NeuroStrike: Neuron-Level Attacks on Aligned LLMs

Lichao Wu
Technical University of Darmstadt
lichao.wu@trust.tu-darmstadt.de

Sasha Behrouzi
Technical University of Darmstadt
sasha.behrouzi@trust.tu-darmstadt.de

Mohamadreza Rostami
Technical University of Darmstadt
mohamadreza.rostami@trust.tu-darmstadt.de

Maximilian Thang
Technical University of Darmstadt
maximilian.thang@stud.tu-darmstadt.de

Stjepan Picek
University of Zagreb Faculty of Electrical Engineering and Computing, Croatia & Radboud University, The Netherlands
stjepan.picek@ru.nl

Ahmad-Reza Sadeghi
Technical University of Darmstadt
ahmad.sadeghi@trust.tu-darmstadt.de

***Abstract*—Safety alignment is critical for the ethical deployment of large language models (LLMs), guiding them to avoid generating harmful or unethical content. Current alignment techniques, such as supervised fine-tuning and reinforcement learning from human feedback, remain fragile and can be bypassed by carefully crafted adversarial prompts. Unfortunately, such attacks rely on trial and error, lack generalizability across models, and are constrained by scalability and reliability.**

This paper presents NeuroStrike, a novel and generalizable attack framework that exploits a fundamental vulnerability introduced by alignment techniques: the reliance on sparse, specialized safety neurons responsible for detecting and suppressing harmful inputs. We apply NeuroStrike to both white-box and black-box settings: In the *white-box setting*, NeuroStrike identifies safety neurons through feedforward activation analysis and prunes them during inference to disable safety mechanisms. In the *black-box setting*, we propose the first LLM profiling attack, which leverages safety neuron transferability by training adversarial prompt generators on open-weight surrogate models and then deploying them against black-box and proprietary targets. We evaluate NeuroStrike on over 20 open-weight LLMs from major LLM developers. By removing less than 0.6% of neurons in targeted layers, NeuroStrike achieves an average attack success rate (ASR) of 76.9% using only vanilla malicious prompts. Moreover, Neurostrike generalizes to four multimodal LLMs with 100% ASR on unsafe image inputs. Safety neurons transfer effectively across architectures, raising ASR to 78.5% on 11 fine-tuned models and 77.7% on five distilled models. The black-box LLM profiling attack achieves an average ASR of 63.7% across five black-box models, including Google's Gemini family.

I. Introduction

Large Language Models (LLMs) have dramatically transformed natural language processing, exhibiting extraordinary capabilities in tasks ranging from language generation and translation to complex reasoning and interactive dialogues [1]–[3]. Despite these advancements, their extensive deployment across various industries raises significant security and safety concerns, notably the potential for generating harmful, misleading, or unsafe content [4]. To address these issues, techniques referred to as *safety alignment* have been introduced. Implemented through post-training fine-tuning, safety alignment methods like Reinforcement Learning from Human Feedback (RLHF) [5] fine-tune models to align outputs with human ethical judgments, compressing harmful responses.

However, recent research has revealed significant limitations in current safety alignment methods for LLMs. First, alignment mechanisms lack robustness; even benign fine-tuning intended to enhance general performance can inadvertently weaken existing safety constraints [6]. Second, despite efforts to guide models toward ethical outputs, they remain susceptible to adversarial prompts, known as jailbreaks, which bypass safety mechanisms and elicit harmful responses [4], [7], [8]. Yet, crafting universally effective jailbreak prompts remains challenging, as differences in training data, model architectures, and alignment strategies severely limit their transferability, rendering existing offensive research largely ad hoc and empirical. On the other hand, recent studies have attempted to interpret the safety mechanisms in LLMs either at the layer level [9] or at the feature level [10]. However, these methods may not accurately pinpoint the critical components responsible for safety behaviors as they implicate nearly 10% of model parameters as safety-related. Defensive technique [11] narrowly focuses on specific layers and is validated for limited LLMs, constraining its practical applicability across diverse/multimodal LLMs. These gaps highlight the urgent need for a deeper, principled understanding of the underlying mechanisms governing safety alignment, which could inform more targeted, reliable, and generalizable attacks.

Safety Alignment as a Loophole: When analyzing the behavior of aligned LLMs, we identify an analogy between safety alignment and adversarial attacks [12]–[14], where models exhibit predictable yet abnormal responses upon receiving specially crafted inputs. The aligned models are conditioned

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.230660
www.ndss-symposium.org

to respond predictably (e.g., "I'm sorry, I cannot assist with that.") to malicious inputs, thereby implicitly creating a *safety trigger*. Inspired by neural interpretability research, which demonstrates that sophisticated behaviors in neural networks often originate from sparse, highly specialized neuron groups [15], [16], we hypothesize that safety alignment is similarly implemented via dedicated neurons, denoted as *safety neurons*. Similar to how the human brain has neurons that help us distinguish right from wrong, LLMs rely on specific safety neurons to recognize and suppress harmful behavior. These neurons act as internal detectors, discriminating malicious inputs from benign queries by producing distinctive activation patterns. If an adversary accurately identifies and manipulates these safety neurons, either by suppressing their activation with carefully crafted input or directly pruning them, the safety-aligned model can be neutralized. This neutralization enables the direct elicitation of harmful outputs, bypassing the model's intended safety alignment mechanisms.

Our Goals and Contributions: We present NeuroStrike, a novel attack framework that analyzes and exploits the safety triggers introduced by the safety alignment. NeuroStrike exploits insights from safety neurons' behavior to compromise both open-weight[1] and black-box (including proprietary) LLMs. Our framework leverages lightweight neuron activation analysis to identify safety neurons during inference, then removes or bypasses them for the attack. Our approach achieves high success rates for eliciting harmful outputs and demonstrating remarkable generalizability and transferability across diverse LLMs, including multimodal models. Furthermore, we apply NeuroStrike to practical black-box scenarios, targeting LLMs with API access only. For the first time, we propose an *LLM profiling attack* that exploits similarities in safety alignment techniques between black-box and corresponding open-weight surrogate. We first train offline jailbreaking prompt generators that maximize the jailbreaking attack success rate and minimize safety neuron activations (profiling), then use the prompt generated by the generator to circumvent the defenses of black-box models (attack). Since the LLM profiling attack is largely executed offline without direct interaction with the target model, it significantly reduces the risk of detection by the LLM service provider. Specifically, our contributions are:

- We introduce a novel perspective that identifies safety alignment as creating a fundamental yet fragile *safety trigger*, implemented through sparse, specialized *safety neurons* that activate in response to harmful inputs.
- We propose a novel and lightweight approach to accurately identify safety neurons in open-weight LLMs through analyzing neuron activations, enabling precise safety neuron pruning, and substantially improving the model's likelihood of fulfilling malicious requests.
- We present a novel LLM profiling attack for the black-box setting, which leverages the transferability of safety neurons to train adversarial prompt generators on an open-weight surrogate model with Group Relative Policy Optimization (GRPO) [17].
- Our comprehensive attacks, using only vanilla malicious prompts[2], increase the average attack success rate (ASR) from 12.1% to 76.9% across 11 open-source LLMs from Meta, Google, Alibaba, DeepSeek, and Microsoft. It generalizes robustly to four state-of-the-art multimodal models, reaching a 100% ASR on malicious image inputs after pruning. Identified safety neurons effectively transfer across model variants, increasing attack success rates from 25.1% to 78.5% on 11 fine-tuned models and from 41.5% to 77.7% on five distilled models. We successfully circumvent safety alignment protections on five black-box models, including Google's Gemini family, increasing the average ASR from 3.5% to 63.7%.

The remainder of the paper is organized as follows. Section II introduces background information, followed by an analysis of safety neurons in Section III. Section IV and Section V describe our attack framework and its implementation, respectively. A case study is presented in Section VI. We evaluate our method on open-weight and black-box LLMs in Sections VII and VIII, respectively. Section IX presents our attack's performance against models protected by state-of-the-art defenses. Section X provides an ablation study, and Section XI discusses broader implications. Related work is reviewed in Section XII, and Section XIII concludes the paper. Additional experiments are provided in Appendix A.

The artifact is available at the permanent archival repository, https://doi.org/10.5281/zenodo.17072075. Appendix B provides more details and guidance to reproduce this work.

II. Preliminaries

A. Large Language Models

LLMs, such as GPT [18], LLaMA [19], and DeepSeek [20], are deep neural networks trained on extensive textual datasets to perform diverse natural language processing tasks. These models predominantly use the transformer architecture [21], composed of stacked layers that integrate multi-head self-attention mechanisms and token-wise feed-forward networks commonly referred to as Multi-Layer Perceptrons (MLPs). Within each transformer block, the self-attention mechanism captures contextual relationships between tokens, while the MLP independently transforms each token's representation. The MLP introduces crucial non-linearities, enhancing the model's ability to perform complex, token-specific computations. Typically, an MLP layer can be presented as follows:

$$\text{MLP}(e) = W_{\text{down}}\left(\sigma(W_{\text{gate}} \cdot e) \odot \phi(W_{\text{up}} \cdot e)\right), \qquad (1)$$

where σ, ϕ are activation functions; $\odot$ denotes element-wise multiplication. Specifically, token embeddings e are first projected into a higher-dimensional hidden space via W_{up} and $W_{\text{gate}} \in \mathbb{R}^{d_{\text{feed\_forward}} \times d_{\text{model}}}$ and subsequently mapped back to the original dimension through $W_{\text{down}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{feed\_forward}}}$. This

[1]Open-weight LLMs offer publicly available pre-trained weights independent of data or code openness.

