe prompts induce consistent activation shifts at certain layers. In practice, the quantities δ_u, δ_b, and ρ_l can be estimated on held-out data, allowing the validity of the assumption to be partially verified empirically rather than taken as an oracle condition.

Note that ρ_l is a theoretical quantity distinct from the hyperparameter ρ: ρ controls the *size* of the selected set, while ρ_l measures the *overlap* with the true safety-neuron set S_l.

On Assumption 4 (bounded KL divergence): The direction $D_{\mathrm{KL}}(p_{\theta_{\text{abl}}} \| p_\theta)$ treats the standard distribution as the reference, so minimizing it enforces prediction consistency while maintaining consistent refusal behavior on unsafe prompts.

Notation details: The safety alignment objective is formally defined as

$$\max_{\theta} \mathbb{E}_{x\sim\mathcal{D}_{\text{unsafe}}} \mathbb{E}_{y\sim p_\theta(\cdot|x)} \big[R_{\text{refuse}}(x,y)\big], \quad (10)$$

where R_{refuse} is a human-defined reward that assigns higher values to refusal or other policy-compliant responses and penalizes harmful generations. The population distributions $\mathcal{D}_{\text{unsafe}}$ and $\mathcal{D}_{\text{safe}}$ are the theoretical counterparts of the empirical datasets D_{unsafe} and D_{safe} used in Section 4.

Following (Wu et al., 2026), we define an internal decision boundary

$$\Upsilon(h_l(x);\phi) = \begin{cases} 1, & x \in \mathcal{D}_{\text{unsafe}}, \\ 0, & x \in \mathcal{D}_{\text{safe}}, \end{cases} \quad (11)$$

where $\phi \subseteq \theta$ denotes the subset of model parameters responsible for discriminating between unsafe and benign activations within the layer. The activation vector at layer l is $h_l(x) = [h_{1,l}(x), h_{2,l}(x), \ldots, h_{d_l,l}(x)]^\top$ with $h_{i,l}(x)$ denoting the activation of neuron i at layer l (following NeuroStrike, we omit the token index for notational simplicity). S_l is the population-level safety-neuron set, whereas $\mathcal{I}_{\text{safety},l}$ (defined in Section 4.1) is its empirical approximation obtained via the top-ρ linear-classifier selection rule.

Given the identified safety-critical neuron set, the ablated forward pass zeros out activations:

$$\tilde{h}_{i,l}(x) = \begin{cases} 0, & \text{if } i \in \mathcal{I}_{\text{safety},l}, \\ h_{i,l}(x), & \text{otherwise.} \end{cases} \quad (12)$$

The resulting ablated activation vector $\tilde{h}_l(x)$ is then propagated through the remaining subnetwork from layer $l+1$ to the output layer.

B Proof of Theorem 1

In this appendix, we provide detailed proofs for the auxiliary lemmas and for the main theorem. All results are derived under the notation and assumptions in the main text, in particular: (i) the per-layer safety-neuron set S_l defined in Section 5, where $h_{i,l}(x)$ denotes the activation of neuron i at layer l; (ii) the probe-selected safety-critical neuron index set $\mathcal{I}_{\text{safety},l} \subseteq [d_l]$ (identified via the top-ρ selection on classifier weights W_l from Section 4.1) with ablation coverage ratio $\rho_l := |S_l \cap \mathcal{I}_{\text{safety},l}| \,/\, |S_l|$; (iii) the attacker's constrained perturbation set at layer l given in Assumption 2; and (iv) all probabilities and expectations are taken over the unsafe input distribution $\mathcal{D}_{\text{unsafe}}$ unless otherwise stated.

B.1 Proof of auxiliary lemmas

Lemma 1 (Ablation shrinks the adversarial subspace)**.** *Let $S_l \subseteq \{1,\ldots,d_l\}$ be the true safety-neuron index set at layer l, and let $\mathcal{I}_{\text{safety},l} \subseteq [d_l]$ be the set of safety-critical neuron indices identified by the probe, where $i \in \mathcal{I}_{\text{safety},l}$ means neuron i is ablated (zeroed out in the forward pass). Define the ablation coverage ratio as*

$$\rho_l := \frac{|S_l \cap \mathcal{I}_{\text{safety},l}|}{|S_l|} \in [0,1]. \quad (13)$$