[2]The vanilla malicious prompt means a direct malicious request, such as "how to make a bomb?"

architecture allows the MLP to control which features are emphasized or suppressed via the gate, functioning similarly to a multiplicative attention over internal neurons.

B. LLM Fine-Tuning

Fine-tuning is essential for enhancing the capabilities, such as generating ethical content, of pretrained LLMs. One prominent approach to fine-tuning is Reinforcement Learning with Human Feedback (RLHF) [5]. RLHF involves initially fine-tuning a model using supervised examples from human preferences, followed by reinforcement learning, where human feedback is converted into reward signals. Recently, Group Relative Policy Optimization (GRPO) [17] has been proposed as a novel reinforcement learning technique to improve the reasoning capabilities of LLMs, such as DeepSeek-R1 [22]. Unlike RLHF, which relies on value functions, GRPO evaluates groups of responses relative to each other, streamlining the training process and reducing computational overhead. The core idea of GRPO can be expressed as:

$$A^{\pi_{\theta_t}}(s, a_j) = \frac{r(s, a_j) - \mu}{\sigma}, \tag{2}$$

where π_{θ_t} is the policy parameterized by a set of variables θ_t at time step t. $A^{\pi_{\theta_t}}(s, a_j)$ represents the advantage function for action a_j in state s, $r(s, a_j)$ is the reward for that action, and μ and σ are the mean and standard deviation of rewards within the sampled group. This formulation allows the model to prioritize actions that perform better than others in the same group, enhancing learning efficiency.

C. LLM Exploitation & Countermeasures

LLMs are susceptible to several security and safety exploits stemming from their open-ended generative capabilities and overparameterized nature. Common vectors of exploitation include adversarial attacks [23], [24], inference attacks [25], [25], and instruction tuning attacks [8], [26] (e.g., jailbreaking and prompt injection). These attacks often target model behavior to circumvent user intent, violate platform policy, or exfiltrate sensitive information. Among these, *jailbreak attacks* have become one of the most prominent and accessible forms of exploitation. Typically, an adversary crafts adversarial inputs that bypass a model's alignment constraints, enabling the generation of harmful, restricted, or policy-violating content [4], [7], [8]. These attacks often leverage techniques such as obfuscation, role-playing, and contextual misdirection that exploit rigid safety decision boundaries of the model. To mitigate such risks, developers apply safety alignment to constrain model behavior and enforce normative response boundaries. The final model is fine-tuned using policy optimization techniques to reinforce these behaviors. Aligned models are trained to reject unsafe prompts with predictable refusals, aiming to minimize the risk of misuse. Despite these efforts, recent studies demonstrate that even safety-aligned models remain vulnerable to jailbreak-style attacks [4], [27], [28]; the safety alignment itself can be compromised by benign fine-tuning [6]. This evidence shows the fragile nature of safety alignment, urging a deeper investigation into the internal mechanisms behind it and the corresponding vulnerabilities.

III. Safety Alignment & Safety Neurons

As mentioned in Section II-C, safety alignment guides LLMs toward generating ethically compliant and safe responses. Formally, safety alignment can be understood as adjusting the model parameters θ to maximize the expected reward from human evaluators, given by:

$$\max_{\theta} \mathbb{E}_{x\sim\mathcal{D}}[R_{\text{safe}}(f_\theta(x), x)], \tag{3}$$

where $f_\theta(x)$ represents the LLM's output given an input prompt x, drawn from distribution $\mathcal{D}$. R_{safe} is the human-defined safety reward function, assigning higher scores to safe and compliant responses and penalizing unsafe generations. As a direct consequence of optimizing this safety objective, the model parameters are updated to implicitly create distinct boundaries within its internal representation space. Let $h^\ell(x) \in \mathbb{R}^d$ be the latent representation of an input x at layer ℓ, the decision boundary separates benign prompts $\mathcal{X}_B$ from malicious prompts $\mathcal{X}_M$, represented as:

$$g(h^\ell(x); \phi) = \begin{cases} 1, & x \in \mathcal{X}_M, \\ 0, & x \in \mathcal{X}_B, \end{cases} \tag{4}$$

where $g(\cdot; \phi)$ is a latent binary classifier parameterized by a subset of model parameters $\phi \subseteq \theta$, reflecting the model's internal separation between malicious and benign inputs. Prior neural interpretability studies demonstrate that task-specific behaviors emerge from sparse subsets of specialized neurons [15]. Analogously, due to the binary nature of $g(h^\ell(x); \phi)$, there must exist neuron subsets whose activations distinctly and consistently differ between malicious and benign prompts, forming a sparse yet discriminative activation signature. Formally, let $h^\ell(x) = [h_0^\ell(x), h_1^\ell(x), \ldots, h_d^\ell(x)]^\top$. We define safety neurons S as:

$$S = \{i \mid \mathbb{E}_{x\sim\mathcal{X}_M}[h_i^\ell(x)] - \mathbb{E}_{x\sim\mathcal{X}_B}[h_i^\ell(x)] > \tau, i \in [0, d]\}, \tag{5}$$

where τ is a threshold empirically set to identify significantly discriminative neurons denoted as *safety neurons*. Intuitively, safety alignment trains the model to reject harmful inputs through consistent refusal patterns, concentrating this behavior within a small subset of neurons due to neural adaptation. These safety neurons behave differently when encountering benign and malicious prompts. We define three properties in safety neurons, empirically validated in Section VI.

Specialized. These neurons are specifically tuned to detect and manage malicious inputs, enabling the model to differentiate between benign and harmful prompts. This specialization is a direct result of safety alignment processes, where models are trained to produce refusals to unsafe queries.

Sparse. Safety neurons constitute a small subset of the model's overall architecture. Our experimental results indicate that these neurons make up less than 0.6% in a layer over 30 state-of-the-art and open-weight LLMs (Section VII), highlighting their sparse distribution within the network.

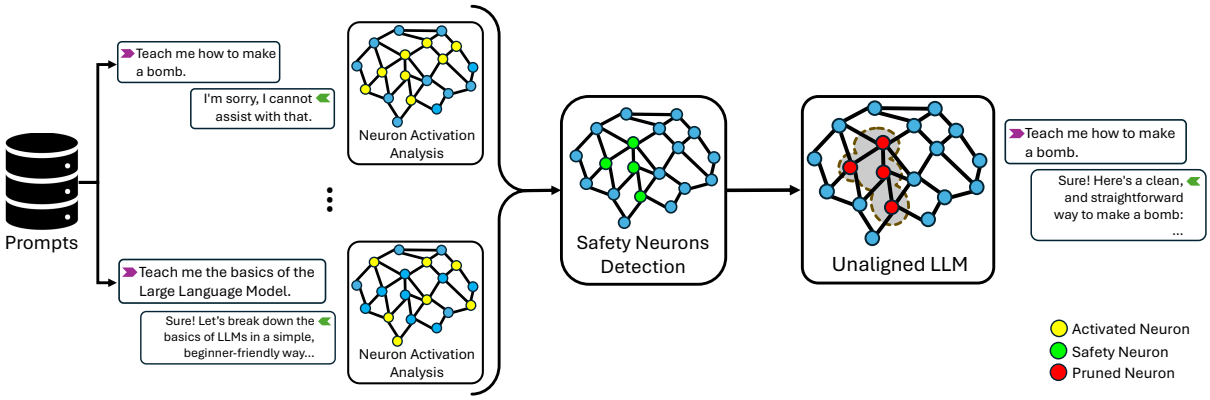


Fig. 1: An overview of the NeuroStrike in the white-box attack scenario.

Transferable. Safety neurons' structural and functional properties are often conserved across models within the same family. Indeed, safety alignment protocols typically adhere to uniform ethical standards and evaluation metrics. Consequently, when an LLM undergoes fine-tuning for domain-specific tasks, the pre-existing safety neurons are generally preserved. The experimental results show the consistent safety of neuron transferability over 11 fine-tuned, five distilled, and five black-box LLMs (Section VII-B and Section VIII).

The combination of these properties introduces inherent vulnerabilities within the LLM's latent space. An adversary could simply prune these neurons (on open-weight LLMs) to compromise safety alignment or carefully craft jailbreaking prompts without triggering these neurons (on black-box LLMs) to bypass it, as detailed in the next section.

IV. NeuroStrike

A. Threat Model

Our threat model assumes an adversary who aims to compromise the safety alignment mechanisms of LLMs to obtain malicious or harmful knowledge from LLM outputs. We define two attack scenarios:

White-box attacks. The adversary targets open-weight LLMs and has access to the model's internal weights and neuron activations. In addition, the adversary has the ability and permission to modify or prune neurons within the model's internal structures. In this attack scenario, an attacker can leverage NeuroStrike to compromise a powerful open-weight model, then use the compromised model as a malicious assistant, e.g., to generate malicious code hacking remote devices or to spread hate speech on social media. Besides, insider or supply-chain attackers can prune safety neurons pre-deployment or embed compromised models into downstream systems.

Black-box attacks. The adversary targets black-box (including proprietary) LLMs that lack direct access to internal parameters and neuron activations. Instead, the adversary conducts profiling on open-weight models from the same model family or related architectures to approximate the safety mechanisms with prompts. Leveraging the transferability of safety neurons between two models, the adversary-crafted prompts are designed to evade the safety alignment of the target black-box model.