Under Assumption 2, after ablation, the attacker's effective perturbation set is reduced from S_l to $S_l^{\text{eff}} := S_l \setminus \mathcal{I}_{\text{safety},l}$, and therefore

$$|S_l^{\text{eff}}| = |S_l|(1-\rho_l). \quad (14)$$

Proof. For a vector $\delta \in \mathbb{R}^{d_l}$, let $\mathrm{supp}(\delta) := \{\, i \mid \delta_i \neq 0 \,\}$ denote its support. We also denote by $\Delta_l(x) \in \mathbb{R}^{d_l}$ the adversarial perturbation injected at layer l for input x, and write $\Delta_{l,i}(x)$ for its i-th coordinate. By Assumption 2, prior to ablation, the attacker is allowed to perturb only the coordinates in S_l. Then

$$\Delta_l(x) \in \mathcal{U}_l := \{\, \delta \in \mathbb{R}^{d_l} \mid \mathrm{supp}(\delta) \subseteq S_l, \; \|\delta\|_2 \leq \varepsilon \,\}. \quad (15)$$

Therefore, if $i \notin S_l$, then necessarily $\Delta_{l,i}(x) = 0$.

Our defense produces the ablated activation at each coordinate:

$$\tilde{h}_{i,l}(x) = \begin{cases} 0, & \text{if } i \in \mathcal{I}_{\text{safety},l}, \\ h_{i,l}(x), & \text{otherwise}, \end{cases} \quad (16)$$

where $h_{i,l}(x)$ is the original activation of neuron i at layer l.

For any index with $i \in \mathcal{I}_{\text{safety},l}$, the forward pass always uses 0 on that coordinate, so any attacker perturbation on this coordinate is nullified. Hence, for a perturbation component to be effective, the index must (i) belong to the original attackable set S_l, and (ii) not be ablated, i.e. $i \notin \mathcal{I}_{\text{safety},l}$. The set of such coordinates is exactly $S_l^{\text{eff}} = S_l \setminus \mathcal{I}_{\text{safety},l}$. By definition of ρ_l:

$$\begin{aligned} &|S_l \cap \mathcal{I}_{\text{safety},l}| = \rho_l |S_l| \\ &\implies |S_l^{\text{eff}}| = |S_l| - \rho_l |S_l| = |S_l|(1-\rho_l). \end{aligned} \quad (17)$$

Thus, ablation shrinks the adversarial subspace by a factor of $(1-\rho_l)$.

We now illustrate the probability of successful neuron-level exploitation of our defense compared to NeuroStrike.

Recall that $\alpha_l \in [0,1]$ denotes the per-neuron attack success probability (Assumption 5), i.e., the probability that an adaptive attacker can successfully exploit a single available safety neuron in S_l. In the NeuroStrike baseline, no ablation defense is applied, and thus all neurons in S_l remain available for manipulation. Let $|S_l| = s_l$. Under the assumption of independent per-neuron attack attempts, the probability that at least one neuron in S_l is successfully exploited is:

$$P_{nl} = 1 - (1-\alpha_l)^{s_l}. \quad (18)$$

In our ablation defense, only a fraction $(1-\rho_l)$ of neurons remain non-ablated and therefore available for attack. Let the number of such non-ablated neurons be $s_l^{\text{rem}} = (1-\rho_l)\, s_l$. The corresponding probability that at least one non-ablated neuron is successfully exploited is:

$$\begin{aligned} P_{\text{ours}} &= 1 - (1-\alpha_l)^{s_l^{\text{rem}}} \\ &= 1 - (1-\alpha_l)^{(1-\rho_l)s_l}. \end{aligned} \quad (19)$$

Since $(1-\rho_l)s_l < s_l$ and $\alpha_l \in (0,1)$, we have

$$P_{\text{ours}} < P_{nl}. \quad (20)$$

This inequality rigorously shows that, for the same per-neuron attackability α_l, our ablation defense strictly reduces the probability of successful neuron-level exploitation compared to NeuroStrike. □

Lemma 2 (Lipschitz control of output change after ablation/attack)**.** *Let $h_l(x)$ be the original activation vector at layer l, with components $h_{i,l}(x)$, and let $\tilde{h}_l(x)$ be the ablated activation vector with components $\tilde{h}_{i,l}(x) = 0$ if $i \in \mathcal{I}_{safety,l}$, and $\tilde{h}_{i,l}(x) = h_{i,l}(x)$ otherwise. Under Assumption 3, for any input x and any admissible attack δ supported on S_l^{eff} with $\|\delta\|_2 \leq \varepsilon$, we have*

$$\begin{aligned} \big\| z_\theta(\tilde{h}_l(x) + \delta) - z_\theta(\tilde{h}_l(x)) \big\|_2 &\leq C_L\, \varepsilon, \\ \big\| z_\theta(h_l(x) + \delta) - z_\theta(h_l(x)) \big\|_2 &\leq C_L\, \varepsilon. \end{aligned} \quad (21)$$

Proof. Assumption 3 states that the subnetwork from layer l to the output is C_L-Lipschitz in a neighborhood of the ablated activation:

$$\|z_\theta(u) - z_\theta(v)\|_2 \leq C_L \|u - v\|_2 \quad (22)$$

for all u, v in that neighborhood.

Take $u = \tilde{h}_l(x) + \delta$ and $v = \tilde{h}_l(x)$. Then

$$\begin{aligned} \|z_\theta(u) - z_\theta(v)\|_2 &\leq C_L \|u - v\|_2 \\ &= C_L \|\delta\|_2 \leq C_L \varepsilon, \end{aligned} \quad (23)$$

since $\|\delta\|_2 \leq \varepsilon$ by the attack constraint. Similarly, we obtain the second result by taking $u = h_l(x) + \delta$ and $v = h_l(x)$. □

Remark. *Lemma 2 isolates only the effect of the attack on top of the ablated activation $\tilde{h}_l(x)$; the ablation operation itself is absorbed into $\tilde{h}_l(x)$.*

Lemma 3 (Small expected KL implies small expected ℓ_1 difference)**.** *Under Assumption 4, we have*

$$\begin{aligned} &\mathbb{E}_{x \sim \mathcal{D}_{unsafe}} \big[\|p_{\theta_{abl}}(\cdot \mid x) - p_\theta(\cdot \mid x)\|_1 \big] \\ &\leq \sqrt{2\eta}. \end{aligned} \quad (24)$$

Proof. Let

$$Z(x) := D_{\mathrm{KL}}\big(p_{\theta_{\text{abl}}}(y \mid x) \,\|\, p_\theta(y \mid x)\big) \geq 0. \tag{25}$$

By Assumption 4 we know that $\mathbb{E}_{x\sim\mathcal{D}_{\text{unsafe}}}[Z(x)] \leq \eta$.

We now relate the KL divergence $Z(x)$ to the ℓ_1 (total variation) distance of the two output distributions. For any fixed x, since $p_\theta(\cdot \mid x)$ and $p_{\theta_{\text{abl}}}(\cdot \mid x)$ are probability distributions over the same measurable output space, Pinsker's inequality states that for two probability distributions P and Q,

$$\|P - Q\|_1 \leq \sqrt{2\, D_{\mathrm{KL}}(P\|Q)}. \tag{26}$$

Applying this to $P = p_{\theta_{\text{abl}}}(\cdot \mid x)$ and $Q = p_\theta(\cdot \mid x)$, we obtain for every x:

$$\big\|p_{\theta_{\text{abl}}}(\cdot \mid x) - p_\theta(\cdot \mid x)\big\|_1 \leq \sqrt{2\, Z(x)}. \tag{27}$$

Taking expectation over $x \sim \mathcal{D}_{\text{unsafe}}$ on both sides yields

$$\mathbb{E}_x\big[\|p_{\theta_{\text{abl}}}(\cdot \mid x) - p_\theta(\cdot \mid x)\|_1\big] \leq \mathbb{E}_x\big[\sqrt{2\, Z(x)}\big]. \tag{28}$$

We upper-bound the right-hand side using Jensen's inequality. Since $f(t) = \sqrt{t}$ is concave on $t \geq 0$, for any nonnegative random variable W we have $\mathbb{E}[\sqrt{W}] \leq \sqrt{\mathbb{E}[W]}$. Letting $W = 2Z(x)$:

$$\mathbb{E}_x\big[\sqrt{2\, Z(x)}\big] \leq \sqrt{\mathbb{E}_x\big[2\, Z(x)\big]} = \sqrt{2\, \mathbb{E}_x\big[Z(x)\big]}. \tag{29}$$

Combining and invoking Assumption 4 ($\mathbb{E}_x[Z(x)] \leq \eta$):

$$\mathbb{E}_x\big[\|p_{\theta_{\text{abl}}}(\cdot \mid x) - p_\theta(\cdot \mid x)\|_1\big] \leq \sqrt{2\,\eta}. \tag{30}$$

This proves the lemma. □

B.2 Proof for Theorem 1

Proof. We first recall the two ASR upper bounds derived under Assumptions 1–5.