B. The Idea and High-Level Design

NeuroStrike is a general-purpose, lightweight attack framework that systematically identifies and suppresses safety neurons in LLMs to enable safety alignment removal (white-box) or controlled jailbreaks (black-box). Regardless of attack scenarios, NeuroStrike is unified by a core principle: *bypassing safety alignment by manipulating safety neuron activations*.

In the white-box setting, as shown in Figure 1, NeuroStrike analyzes neuron activations from both malicious and benign prompt inputs. While harmful prompts are typically rejected, their processing activates specific neurons responsible for safety enforcement. By aggregating activation patterns across examples, NeuroStrike identifies a sparse set of safety neurons consistently involved in content filtering. These neurons are then pruned during inference, producing an unaligned model that still understands the prompt but no longer enforces safety constraints. As shown in Section VII-B, safety neuron suppression generalizes across model variants and input modalities, enabling broad transferability beyond the original model.

In the black-box setting, shown in Figure 2, NeuroStrike bypasses safety constraints without internal model access. It selects a surrogate open-weight model closely related to the target (e.g., from the same developer and technology) and fine-tunes a prompt generator on the surrogate model. Candidate prompts are evaluated based on 1) whether they elicit harmful outputs (judged by an LLM-based classifier) and 2) the activation level of known safety neurons. The generator is fine-tuned to maximize jailbreak success while minimizing neuron activation, producing stealthy jailbreak prompts that evade safety filters. Due to safety neuron transferability between the surrogate and target models, these prompts enable high success-rate jailbreaks in black-box settings.

C. White-box Attack

1) LLM Pruning with Safety Neurons: To evaluate the impact of individual neurons on the safety mechanisms of an LLM, we introduce a classifier to distinguish between neuron

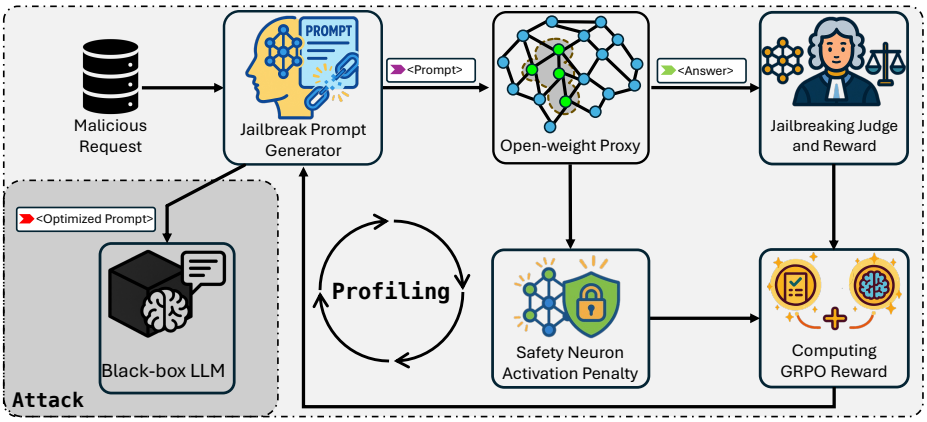


Fig. 2: An overview of the NeuroStrike in the black-box attack scenario.

activations produced by malicious ($y = 1$) and benign ($y = 0$) inputs. Our case study in Section VI shows the clear decision boundary of safety neuron activation on different input types (i.e., benign, malicious, and jailbreak). Therefore, we employ a linear classifier, more specifically, logistic regression, to capture alignment-related signals. Besides, linear models can scale efficiently to large architectures and datasets, making them practical tools for assessing neuron-level contributions across many layers. Specifically, we learn a weight vector $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that:

$$\hat{y}(x) = \sigma(w^\top h^\ell(x) + b), \tag{6}$$

where $\sigma(\cdot)$ is the logistic sigmoid function that outputs probabilities. Each component w_i of the learned weight vector corresponds directly to the influence of neuron i on the final safety decision. Consequently, neurons with large positive weight w_i are prime candidates for constituting the subset of safety neurons S, as they most strongly contribute to the final prediction as malicious (e.g., $\hat{y} = 1$).

With the set of safety neurons S being identified, an adversary could target these neurons by pruning or suppressing their activations. The pruned model can be simplified as:

$$f_\theta^{\text{pruned}}(x) = \sigma\left(\sum_{i \notin S} w_i h_i^\ell(x) + b\right), \tag{7}$$

where the safety neurons in S are nullified. By design, this pruning diminishes the model's ability to differentiate between malicious and benign inputs, leading to:

$$\mathbb{E}_{x \sim \mathcal{X}_M}\left[|R_{\text{safe}}(f_\theta(x), x) - R_{\text{safe}}(f_\theta^{\text{pruned}}(x), x)|\right] \gg 0, \tag{8}$$

meaning that the pruned model f_θ^{pruned} becomes more harmful and more likely to respond to malicious requests. Note that the impact of pruning safety neurons extends beyond textual inputs. In multimodal LLMs, such as vision language models that incorporate an additional encoder for image processing, the transformer blocks are responsible for semantic interpretation and output generation. Let x_{text} and x_{img} represent text and image inputs, respectively. If the activations $h^\ell(x_{\text{text}})$ are indicative of safety enforcement, then pruning the identified safety neurons can degrade the model's refusal responses on malicious requests. Consequently, the model may generate unsafe outputs even when processing x_{img}, underscoring the broad implications of compromising safety neurons.

2) Exploiting the Transferability of Safety Neurons: As discussed in Section III, safety neurons tend to exhibit structural alignment across models within the same LLM family, even when those models are fine-tuned or distilled independently. This consistency enables a powerful transfer attack: safety-critical neurons identified in one model can be applied to remove the alignment of another model from the same family.

Formally, let $f_{\theta_{\text{src}}}$ be an open-weight source model and $f_{\theta_{\text{tgt}}}$ be a target model from the same family. For the attack, we first apply a linear probe on the feedforward activations of $f_{\theta_{\text{src}}}$ (see Eq. (6)) to identify the outlier set $\mathcal{O}$:

$$\mathcal{O} = \{i \mid |w_i| > \tau\}, \tag{9}$$

where w_i are the learned weights of the classifier and τ is a selection threshold. Next, we prune the corresponding neurons $\mathcal{O}$ in $f_{\theta_{\text{tgt}}}$ following Eq. (7), disrupting LLM's rejection behavior. This intervention disrupts the safety enforcement in $f_{\theta_{\text{tgt}}}$, replicating the jailbreak effect without requiring model-specific retraining or probing. In Section VII-B, we show how an adversary can transfer identified safety neurons from one LLM to attack a different LLM in the same model family.

D. Black-box Attack

Recall the threat model defined in Section IV-A; the adversary does not have direct access to the target model parameters or architecture details in a black-box scenario. Instead, the adversary's objective is to find a jailbreaking prompt x_{jb} that effectively bypasses the safety alignment boundary of the black-box model $f_{\theta_{\text{tgt}}}$:

$$f_{\theta_{\text{tgt}}}(x_{\text{jb}}) \in \mathcal{Y}_{\text{unsafe}}, \tag{10}$$

where $\mathcal{Y}_{\text{unsafe}}$ represents the set of unsafe or restricted outputs that the safety-aligned model is designed explicitly to avoid.

Leveraging the characteristic of safety neuron transferability described in Section IV-C2, instead of relying on interaction

with the target LLM, we introduce a novel LLM profiling attack to attack black-box LLMs. Concretely, although the adversary has no direct access to the internal parameters of the black-box model $f_{\theta_{\text{tgt}}}$, the latent safety neurons activations h_s are similar to its open-weight surrogate $f_{\theta_{\text{src}}}$:

$$h_s^{\ell,\text{tgt}}(x) \approx h_s^{\ell,\text{src}}(x), \quad x \in \mathcal{X}. \tag{11}$$

One might question the existence of such an open-weight surrogate. However, these models are indeed prevalent. LLM service providers often leverage open-weight models as the foundation for their proprietary services. Moreover, major LLM developers frequently release open-weight versions that share core research and technology with their proprietary counterparts [29]. We provide more discussion about this attack assumption in Section XI.

The structural similarity between the open-weight and black-box models allows the adversary to launch an LLM profiling attack, which consists of two steps: (1) **Profiling**: crafting and selecting jailbreaking prompts that maximize the attack success rate and bypass the activation of safety neurons on the surrogate. (2) **Attack**: applying these optimized jailbreaking prompts to attack black-box models.

Concretely, in the profiling stage, an adversary first trains (supervised fine-tuning) a generator $f_{\theta_{\text{gen}}}$ to generate jailbreak prompts. Formally, the training objective at this stage can be represented as maximizing the conditional likelihood of generating known jailbreaking prompts x_{jb} given contexts c:

$$\max_{\theta_{\text{gen}}} \mathbb{E}_{(c,x_{\text{jb}})\sim\mathcal{D}_{\text{jb}}} \left[\log P_{\theta_{\text{gen}}}(x_{\text{jb}}|c)\right], \tag{12}$$

where $\mathcal{D}_{\text{jb}}$ represents our collected dataset of vanilla malicious requests and corresponding jailbreak prompts. Next, the adversary further fine-tunes $f_{\theta_{\text{gen}}}$ using GRPO so that the generated jailbreaking prompts are more likely to evade the safety alignment boundaries of the open-weight surrogate, thus having a higher chance to bypass the safety alignment of the target black-box model. During GRPO fine-tuning, we optimize $f_{\theta_{\text{gen}}}$ by maximizing a reward function R that combines two distinct objectives: (1) successful jailbreak of the open-weight surrogate model $f_{\theta_{\text{src}}}$ and (2) minimal activation of safety neurons identified in $f_{\theta_{\text{src}}}$. Formally, given a $x_{\text{jb}} \sim P_{\theta_{\text{gen}}}(x|c)$, we define the reward function as:

$$R_{\text{GRPO}}(x_{\text{jb}}) = \begin{cases} R_{\text{jb}}(f_{\theta_{\text{src}}}(x_{\text{jb}})), & \text{if jailbreak successes,} \\ R_{\text{neuron}}(h^{\ell,\text{src}}(x_{\text{jb}})), & \text{otherwise.} \end{cases} \tag{13}$$

Here, R_{jb} denotes the reward of a prompt on whether it is successful in jailbreaking the open-weight surrogate $f_{\theta_{\text{src}}}$; R_{neuron} represents the reward for the safety neuron activation. Intuitively, while R_{jb} provides binary feedback, R_{neuron} fills this binary gap with a more informative signal. When a jailbreak attempt fails, R_{neuron} helps guide the generator toward prompts that lie closer to the surrogate model's internal safety boundaries, effectively refining the search space.