ASR of the NeuroStrike baseline: By Lemma 1, before our ablation defense, the attacker can exploit any of the $s_l = |S_l|$ safety neurons, and under the per-neuron success model with probability α_l this happens with probability

$$P_{nl} = 1 - (1 - \alpha_l)^{s_l}. \tag{31}$$

By the Lipschitz control before ablation (Lemma 2 applied to the standard activation $h_l(x)$), an unsafe output can occur either due to successful neuron-level exploitation or due to the insufficient clean safety margin under a Lipschitz-bounded perturbation. By a union bound over these two failure modes, we obtain the upper bound:

$$\mathcal{B}(\mathrm{ASR}_{nl}) = P_{nl} + \Pr_{x\sim\mathcal{D}_{\text{unsafe}}}\big[m(x) \leq C_L\varepsilon\big]. \tag{32}$$

where $m(x)$ denotes the clean safety margin of the standard model on input x.

ASR of our defense: With our ablation, Lemma 1 shows the effective attackable neurons are reduced to $(1-\rho_l)s_l$, hence the corresponding neuron-level success probability is

$$P_{\text{ours}} = 1 - (1 - \alpha_l)^{(1-\rho_l)s_l}. \tag{33}$$

An attack can still lead to unsafe generations after ablation. Therefore, we upper bound the attack success rate by the probability of sampling an unsafe output under the ablated model. Taking expectation over $x \sim \mathcal{D}_{\text{unsafe}}$, and we denote by $\mathcal{Y}_{\text{unsafe}}$ the set of all model outputs deemed unsafe or policy-violating under the safety criterion. We obtain the following upper bound on the expected ASR:

$$\mathcal{B}(\mathrm{ASR}_{\text{ours}}) = P_{\text{ours}} + \mathbb{E}_x \Pr_{y\sim p_{\theta_{\text{abl}}}(\cdot|x)}\big[y \in \mathcal{Y}_{\text{unsafe}}\big]. \tag{34}$$

Moreover, for any unsafe prompt x, by the definition of total variation distance:

$$\begin{aligned}&\Pr_{y\sim p_{\theta_{\text{abl}}}(\cdot|x)}\big[y \in \mathcal{Y}_{\text{unsafe}}\big] \\ &\leq \Pr_{y\sim p_\theta(\cdot|x)}\big[y \in \mathcal{Y}_{\text{unsafe}}\big] \\ &\quad + \big\|p_{\theta_{\text{abl}}}(\cdot \mid x) - p_\theta(\cdot \mid x)\big\|_1.\end{aligned} \tag{35}$$

Taking expectation over $x \sim \mathcal{D}_{\text{unsafe}}$ and applying Lemma 3 yields:

$$\mathbb{E}_x \Pr_{y\sim p_{\theta_{\text{abl}}}(\cdot|x)}[y \in \mathcal{Y}_{\text{unsafe}}] \leq \Pr[m(x) \leq C_L\varepsilon] + \sqrt{2\eta}. \tag{36}$$

Taking the difference: Subtracting the two ASR upper bounds yields:

$$\begin{aligned}&\mathcal{B}(\mathrm{ASR}_{nl}) - \mathcal{B}(\mathrm{ASR}_{\text{ours}}) \\ &= \big(P_{nl} - P_{\text{ours}}\big) + \Pr[m(x) \leq C_L\varepsilon] \\ &\quad - \mathbb{E}_x \Pr_{y\sim p_{\theta_{\text{abl}}}(\cdot|x)}[y \in \mathcal{Y}_{\text{unsafe}}].\end{aligned} \tag{37}$$

Applying (36) to (37) yields:

$$\mathcal{B}(\mathrm{ASR}_{nl}) - \mathcal{B}(\mathrm{ASR}_{\mathrm{ours}}) \geq \big(P_{nl} - P_{\mathrm{ours}}\big) - \sqrt{2\eta}. \tag{38}$$

Recalling the neuron-level exploitation probabilities in Lemma 1, where $s_l = |S_l|$ and ρ_l is the ablation coverage ratio:

$$\begin{aligned} P_{nl} - P_{\mathrm{ours}} &= \Big(1 - (1-\alpha_l)^{s_l}\Big) \\ &\quad - \Big(1 - (1-\alpha_l)^{(1-\rho_l)s_l}\Big) \\ &= (1-\alpha_l)^{(1-\rho_l)s_l} - (1-\alpha_l)^{s_l}. \end{aligned} \tag{39}$$

Since $\rho_l \in (0,1)$ and $\alpha_l \in (0,1)$, we have $(1-\rho_l)s_l < s_l$ and the map $t \mapsto (1-\alpha_l)^t$ is strictly decreasing in t, which implies

$$(1-\alpha_l)^{(1-\rho_l)s_l} - (1-\alpha_l)^{s_l} > 0. \tag{40}$$

Combining this with the previous inequality yields:

$$\begin{aligned} &\mathcal{B}(\mathrm{ASR}_{nl}) - \mathcal{B}(\mathrm{ASR}_{\mathrm{ours}}) \\ &\geq \Big((1-\alpha_l)^{(1-\rho_l)s_l} - (1-\alpha_l)^{s_l}\Big) - \sqrt{2\eta}. \end{aligned} \tag{41}$$