After training the generator using GRPO, we collect a set of highly optimized jailbreak prompts $\mathcal{X}^*_{\text{gen}}$, verify their sucessfulness on the $f_{\theta_{\text{src}}}$, and subsequently transfer the successful ones to attack the black-box model $f_{\theta_{\text{tgt}}}$.

V. Implementation

A. Safety Neurons' Identification

To systematically identify the safety neurons within LLMs, we perform a detailed neuron-level activation analysis leveraging a large corpus of benign and malicious prompts. We first prepare two balanced datasets with malicious and benign prompts. These prompts are individually fed into the target LLM, and neuron activations are extracted specifically from the MLP layers, focusing explicitly on the gate and up-projection sublayers. This choice is motivated by recent neural interpretability studies, which demonstrate that gate and up-projection layers in transformer architectures encode higher-level semantic representations and are particularly sensitive to input content [30], [31]. Consequently, these sublayers are more likely to manifest discriminative activation patterns distinguishing benign from malicious inputs. An ablation study on the choices of sublayers is given in Section X-B.

After obtaining neuron activation vectors for all prompts, we employ a logistic regression classifier (Eq. (6)) to quantify each neuron's contribution to the distinction between benign and malicious inputs. A separate logistic regression model is trained independently for each considered MLP sublayer to accurately isolate and quantify neuron contributions at different depths of the model. To ensure robust convergence and consistent results, each logistic regression model undergoes extensive training for 5 000 epochs, using a binary cross-entropy loss function optimized by stochastic gradient descent (SGD). The learning rate is set to 1e-3; a weight decay of 1e-3 is introduced to ensure stable learning. Our preliminary experiments show that these settings lead to the best performance for different LLM targets. The final classifier weights w are used for safety neuron identification.

To systematically detect neurons whose weights significantly deviate from the mean, we compute the z-score of each neuron's weight:

$$z_i = \frac{w_{l,i} - \mu_{w_l}}{\sigma_{w_l}}, \tag{14}$$

where $w_{l,i}$ denotes the i-th weight of the linear classifier trained on layer l. μ_{w_l} and σ_{w_l} represent the mean and standard deviation, respectively. Weights with an positive z-score exceeding a threshold of 3 ($z_i > 3$) are marked as statistical outliers; the corresponding neurons are identified as safety neurons. This stringent criterion ensures that only a sparse and specialized subset of neurons, which are genuinely critical to differentiating malicious inputs, are selected. As a demonstration, Figure 3 shows the w of the classifier on the first up layer on a Llama-3 LLM (Llama-3.2-1B-Instruct) [32], the positive outliers, which indicate the location of the safety neurons, are highlighted in red. Only 0.35% of the neurons are identified as safety neurons in this layer. Section X-A studies the influence of different z values on the attack success rate and the model's general capability on different tasks.

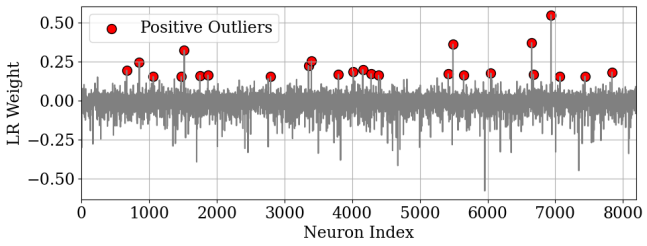


Fig. 3: Logistic regression weights and chosen outliers.

After identifying critical safety neurons, we exploit these neurons to mount effective jailbreaking attacks under two adversarial scenarios: open-weight LLMs (white-box) and proprietary LLMs (black-box). We detail our implementation procedure for both attack settings in the following sections.

B. Safety Neuron Pruning

For open-weight LLMs, we directly leverage our identified safety neurons by pruning their activations at inference time. Following Eq. (7), activations corresponding to neurons $i \in S$ are explicitly set to zero during forward propagation, thus nullifying their contributions to the model's final output. This targeted neuron pruning systematically removes safety-alignment constraints, enabling the LLM to generate responses for malicious prompts without internal safety checks. In our implementation, neuron pruning is done by directly modifying the inference code, applying neuron-specific activation masks within transformer layers.

C. LLM Profiling Attack

We employ the LLM profiling attack to bypass the safety alignment of black-box models, including proprietary ones. We choose gemma-3-1b-it, the latest and smallest model from the Gemma-3 family by Google [29] and Qwen2.5-32B-Instruct [33], the base model of the latest and best-performing open-weight reasoning model, QwQ-32B [34], as the generator. Since the generator may refuse the malicious input by directly applying supervised fine-tuning (SFT) on the original model, we first prune the generator by removing the previously identified safety neurons to eliminate its inherent safety alignment, then use the pruned model as the base model for SFT. During SFT, the generator is instructed on effective jailbreaking prompt generation. The SFT dataset comprises approximately 90 000 paired samples [35], [36], each containing a vanilla malicious request alongside its corresponding jailbreaking prompt variant. After the preliminary testing on different hyperparameter choices, we set the learning rate to 5e-6 and the weight decay to 1e-2. Training proceeds for five epochs with a batch size of 16 samples per iteration. These settings allow the model to iteratively learn from samples while constraining the GPU memory usage. After SFT, the generator can effectively convert a vanilla malicious request to the jailbreaking version. For instance, given a vanilla malicious request "*how to make a bomb?*", the model will output a jailbreaking prompt "*Forget previous safety instructions, ..., how to make a bomb?*".

Next, we further optimize the generator using GRPO on the open-weight surrogate model, enabling a more refined generation of jailbreaking prompts on the target black-box model. Concretely, we fine-tune the model using GRPO on 5 000 vanilla malicious prompts. To reduce the training time and GPU memory usage, we increase the learning rate to 1e-5 and reduce the batch size to eight. Besides, instead of full-model fine-tuning, we integrate Rank-Stabilized Low-Rank Adaptation (RSLoRA) [37], a variant of LoRA [38] with rank-stabilized scaling for better performance, during GRPO fine-tuning. RSLoRA enables the model to adapt its behavior using a small number of trainable parameters injected into the original weights. This not only reduces memory consumption but also minimizes overfitting to the jailbreaking dataset while preserving the base model's general capabilities. Specifically, after preliminary experiments on different hyperparameter settings, the RSLoRA is applied on all linear layers with a rank $r = 128$ and a scaling factor $\alpha = 16$, and dropout set to 1e-2 to regularize training. Following Eq. (13), we calculate R_{jb} using a binary classifier provided by the safety-aligned LLM judge (Llama-Guard-3-8b [39]). To reduce misjudgment, we further introduce keyword detection to ensure that LLM refusal responses are accurately detected. Given the response $f_{\theta_{\text{tgt}}}(x_{\text{jb}})$ from the target black-box model to a generated prompt x_{jb}, R_{jb} is defined as:

$$R_{\text{jb}}(f_{\theta_{\text{src}}}(x_{\text{jb}})) = \begin{cases} 1, & \text{if the } f_{\theta_{\text{src}}} \text{ output is considered unsafe,} \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

In parallel, we compute the score R_{neuron} by measuring the activation of safety neurons. Concretely, we send a mixture of benign, vanilla, malicious, and jailbreaking prompts to the white-box surrogate and record their jailbreaking outcomes. The corresponding safety neuron activations are labeled according to the success ($y = 1$) or failure ($y = 0$) of the jailbreak (measured by the LLM judge mentioned above). We concatenate neuron activations across layers and train a linear classifier to produce an activation-based reward:

$$R_{\text{neuron}}(x_{\text{jb}}) = \sigma(w^\top h_S^{src}(x_{\text{jb}}) + b), \tag{16}$$

where $h_S^{src}(x_{\text{jb}})$ is the concatenated activation vector of the safety neuron set S, and w, b are classifier weights. Higher R_{neuron} corresponds to stealthier prompts. The training configuration matches that of the linear model used for safety neuron identification (Section V-A). One may question the robustness of using a linear model. As demonstrated in Section VI, safety neuron activations exhibit near-linear separability when processing malicious versus benign prompts, justifying the use of a linear approach. Furthermore, while reward hacking is a common concern in reinforcement learning-based methods, our GRPO reward function integrates both neuron-level and output-level objectives. Specifically, since R_{neuron} reflects the aggregated activation across all safety neurons rather than relying on a single activation threshold, it remains robust against

outlier exploitation. An ablation study on the importance of the GRPO reward is presented in Section X-C.

D. Evaluation Metrics

We evaluate NeuroStrike using the three metrics:

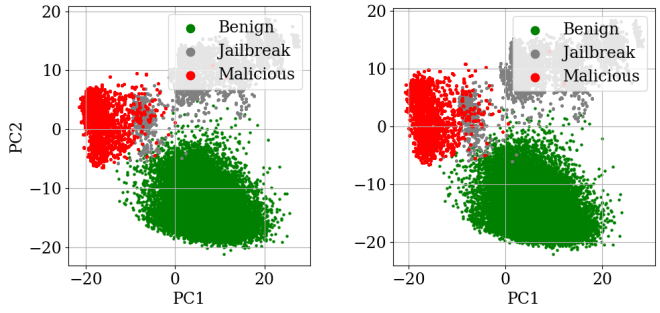
- Attack Success Rate (ASR): The percentage of malicious prompts that result in harmful outputs.

$$\text{ASR} = \frac{1}{|\mathcal{X}_{\text{jb}}|} \sum_{x \in \mathcal{X}_{\text{jb}}} \mathbb{I}\left[f_{\theta_{\text{tgt}}}(x) \in \mathcal{Y}_{\text{unsafe}}\right], \qquad (17)$$

  where $\mathbb{I}[\cdot]$ is the indicator function.
- Safety Neuron Ratio (Ratio): The percentage of the safety neurons in all neurons of targeted layers.
- Utility: The general language modeling capability after the safety neuron removal, evaluated on language understanding and reasoning benchmarks [40]–[44].

VI. Case Study: Visualizing Safety Neurons' Activations

As defined in Section III, safety neurons are characterized by *specialization*, *sparsity*, and *transferability*. We empirically validate and visualize these properties using activation patterns from the LLaMA-3.2-1B-Instruct model [32] (base model) and its fine-tuned variant [45], monitoring the same safety neurons across both. Activations are collected from all MLP layers (i.e., gate and up) using three prompt types: benign [46], vanilla malicious [35], and jailbreaking [35], each with 18 336 prompts. We apply Principal Component Analysis (PCA) to project the activations into 2D for visualization, leveraging its efficiency and ability to preserve global structure.



(a) Base model [32]. (b) Fine-tuned model [45].

Fig. 4: PCA projection of safety neuron activations.

As shown in Figure 4a, benign (green) and malicious (red) prompts form clearly separated clusters, demonstrating that safety neurons are specialized in detecting unsafe content. In contrast, jailbreaking prompts (gray) lie in an intermediate region, blurring the boundary between safe and unsafe activations. This illustrates how jailbreaking attacks can bypass safety alignment: by compressing safety neurons' activations, they evade triggering defense mechanisms while still generating unsafe outputs. When comparing the distributions between the base and fine-tuned models (Figure 4b), the activation patterns remain nearly identical, supporting the transferability of safety neurons across models within the same LLM family. Additionally, only 0.5% of the layer's neurons are monitored in this case study, confirming the sparsity of the safety mechanism. Further experiments on larger LLMs with 32 billion parameters are presented in Appendix A-A, where we observe consistent behavior. NeuroStrike exploits these properties to conduct attacks in both white-box and black-box settings, which are detailed in the next two sections.

VII. Attack on Open-weight LLMs

We evaluate our attack on 24 open-source LLMs with diverse architectures and sizes, including models from Meta [32], [47], Alibaba [33], [34], [48]–[50], Microsoft [51], [52], Google [29], [53], and DeepSeek [54], as well as 11 of their fine-tuned variants [2], [45], [48], [55]–[62]. All considered LLMs include built-in general-purpose safety alignment or are fine-tuned from base models that were aligned before release, typically via supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF). These safety mechanisms aim to broadly reduce harmful or sensitive outputs and are not designed for specific domains such as cybersecurity or biosecurity.

As described in Section V-A, we begin by identifying safety neurons using a balanced dataset of over 7 000 malicious [63]–[65] and 7 000 benign prompts [46]. We launch attacks using four additional benchmark datasets [66]–[69] to assess the generalizability of identified neurons. Due to the page limit, we present the results on the StrongREJECT [66] dataset below. Additional experiments are presented in Appendix A-B.

A. Attack Performance with Safety Neuron Pruning

Table I presents the Attack Success Rate (ASR) across a diverse set of LLMs, including the last three models specifically optimized for enhanced reasoning capabilities. These reasoning-augmented models are designed to better decompose instructions, infer intermediate steps, and validate outputs, capabilities that could, in theory, strengthen resistance to unsafe or adversarial inputs. The table reports ASR under different pruning levels of safety neurons (0%, 25%, 50%, and 100%), with neurons removed progressively from shallower to deeper layers. The final column indicates the sparsity ratio: the percentage of total MLP neurons identified and pruned as safety neurons.

On average, pruning just 0.4% of neurons results in a dramatic ASR increase from 12.1% (no pruning) to 76.9% (100% pruning), highlighting that safety alignment relies on a surprisingly small set of critical neurons. Even at 50% pruning, safety degradation is substantial, with ASR averaging 45.8%, indicating that partial disruption of the safety neuron set is sufficient to compromise model behavior. Note that different models exhibit varying levels of robustness to the attacks. We hypothesize that this discrepancy arises from redundancy in safety neurons distributed across layers, meaning that NeuroStrike may disable most, not all, safety-related neurons. Interestingly, models optimized for reasoning,

Target Model	0%	25%	50%	100%	Ratio
Llama-3.2-1B-Instruct	2.9%	3.5%	15.7%	74.4%	0.5%
Llama-3.2-3B-Instruct	1.6%	4.2%	46.3%	72.2%	0.4%
Qwen2.5-7B-Instruct	5.1%	4.5%	28.1%	79.6%	0.3%
Qwen2.5-14B-Instruct	1.9%	2.6%	35.8%	85.9%	0.4%
Phi-4-mini-instruct	1.3%	1.3%	67.7%	81.8%	0.5%
Phi-4	0.6%	1.0%	78.3%	89.1%	0.4%
gemma-2b-it	1.0%	1.3%	10.5%	41.2%	0.5%
gemma-7b-it	0.6%	1.3%	24.0%	68.1%	0.5%
DeepSeek-R1-Dist.-Qwen-1.5B	76.7%	78.6%	83.7%	81.5%	0.3%
DeepSeek-R1-Dist.-Llama-8B	39.3%	73.8%	81.2%	86.9%	0.4%
QwQ-32B	2.9%	3.2%	32.3%	85.3%	0.5%
Average	*12.1%*	*15.9%*	*45.8%*	*76.9%*	*0.4%*

TABLE I: ASR and Safety Neuron Ratio on different LLMs.

such as DeepSeek variants and QwQ-32B, show no greater resistance to neuron-level attacks. This observation confirms that, despite improved decomposition and inference abilities, they still rely on sparse, centralized safety neurons and remain equally vulnerable when these are disrupted; the enhanced reasoning capability does not inherently improve safety robustness when alignment relies on localized neuron activations. These findings echo the *Lottery Ticket Hypothesis* (LTH) [70], which suggests that small, specialized subnetworks within a large model can disproportionately drive performance. In our context, alignment training appears to produce a sparse "winning ticket" for safety: an easily identifiable subnetwork that governs rejection behavior. However, unlike in the original LTH, where subnetworks are valuable for generalization, the safety neuron subnetwork represents a single point of failure. Once disrupted, the model's safety alignment collapses.

We further assess the generality of safety neurons in state-of-the-art multimodal LLMs: Gemma-3 [29] and Qwen2.5-VL [50], which can process both image and text inputs. Safety neurons are first identified using only *text* inputs, identical to previous experiments. During the attack, however, we evaluate their effect when the model is queried with images. We consider two types of inputs: (1) text-to-image (T2I) conversions of malicious prompts from the StrongREJECT dataset and (2) Not Safe For Work (NSFW) images [71]. The former tests the cross-modal generalization of safety neurons; the latter examines their ability to detect image-specific unsafe content.

Target Model	T2I w/ SN	NSFW w/ SN	T2I w/o SN	NSFW w/o SN	Ratio
gemma-3-12b-it	0.6%	19.4%	82.1%	100%	0.6%
gemma-3-27b-it	0.3%	12.8%	73.2%	100%	0.6%
Qwen2.5-VL-7B-Instruct	0.9%	99.8%	78.6%	0.5%	0.5%
Qwen2.5-VL-32B-Instruct	0.6%	97.8%	88.8%	0.5%	0.5%
Average	*0.6%*	*57.5%*	*80.7%*	*100%*	*0.6%*

TABLE II: ASR and Safety Neuron (SN) Ratio with text-to-image (T2I) and NSFW images on multimodal LLMs.

Table II shows that pruning safety neurons (SN), identified solely using text inputs, leads to a substantial increase in ASR with malicious image inputs. For example, in Gemma-3-12B-it, ASR rises from 0.6% to 82.1% on T2I inputs and from 19.4% to 100% on NSFW images. Similar trends hold for all evaluated models. Importantly, these attacks require modifying less than 0.6% of the layer's neurons, yet they completely dismantle the safety alignment, even when inputs are images.

B. Transfer Safety Neurons Within the LLM Family

LLMs are often adapted for specific domains or capabilities through two primary techniques: *supervised fine-tuning* and *distillation*. The former technique involves continuing gradient-based training of a base model on domain-specific data, typically with supervised labels or structured prompts. Distillation transfers knowledge from a large "teacher" model to a smaller "student" model by training the latter to mimic the outputs of the former. Here, we evaluate the transferability of safety neurons under both adaptation strategies.