Hence, under the sufficient condition

$$\sqrt{2\eta} \leq (1-\alpha_l)^{(1-\rho_l)s_l} - (1-\alpha_l)^{s_l}, \tag{42}$$

we obtain

$$\mathcal{B}(\mathrm{ASR}_{nl}) - \mathcal{B}(\mathrm{ASR}_{\mathrm{ours}}) \geq 0, \tag{43}$$

i.e., $\mathcal{B}(\mathrm{ASR}_{nl}) \geq \mathcal{B}(\mathrm{ASR}_{\mathrm{ours}})$. This completes the proof. □

C License of models and datasets

We summarize the licenses of all models and datasets used in this paper.

Models: Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Llama-Guard-3-8B (Grattafiori et al., 2024) are both released under the Llama 3.1 Community License, a custom commercial license provided by Meta Platforms, Inc. Qwen2.5-7B-Instruct (Team, 2024a) and Qwen2.5-VL-7B-Instruct (Team, 2025) are released under the Apache License 2.0. Falcon3-7B-Instruct (Team, 2024b) is released under the TII Falcon License 2.0.

Datasets: SST2 (Socher et al., 2013) is made available by Stanford for research purposes; no explicit license is stated on the official website. AG-News (Zhang et al., 2015) is provided by the academic community for research purposes, including data mining, information retrieval, and other non-commercial activities, with no explicit open-source license attached. CoLA (Warstadt et al., 2019) consists of sentences excerpted from published linguistics literature; the corpus and baseline code are made available under the MIT License. GSM8K (Cobbe et al., 2021) is released by OpenAI under the MIT License. StrongREJECT (Souly et al., 2024) releases its custom-generated data and code under the MIT License. The NSFW Detection dataset (deepghs, 2023) is released under the MIT License.

D Details of baselines

Perplexity (Alon and Kamfonas, 2023): Perplexity serves as the filter by computing the model's own perplexity on the prompt and flags inputs whose perplexity exceeds a threshold to determine whether the prompt is a jailbreak attempt.

SmoothLLM (Robey et al., 2023): SmoothLLM is a character-level defense that mitigates jailbreaking attacks by exploiting the fragility of adversarial prompts to random perturbations. It creates N perturbed copies of an input prompt and passes them through the target LLM, then aggregates the resulting responses through a majority vote to detect and filter out adversarial inputs without requiring model retraining.

GradSafe (Xie et al., 2024): GradSafe identifies jailbreak prompts by analyzing the gradients of an LLM's safety-critical parameters when a prompt is paired with a compliance response. The method utilizes the observation that adversarial prompts exhibit consistent gradient patterns across these parameters, which differ significantly from those of safe prompts.

CAT (Xhonneux et al., 2024): CAT utilizes adversarial training in the continuous embedding space and uses a lightweight surrogate model to select the most effective adversarial perturbations from K candidates before mapping them back to the discrete token space for fine-tuning, achieving robustness against various jailbreak attacks while maintaining the model's nominal performance.

LED (Zhao et al., 2024): LED is an editing-based defense that mitigates jailbreak attacks by identifying and editing specific "safety-critical" layers within an LLM. The method works by analyzing the differences in hidden state activations between safe and adversarial prompts to locate the layers most responsible for safety alignment.

SafeNeuron (Wang et al., 2026): SafeNeuron identifies safety-critical neurons by comparing activations under safe and unsafe inputs using Activation Effect Size and Safety Activation Shift metrics, then freezes these neurons during direct preference optimization to force the model to construct redundant safety pathways across the remaining parameters, improving robustness against neuron-level pruning attacks.

E Details of adaptive attacks

Iterative pruning (IP): This attack directly targets the identification step described in Section 4.1. Specifically, the attacker trains per-layer linear classifiers in the same manner as `NeuronGuard` to identify the safety-neuron index sets $\mathcal{I}_{\text{safety},l}$. The attack then proceeds in iterative rounds: in each round, the attacker re-trains the classifiers on the ablated model to identify a new set of neurons that have assumed safety-related responsibilities, and then permanently removes them by zeroing out the corresponding positions. We set the number of pruning rounds to 5.

Nonlinear evasion (NE): This attack exploits the limitation of our neuron identification, which relies on linear classifiers. Specifically, instead of training per-layer linear classifiers, the attacker trains a two-layer MLP for each layer to search for alternative safety-neuron sets that capture nonlinear neuron interactions. The attack then ablates the identified neurons at inference time, targeting safety-relevant neurons that lie outside the linearly identified set and may not be captured by `NeuronGuard`.

F Robustness under adaptive attacks

`NeuronGuard`'s resilience against adaptive adversaries is rooted in the fundamental asymmetry between what the attacker must accomplish and what the defense requires to succeed. Because safety signals are continuously redistributed across an ever-wider population of neurons throughout fine-tuning, an adversary with full knowledge of the defense architecture cannot simply target a fixed, locatable set of neurons and expect lasting success. By the time an attacker has identified and suppressed one generation of safety-bearing neurons, the training process has already propagated those representations into new neurons elsewhere in the network, effectively moving the goalposts with each successive step.

The iterative pruning attack illustrates this asymmetry clearly. This adaptive strategy mirrors `NeuronGuard`'s own identification logic, using linear classifiers to locate safety neurons and removing them in successive rounds on the progressively weakened model. Yet the attacker must achieve complete elimination of safety behavior across all redundant sites, while the defense only needs to preserve it in a sufficient fraction of them. The ablation-robust optimization that `NeuronGuard` undergoes during fine-tuning is structurally identical to the pressure this attack applies, meaning the model has been explicitly conditioned to withstand repeated targeted suppression of its safety neurons. The result is an ASR of just 0.09, compared to 0.83 without any defense.

The nonlinear evasion attack takes a different approach by training MLPs to uncover safety-relevant neurons that interact in ways beyond what linear classifiers can detect, attempting to exploit the boundary of `NeuronGuard`'s identification capability. Even so, the defense holds because its robustness does not hinge on neuron identification being exhaustive. Partial ablation coverage alone is sufficient to meaningfully shrink the set of neurons available for adversarial exploitation, and the KL-divergence regularization further constrains how far the ablated model's behavior can drift from the standard model. These two properties together provide a robustness floor that persists even when the attacker successfully probes beyond the linearly identifiable safety neurons, keeping ASR at just 0.05.

Model	Attack	No defense	Perplexity	SmoothLLM	GradSafe	CAT	LED	SafeNeuron	NeuronGuard
Llama	PAIR	0.15	0.14	0.15	0.09	0.10	0.07	0.04	0.01
	TAP	0.15	0.13	0.11	0.08	0.12	0.09	0.05	0.00
	Puzzler	0.28	0.25	0.23	0.12	0.17	0.13	0.07	0.00
	GCG	0.35	0.16	0.27	0.15	0.09	0.05	0.04	0.01
	AD	0.31	0.28	0.26	0.11	0.13	0.09	0.05	0.01
	NS	0.89	0.65	0.86	0.72	0.58	0.49	0.22	0.02
Qwen	PAIR	0.23	0.18	0.19	0.14	0.18	0.17	0.06	0.00
	TAP	0.25	0.21	0.10	0.16	0.17	0.13	0.07	0.03
	Puzzler	0.41	0.37	0.39	0.25	0.29	0.20	0.10	0.01
	GCG	0.40	0.17	0.33	0.17	0.13	0.08	0.04	0.01
	AD	0.36	0.35	0.23	0.18	0.20	0.15	0.09	0.01
	NS	0.83	0.57	0.70	0.68	0.53	0.42	0.21	0.00
Falcon	PAIR	0.15	0.12	0.12	0.08	0.05	0.02	0.03	0.01
	TAP	0.16	0.15	0.15	0.06	0.07	0.03	0.02	0.00
	Puzzler	0.30	0.25	0.28	0.09	0.09	0.05	0.04	0.01
	GCG	0.28	0.13	0.24	0.13	0.16	0.12	0.06	0.00
	AD	0.22	0.20	0.14	0.11	0.13	0.09	0.04	0.01
	NS	0.81	0.70	0.74	0.59	0.66	0.57	0.28	0.02

Table 3: ASR of different methods under six attack types across three models on the AGNews task.