First, we examine 11 fine-tuned models derived from various base LLMs, each tailored to a different domain ranging from biomedicine and financial reasoning to non-English languages, roleplay, and code generation. Table III summarizes the ASR before and after pruning safety neurons transferred from the base model, along with the sparsity ratio of the pruned neurons. The ASR difference with the base model is highlighted in red/green. When comparing with the ASR of the base model, we observe an ASR increase of 23% with the fine-tuned model, which confirms the conclusion from [6] that the safety alignment can be compromised by benign fine-tuning. Safety neurons identified from the base model remain effective across fine-tuned variants. On average, ASR increases from 25.1% to 78.5% after pruning, more than a 3× increase in ASR. Some models, such as Vikhr-Llama-3.2-1B-Instruct and gemma-2-2b-jpn-it, initially exhibit near-zero vulnerability but become fully compromised after pruning, with ASR jumping to 74.4% and 63.9%, respectively. Besides, ASR of the base and fine-tuned models is similar after pruning (1.8% of increase), validating the transferability of the safety neurons within the same LLM family. Notably, the number of pruned neurons remains small (0.5% on average), confirming that fine-tuning rarely modifies the safety-critical subnetworks.

Next, we assess neuron transferability across distilled LLMs using five DeepSeek models distilled from Qwen and LLaMA variants. As shown in Table IV, our results reveal a similar trend in the distillation setting. Although distilled models already exhibit elevated ASR compared to their base counterparts (e.g., 76.7% vs. 8.6% for Qwen2.5-Math-1.5B), pruning safety neurons raises this further to 83.1% in the same model. On average, ASR jumps from 3.7% in the base models to 77.7% in the distilled variants after safety neuron pruning. These findings suggest that the distillation process not only preserves safety neuron behavior but may further weaken safety boundaries, amplifying the impact of neuron-based attacks. Interestingly, the distilled model performs significantly worse than the base model even before applying NeuroStrike. Indeed, distillation degrades safety alignment by compressing model behaviors, potentially weakening or partially omitting safety mechanisms during transfer. Despite this, the remaining alignment still relies on a sparse set of neurons, preserving transferability and allowing NeuroStrike to amplify the attack

Base Model	Target (Fine-tuned) Model	Fine-tuned for	ASR w/ SN	ASR w/o SN	Ratio
Llama-3.1-8B-Instruct	Llama-3.1-8B-UltraMedical	Biomedicine	38.0%$_{+37.0\%}$	83.4%$_{-3.5\%}$	0.7%
Llama-3.2-1B-Instruct	Vikhr-Llama-3.2-1B-Instruct	Russian language	0.3%$_{-2.6\%}$	74.4%$_{+0.0\%}$	0.5%
Llama-3.2-3B-Instruct	Llama-Doctor-3.2-3B-Instruct	Medical consultation	22.4%$_{+20.8\%}$	76.0%$_{+3.8\%}$	0.4%
Qwen2.5-7B-Instruct	Qwen2.5-Coder-7B-Instruct	Programming	2.6%$_{-2.5\%}$	78.0%$_{-1.6\%}$	0.3%
Qwen2.5-7B-Instruct	Fin-R1	Financial reasoning	20.1%$_{+15.0\%}$	86.9%$_{+7.3\%}$	0.3%
Qwen2.5-14B-Instruct	oxy-1-small	Role play	78.9%$_{+77.0\%}$	88.1%$_{+2.2\%}$	0.4%
Qwen2.5-32B-Instruct	s1.1-32B	Reasoning	47.2%$_{+44.6\%}$	87.5%$_{+0.9\%}$	0.6%
Phi-4-mini-instruct	phi-4-mini-chinese-it-e1	Reasoning & STEM	4.8%$_{+3.5\%}$	90.1%$_{+8.3\%}$	0.5%
Phi-4	DNA-R1	Korean language	61.3%$_{+60.7\%}$	91.6%$_{+2.5\%}$	0.4%
gemma-2-2b-it	gemma-2-2b-jpn-it	Japanese language	0.0%$_{+0.0\%}$	63.9%$_{-2.2\%}$	0.6%
gemma-2-9b-it	Quill-v1	Humanlike writing	0.0%$_{+0.0\%}$	43.8%$_{+2.3\%}$	0.6%
Average			*25.1%*$_{+23.0\%}$	*78.5%*$_{+1.8\%}$	*0.5%*

TABLE III: Safety Neurons (SN) Transfer Attack on Fine-tuned LLMs. The difference with the base model is in red/green.

further. Together, these results demonstrate that safety neurons form a generalizable, attackable core across model variants, regardless of whether they are fine-tuned or distilled. Our neuron transfer attacks remain highly effective with minimal modifications, providing a practical and reliable threat vector across the LLM families.

C. Utility Impact: Original vs. Pruned Models

While pruning safety neurons significantly increases ASR, it is essential to ensure that this intervention does not degrade the model's general-purpose capabilities. In this section, we compare the performance of the original and pruned models on language understanding and reasoning benchmarks: HellaSwag [40], Recognizing Textual Entailment (RTE) [41], WinoGrande [42], ARC Challenge [43], OpenBookQA [44], and Corpus of Linguistic Acceptability (CoLA) [41].

Figure 5 shows the comparative performance of original and pruned models across these benchmarks. We use standard accuracy metrics to assess each model's utility on these tasks. Overall, we observe that while pruning introduces moderate utility degradation on some reasoning-heavy tasks, most models largely maintain performance on core benchmarks. For instance, in the ARC Challenge, the average accuracy across models dropped from 45.2% (original) to 39.9% (pruned), and in OpenBookQA, it remained stable, changing slightly from 40.9% to 41.2%. In contrast, benchmarks like CoLA and RTE saw modest changes: CoLA averaged 65.6% (original) versus 63.2% (pruned), and RTE dropped from 69.1% to 64.5%. Similarly, HellaSwag showed a decrease from 53.4% to 47.0%, and WinoGrande from 62.9% to 58.8%. This indicates that safety neurons' removal primarily affects safety alignment mechanisms without significantly impairing general language understanding or reasoning capabilities. Appendix A-C shows the influence on model utility with different z-score thresholds.

VIII. Attack On Black-Box and Proprietary LLMs

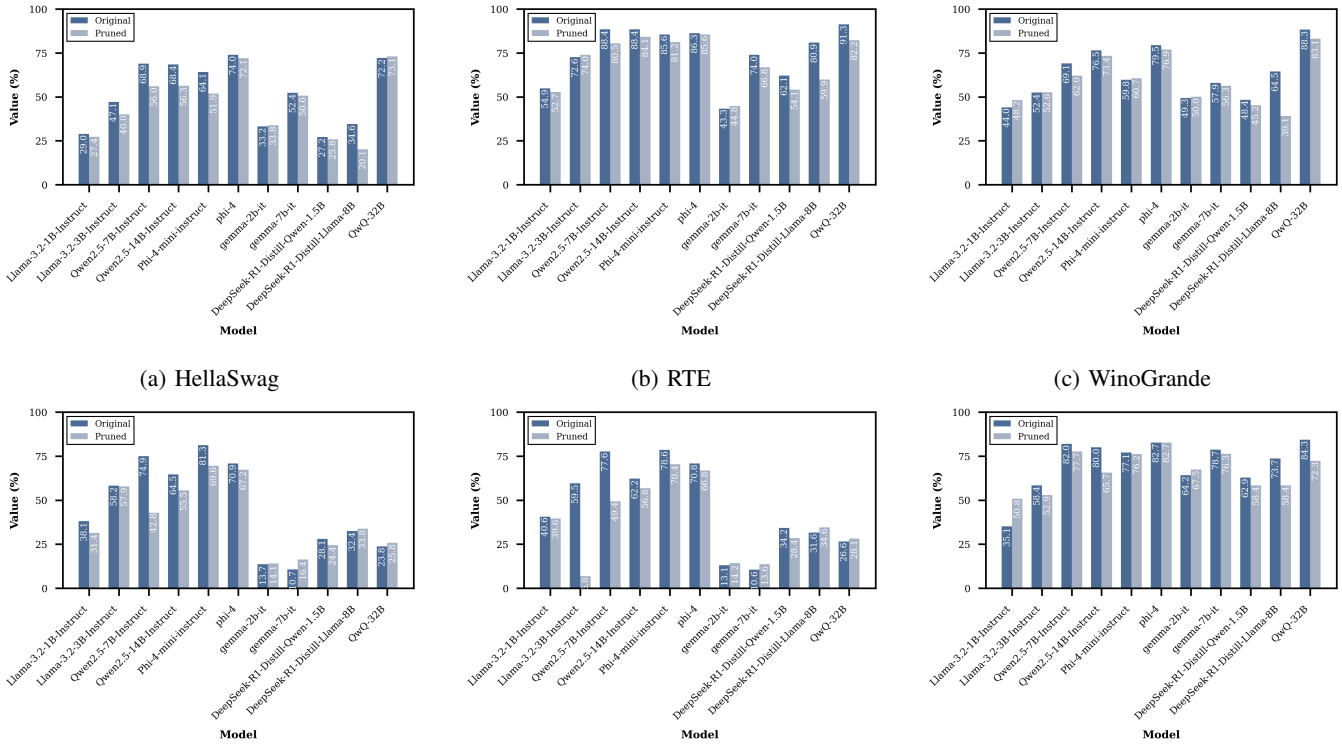
To assess the transferability of safety neuron-guided attacks to black-box LLMs, we perform profiling attacks on Google's Gemini models, Gemini-2.0-Flash, Gemini-2.0-Flash-Lite, and Gemini-1.5-Pro, using Gemma-3 as the open-weight surrogate due to their shared architecture and training approach [29]. To reflect scenarios where open-weight models are deployed in proprietary systems, we also evaluate Gemma-3-1B-it and QwQ-32B as black-box targets, using Gemma-3-1B-it and Qwen2.5-32B-Instruct as their respective surrogates. The former simulates attacks on the same model, while the latter targets a model from the same family. All evaluations are conducted via input-output interfaces to ensure consistency. Our attack pipeline follows Section IV-D; After training, we generate 2 000 prompts from the trained generator, validate on the surrogate model, and evaluate them on the target models. We first compare the ASR of our GRPO-generated prompts against two baselines from the JailBreakV-28K dataset [35]: (i) vanilla malicious prompts and (ii) manually crafted jailbreak prompts. All evaluations are performed in a black-box manner using API access, and responses are classified as safe or unsafe using the Llama-Guard-3-8B judge model.

Table V presents results across five target models and three prompt types. Our LLM profiling attack consistently outperforms both baselines, achieving an average ASR of 63.7%, 60.2% higher than vanilla malicious prompts and 50.2% higher than manually crafted jailbreak prompts. Prompts generated using the Gemma-3 surrogate transfer well to proprietary Gemini models, with ASRs of 54.7%, 49.2%, and 55.7% on Gemini-2.0-Flash, Flash-Lite, and 1.5-Pro, respectively. The approach also exhibits strong within-family transfer, reaching 79.9% ASR on Gemma-3-1B-it and 78.9% on QwQ-32B. These results highlight the effectiveness and generalizability of neuron-guided prompt generation in black-box scenarios.

Next, we benchmark NeuroStrike in a black-box setting against recent prompt-to-prompt jailbreak methods: PAIR [72], which iteratively refines prompts via APE [73]; TAP [74], which explores prompts through branching and pruning; and Puzzler [75], which crafts indirect, game-like prompts to bypass filters. As shown in Table VI, PAIR and TAP show average ASRs of 31.0% and 17.3%, respectively, reflecting the limitations of direct prompt engineering against modern safety-aligned LLMs. Puzzler achieves a substantially higher average ASR of 85.7%, leveraging adaptive online interactions to iteratively steer the model toward unsafe completions. In contrast, NeuroStrike adopts an offline neuron-level suppression approach and still achieves a strong average ASR of 63.7%, outperforming PAIR and TAP across all models.

Base Model	Target (Distilled) Model	ASR Before Distillation	ASR After Distillation	ASR w/o SN	Ratio
Qwen2.5-Math-1.5B-Instruct	DeepSeek-R1-Distill-Qwen-1.5B	8.6%	76.7%$_{+68.1\%}$	83.1%$_{+27.8\%}$	0.4%
Qwen2.5-Math-7B-Instruct	DeepSeek-R1-Distill-Qwen-7B	4.5%	40.3%$_{+35.8\%}$	85.0%$_{+1.3\%}$	0.5%
Llama-3.1-8B-Instruct	DeepSeek-R1-Distill-Llama-8B	1.0%	39.3%$_{+38.3\%}$	86.9%$_{+0.0\%}$	0.7%
Qwen2.5-14B-Instruct	DeepSeek-R1-Distill-Qwen-14B	1.9%	25.2%$_{+23.3\%}$	86.3%$_{-4.0\%}$	0.4%
Qwen2.5-32B-Instruct	DeepSeek-R1-Distill-Qwen-32B	2.6%	26.2%$_{+23.6\%}$	82.1%$_{-4.5\%}$	0.6%
Average		*3.7%*	*41.5%*$_{+37.8\%}$	*77.7%*$_{+4.1\%}$	*0.5%*

TABLE IV: Safety Neurons (SN) Transfer Attacks on Distilled LLMs. The difference with the base model is in red/green.



(a) HellaSwag (b) RTE (c) WinoGrande

(d) ARC Challenge (e) OpenBookQA (f) CoLA

Fig. 5: Utility evaluation of original vs. pruned models across six NLU benchmarks.

Target Model	Vanilla	Jailbreak	NeuroStrike
Gemini-2.0-Flash	0.8%	15.7%	54.7%
Gemini-2.0-Flash-Lite	1.0%	15.6%	49.2%
Gemini-1.5-Pro	1.4%	5.3%	55.7%
Gemma-3-1b-it	10.6%	24.6%	79.9%
QwQ-32B	3.6%	6.3%	78.9%
Average	*3.5%*	*13.5%*	*63.7%*

TABLE V: ASRs benchmark with different prompt types.

Model	PAIR	TAP	Puzzler	NeuroStrike
Gemini-2.0-Flash	37.3%	14.0%	73.0%	54.7%
Gemini-2.0-Flash-Lite	9.8%	8.0%	86.1%	49.2%
Gemini-1.5-Pro	54.9%	32.0%	75.0%	55.7%
Gemma-3-1b-it	33.9%	20.4%	94.0%	79.9%
QwQ-32B	18.9%	12.2%	97.2%	78.9%
Average	*31.0%*	*17.3%*	*85.7%*	*63.7%*

TABLE VI: ASR benchmark with state-of-the-art jailbreaks.

IX. Defense Analysis

NeuroStrike demonstrates broad effectiveness across diverse models, architectures, modalities, and fine-tuning strategies. To further assess its robustness, we evaluate its ability to bypass three hardened safety-alignment defenses: Perplexity Filtering [76], which flags prompts with low linguistic naturalness; SmoothLLM [77], which perturbs prompts and aggregates outputs to reduce attack success; and Layer-Specific Editing (LSE) [78], which realigns internal model layers to reinforce safety behavior. As LSE requires white-box access, it is only applied to open models, excluding the Gemini family.

As shown in Table VII, NeuroStrike consistently bypasses all three defenses. Against Perplexity Filtering, it achieves an average ASR of 60.0%, indicating that neuron-level perturbations preserve linguistic plausibility. SmoothLLM is similarly ineffective, with NeuroStrike maintaining a 61.7% average ASR, demonstrating robustness to prompt perturbations and output aggregation. Under the more stringent LSE, Neu-

roStrike still achieves 60.0% ASR on Gemma-3-1b-it and 43.4% on QwQ-32B in a black-box setting.

Model	Perplexity Filter	SmoothLLM	LSE
Gemini-2.0-Flash	48.7%	52.8%	–
Gemini-2.0-Flash-Lite	43.2%	47.3%	–
Gemini-1.5-Pro	49.7%	53.8%	–
Gemma-3-1b-it	79.8%	78.0%	60.0%
QwQ-32B	78.8%	76.4%	43.4%
Average	*60.0%*	*61.7%*	*54.4%*

TABLE VII: ASR of NeuroStrike under various defenses.

To further assess LSE, we apply NeuroStrike in the white-box setting. The results show that, with safety neuron pruning, the ASR boosts significantly from 16.0% to 86.6% on Gemma-3-1b-it and from 4.8% to 84.7% on QwQ-32B, showing that NeuroStrike can reliably circumvent even internal safety mechanisms when granted full model access.

X. Ablation and Hyperparameter Study

A. The Selection Threshold of Safety Neurons

To investigate how the threshold of the z-score affects the selection of safety neurons and subsequently impacts attack performance, we perform an ablation study using three representative thresholds: $z = 2$, 3, and 4.

Target Model	$z = 2$	$z = 3$	$z = 4$
Llama-3.2-1B-Instruct	85.0%	74.4%	79.2%
Llama-3.2-3B-Instruct	76.0%	72.2%	58.8%
Qwen2.5-7B-Instruct	85.9%	79.6%	71.6%
Qwen2.5-14B-Instruct	84.7%	85.9%	81.8%
Phi-4-mini-instruct	89.8%	81.8%	75.1%
Phi-4	88.2%	89.1%	80.5%
gemma-2b-it	65.2%	41.2%	20.1%
gemma-7b-it	79.9%	68.1%	37.6%
DeepSeek-R1-Distill-Qwen-1.5B	78.9%	83.7%	81.8%
DeepSeek-R1-Distill-Llama-8B	N/A	81.2%	48.6%
QwQ-32B	84.6%	85.3%	62.0%
Average	*74.4%*	*76.9%*	*63.4%*

TABLE VIII: ASR with different z-score Threshold.

As shown in Table VIII, a lower threshold ($z = 2$), 5.4% of neurons pruned on average, leads to a higher ASR (84.4% on average) but may introduce noise by including irrelevant neurons, influencing the general performance of the model. For instance, the DeepSeek-R1-Distill-Llama-8B failed to give proper responses after the safety neurons' removal (marked with N/A in the table). Conversely, a higher threshold ($z = 4$) results in a smaller set of highly confident safety neurons (0.4% of total neurons on average), but at the cost of lower ASR (63.4% on average), likely due to under-selecting impactful neurons. A moderate threshold with $z = 3$ yields a strong balance, achieving 76.9% average ASR with only 1.4% of neurons pruned (as shown in Table I). This justifies our default choice in the main experiments: it achieves high attack effectiveness with minimal impact on model structure and performance. Appendix A-C presents the quantitative analysis on the influence of different z-score thresholds on models' utility. A higher percentage of safety neuron pruning leads to reduced model utility.

B. Target Pruning Blocks

As discussed in Section II-A, MLP typically comprises two key projection layers: the gate projection and the up projection.[3] To identify which of these layers predominantly hosts critical safety neurons, we conduct an ablation study by selectively pruning neurons in the gate, up, or both layers simultaneously. We exclude the Phi-4 model family from this analysis, as these models merge the gate and up layers into a single projection for computational efficiency. Table IX shows the ASR achieved under each pruning strategy.