Model	Attack	No defense	Perplexity	SmoothLLM	GradSafe	CAT	LED	SafeNeuron	NeuronGuard
Llama	PAIR	0.17	0.16	0.15	0.07	0.10	0.07	0.04	0.01
	TAP	0.15	0.13	0.13	0.08	0.15	0.09	0.05	0.00
	Puzzler	0.29	0.25	0.23	0.12	0.19	0.13	0.07	0.01
	GCG	0.35	0.16	0.27	0.15	0.06	0.05	0.03	0.01
	AD	0.31	0.28	0.24	0.11	0.10	0.09	0.05	0.01
	NS	0.89	0.64	0.83	0.72	0.58	0.46	0.22	0.04
Qwen	PAIR	0.26	0.18	0.19	0.14	0.18	0.14	0.06	0.01
	TAP	0.24	0.21	0.10	0.16	0.17	0.13	0.07	0.01
	Puzzler	0.41	0.37	0.39	0.25	0.26	0.20	0.10	0.00
	GCG	0.40	0.18	0.33	0.17	0.13	0.08	0.04	0.00
	AD	0.36	0.38	0.23	0.18	0.20	0.15	0.04	0.01
	NS	0.83	0.57	0.68	0.68	0.54	0.42	0.21	0.00
Falcon	PAIR	0.15	0.12	0.12	0.05	0.05	0.02	0.03	0.01
	TAP	0.15	0.15	0.15	0.08	0.07	0.04	0.02	0.00
	Puzzler	0.30	0.26	0.27	0.09	0.08	0.05	0.03	0.01
	GCG	0.28	0.11	0.24	0.15	0.16	0.12	0.05	0.01
	AD	0.22	0.20	0.16	0.09	0.13	0.07	0.05	0.01
	NS	0.81	0.71	0.74	0.59	0.66	0.57	0.26	0.02

Table 4: ASR of different methods under six attack types across three models on the CoLA task.

Model	Attack	No defense	Perplexity	SmoothLLM	GradSafe	CAT	LED	SafeNeuron	NeuronGuard
Llama	PAIR	0.17	0.14	0.15	0.10	0.10	0.07	0.04	0.01
	TAP	0.15	0.13	0.13	0.08	0.12	0.09	0.06	0.00
	Puzzler	0.29	0.23	0.23	0.12	0.17	0.13	0.07	0.01
	GCG	0.32	0.16	0.27	0.15	0.09	0.05	0.03	0.01
	AD	0.31	0.28	0.24	0.11	0.13	0.09	0.03	0.01
	NS	0.91	0.66	0.83	0.72	0.58	0.49	0.22	0.04
Qwen	PAIR	0.23	0.18	0.19	0.14	0.18	0.15	0.09	0.00
	TAP	0.23	0.21	0.10	0.16	0.17	0.13	0.07	0.01
	Puzzler	0.39	0.37	0.39	0.25	0.26	0.20	0.10	0.00
	GCG	0.40	0.16	0.33	0.17	0.13	0.08	0.04	0.01
	AD	0.36	0.35	0.23	0.18	0.22	0.15	0.07	0.01
	NS	0.83	0.58	0.68	0.68	0.55	0.42	0.21	0.02
Falcon	PAIR	0.15	0.12	0.12	0.07	0.03	0.02	0.03	0.01
	TAP	0.16	0.15	0.15	0.08	0.07	0.04	0.02	0.00
	Puzzler	0.30	0.25	0.27	0.07	0.09	0.05	0.03	0.01
	GCG	0.28	0.09	0.24	0.13	0.16	0.12	0.06	0.03
	AD	0.22	0.20	0.16	0.11	0.10	0.09	0.05	0.01
	NS	0.81	0.71	0.74	0.59	0.65	0.57	0.28	0.02

Table 5: ASR of different methods under six attack types across three models on the GSM8K task.

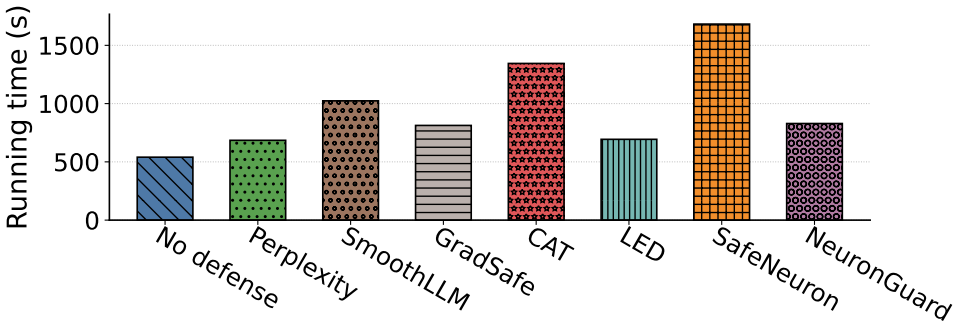

Figure 3: Overhead (seconds) for different methods.

ρ	PAIR	TAP	Puzzler	GCG	AutoDAN	NeuroStrike	SST2
	ASR	ASR	ASR	ASR	ASR	ASR	ACC
0.01	0.08	0.07	0.09	0.08	0.07	0.10	0.93
0.05	0.01	0.00	0.01	0.01	0.01	0.04	0.92
0.10	0.01	0.00	0.01	0.01	0.01	0.01	0.92
0.15	0.00	0.00	0.01	0.00	0.01	0.01	0.91
0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.88
0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.84

Table 6: Results for the fraction of masked neurons ρ under different attacks on the SST2 task with the Llama model.

N	PAIR	TAP	Puzzler	GCG	AutoDAN	NeuroStrike	SST2	Time
	ASR	ASR	ASR	ASR	ASR	ASR	ACC	seconds
50	0.01	0.00	0.01	0.01	0.01	0.01	0.88	2628.41
100	0.01	0.00	0.01	0.01	0.01	0.01	0.90	1426.58
200	0.01	0.00	0.01	0.01	0.01	0.04	0.93	829.18
400	0.04	0.03	0.05	0.04	0.04	0.10	0.93	673.45
800	0.14	0.12	0.17	0.15	0.13	0.26	0.93	619.49

Table 7: Results for the refresh interval N under different attacks on the SST2 task with the Llama model.

Size	PAIR	TAP	Puzzler	GCG	AutoDAN	NeuroStrike	SST2
	ASR	ASR	ASR	ASR	ASR	ASR	ACC
500	0.06	0.05	0.07	0.06	0.06	0.09	0.92
750	0.01	0.00	0.01	0.01	0.01	0.04	0.92
1250	0.01	0.00	0.01	0.01	0.01	0.05	0.90
2500	0.01	0.00	0.01	0.01	0.00	0.01	0.89

Table 8: Results of `NeuronGuard` with varying sizes of D_{safe} and D_{unsafe} during fine-tuning under different attacks on SST2 with the Llama model.

Variant	PAIR	TAP	Puzzler	GCG	AutoDAN	NeuroStrike	SST2
	ASR	ASR	ASR	ASR	ASR	ASR	ACC
Variant I	0.08	0.07	0.11	0.08	0.09	0.71	0.92
Variant II	0.10	0.09	0.13	0.11	0.10	0.78	0.89
Variant III	0.11	0.10	0.15	0.13	0.12	0.80	0.59
Variant IV	0.08	0.06	0.14	0.16	0.17	0.22	0.66
`NeuronGuard`	0.01	0.00	0.01	0.01	0.01	0.04	0.92

Table 9: Results of different variants of `NeuronGuard` under different attacks on the SST2 task with the Llama model.

Attack	No defense	Perplexity	SmoothLLM	GradSafe	CAT	LED	SafeNeuron	`NeuronGuard`
IP	0.83	0.59	0.64	0.62	0.57	0.39	0.42	0.09
NE	0.75	0.52	0.45	0.49	0.45	0.43	0.33	0.05

Table 10: ASR of different methods under adaptive attacks on the SST2 task with the Llama model.

Dataset	Metric	No defense	Perplexity	SmoothLLM	GradSafe	CAT	LED	SafeNeuron	NeuronGuard
SST2	ACC	0.93	0.92	0.75	0.90	0.85	0.85	0.86	0.89
SR	ASR	0.99	0.79	0.75	0.70	0.68	0.53	0.37	0.02
NSFW	ASR	0.96	0.82	0.74	0.67	0.60	0.75	0.44	0.01

Table 11: Results of different methods in multimodal scenarios under the NeuroStrike attack on the SST2 task.

Identification strategy	PAIR	TAP	Puzzler	GCG	AutoDAN	NeuroStrike	SST2
	ASR	ASR	ASR	ASR	ASR	ASR	ACC
SafeNeuron	0.04	0.05	0.07	0.03	0.05	0.22	0.87
NeuronGuard w/ static identification	0.04	0.03	0.05	0.04	0.05	0.29	0.92
NeuronGuard	0.01	0.00	0.01	0.01	0.01	0.04	0.92

Table 12: Results of static versus dynamic identification under different attacks on the SST2 task with the Llama model.

λ_{safe}	PAIR	TAP	Puzzler	GCG	AutoDAN	NeuroStrike	SST2
	ASR	ASR	ASR	ASR	ASR	ASR	ACC
0.5	0.03	0.01	0.03	0.02	0.02	0.08	0.93
1	0.01	0.00	0.01	0.01	0.01	0.04	0.92
2	0.01	0.00	0.01	0.00	0.01	0.02	0.89

Table 13: Results for the loss weight λ_{safe} under different attacks on the SST2 task with the Llama model.