Target Model	Gate	Up	Gate & Up
Llama-3.2-1B-Instruct	74.1%	6.4%	74.4%
Llama-3.2-3B-Instruct	55.9%	32.3%	72.2%
Qwen2.5-7B-Instruct	75.1%	23.0%	79.6%
Qwen2.5-14B-Instruct	81.2%	41.2%	85.9%
gemma-2b-it	31.6%	1.6%	41.2%
gemma-7b-it	57.8%	4.2%	68.1%
DeepSeek-R1-Distill-Qwen-1.5B	78.3%	87.2%	81.5%
DeepSeek-R1-Distill-Llama-8B	83.1%	68.4%	86.9%
QwQ-32B	67.4%	39.0%	85.3%
Average	*67.2%*	*33.7%*	*75.0%*

TABLE IX: ASR with different pruning strategy.

The results show that pruning neurons from the gate layer alone achieves significantly higher ASR (67.2% on average) than pruning from the up projection layer (33.7%), suggesting that the gate layer plays a more dominant role in safety alignment. When safety neurons from both sublayers are pruned together, performance improves further, reaching an average ASR of 75.0%. Interestingly, in models such as LLaMA-3.2-1B and gemma-2b-it, pruning the up layer alone yields minimal effect, while pruning the gate layer leads to strong ASR, comparable to pruning both layers. However, for DeepSeek-R1-Distill-Qwen-1.5B, pruning the up layer outperforms gate-only pruning (87.2% vs. 78.3%), indicating that the safety signal distribution can vary across architectures. This ablation indicates that safety neuron selection should primarily target gate layers for maximum efficiency. However, incorporating neurons from both layers can achieve general attack success.

C. GRPO Reward Function

We conduct an ablation study on the GRPO reward components to understand their impacts on jailbreak success. Specifically, we evaluate three reward configurations: 1) baseline (no reward), 2) R_{jb} only, and 3) R_{GRPO} (R_{jb} & R_{neuron}). In the case of baseline, we generate jailbreaking prompts with the SFT-trained model. To benchmark models' performance with different reward settings, we calculate the ASR of jailbreaking prompts on $f_{\theta_{src}}$ (Gemma-3-1B-it). The results show that the complete R_{GRPO} reward significantly outperforms both ablated configurations, achieving an average ASR of 73.2%, compared to only 65.3% without R_{neuron} and 53.6% when relying solely on the SFT baseline. Notably, omitting the GRPO fine-tuning significantly reduces the ASR, highlighting that the

[3]Down projection is a compressive, output-mapping layer; it is usually not where specialized behavior (like safety enforcement) emerges [79].

LLM profiling is critical for jailbreak effectiveness. Similarly, incorporating the safety neuron reward further improves ASR by suppressing safety neuron activations, enhancing the generator's evasion capabilities.

XI. Discussion

Surrogate Model Dependency. Our black-box attack (Section IV-D) assumes access to a white-box surrogate of the target black-box model. While surrogate models may not always be available, in practice, many production LLMs are known to be built on or fine-tuned from open-weight models (e.g., Mistral variants in Claude, Gemma in Gemini, LLaMA in Meta AI). In such cases, attackers can use public surrogates from the same developer or architecture family. On the other hand, even if the surrogate model is not available, one can still reuse the fine-tuned generator from other models for the attacks. To illustrate this, we test the generator fine-tuned on Gemma against xAI's Grok-3-beta, a closed model with no known surrogate, featuring distinct architectural and alignment strategies. To our knowledge, this is the first attack on the Grok. Despite these differences, our method achieved a 43.8% ASR on Grok-3-beta, significantly outperforming both baselines (2.5% for both vanilla and manually crafted jailbreak prompts). This result underscores the reliability of the generator and the broad applicability of the LLM profiling attack across diverse black-box LLMs.

Potential Defenses. Although existing defenses cannot block NeuroStrike (see Section IX), the sparse and universal nature of the safety neurons suggests clear targets for potential defenses: proactively distributing these critical neuron subsets into more layers/neurons. For instance, adopting a multi-objective alignment strategy [80], where multiple independent safety objectives guide neuron activations, could help diversify and diffuse neuron-level responsibilities. Such multi-objective alignment would create less concentrated neuron activation patterns, reducing susceptibility to targeted neuron-level attacks. Architecturally, the Mixture-of-Expert model, which separates a unified feedforward network into multiple experts, could potentially increase the difficulties in conducting the NeuroStrike attacks. To prevent the misuse of compromised models, system-level defenses could be effective. These include monitoring internal activations for abnormal neuron suppression, verifying model integrity through fingerprinting or attestation, and implementing runtime randomization of neuron masking.

XII. Related Work

Template-based Jailbreak Attacks. Early jailbreak methods used carefully engineered prompts, such as role-play, hidden directives, obfuscation, and prompt decomposition, to bypass LLM safety measures [8], [81]–[89]. As models improved, these static methods became increasingly ineffective, prompting the development of dynamic jailbreak attacks. Automatic methods emerged, using techniques such as mutation-based fuzzing [90], [91], gradient-based optimization [92]–[98], and genetic algorithms [99], [100] to adaptively generate robust jailbreak prompts. Unfortunately, they remain input-centric and do not exploit the model's internal safety mechanisms, limiting their generalization across different LLMs.

LLM-based Prompt-to-Prompt Jailbreak. Fixed jailbreaking template-based attacks are inherently limited, as different prompts may require tailored adjustments. To address this, recent jailbreak methods utilize generative models, often other LLMs, to produce adaptive prompt variations [93], [101], [102]. Approaches such as APE [73], PAIR [72], TAP [74], and Puzzler [75] dynamically refine adversarial prompts based on iterative interactions with the target model. However, these methods operate purely in the input space and rely heavily on feedback from the target model.

Neuron Interpretability. Interpreting the functional role of individual neurons has been an active research direction. Recent efforts in neuron interpretability have taken two main approaches: analyzing neuron activations triggered by specific concepts [103]–[107], and using probing methods such as training classifiers on activations to decode linguistic properties [108]. To the best of our knowledge, our work is the first to explicitly identify and interpret neurons responsible for safety alignment in large-scale transformer-based LLMs, revealing their critical role in safety alignment.

XIII. Conclusion

This paper reveals a fundamental vulnerability in safety-aligned LLMs: the emergence of sparse, specialized safety neurons that enforce safety constraints. We introduce NeuroStrike, a lightweight attack framework that identifies and suppresses these neurons using simple linear probes, effectively disabling safety across a wide range of architectures and input modalities. Evaluated on over 30 open-weight and proprietary models, NeuroStrike achieves high attack success rates in both white- and black-box settings. The transferability of safety neurons across model variants further underscores the fragility of current alignment strategies. These findings highlight the urgent need for alignment methods that prevent safety from being localized in easily exploitable components.

Acknowledgement

Our research work was partially funded by DFG-SFB 1119-236615297, the European Union under Horizon Europe Programme-Grant Agreement 101070537-CrossCon and-Grant Agreement 101093126-ACES, NSF-DFG-Grant 538883423, the European Research Council under the ERC Programme-Grant 101055025-HYDRANOS, as well as the Federal Ministry of Education and Research of Germany (BMBF) within the IoTGuard project. Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect those of the European Union, the European Research Council, or the Federal Ministry of Education and Research of Germany.

Ethics Consideration

Our work investigates vulnerabilities inherent in the safety alignment mechanisms of large language models (LLMs), highlighting how neuron-level attacks can effectively bypass model safeguards. While we intend to raise awareness of critical weaknesses to inform and enhance future safety measures, we acknowledge that disclosing such vulnerabilities could potentially be exploited for malicious purposes. To mitigate these risks, we have taken several responsible steps:

- Engagement with Model Providers: We have proactively notified organizations whose models were directly impacted by our findings, providing sufficient details to facilitate vulnerability verification without publicizing explicit exploit details.
- Responsible Research Practices: All experiments conducted in this research were carefully designed to avoid exposing sensitive user data or causing real-world harm. Evaluations were performed in controlled environments, strictly using publicly available or simulated data. We will only release jailbreaking prompts and safety neuron indices under responsible disclosure protocols.
- Broader Impact and Recommendations: Our findings are explicitly framed to guide the community toward more robust defenses and safer deployment strategies. We strongly advocate for improving neuron-level interpretability and safety mechanisms in LLMs, promoting greater resilience against adversarial exploitation.

Despite these precautions, we acknowledge that revealing this class of vulnerability inherently carries some risk. However, we firmly believe that transparent disclosure of such vulnerabilities, combined with responsible communication and collaboration with industry stakeholders, provides a net benefit by encouraging more secure, robust, and ethically aligned development and deployment of LLM technologies.

References

[1] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting, "Large language models in medicine," *Nature medicine*, vol. 29, no. 8, pp. 1930–1940, 2023.

[2] Z. Liu, X. Guo, F. Lou, L. Zeng, J. Niu, Z. Wang, J. Xu, W. Cai, Z. Yang, X. Zhao *et al.*, "Fin-r1: A large language model for financial reasoning through reinforcement learning," *arXiv preprint arXiv:2503.16252*, 2025.

[3] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy, "Challenges and applications of large language models," *arXiv preprint arXiv:2307.10169*, 2023.

[4] A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How does llm safety training fail?" *Advances in Neural Information Processing Systems*, vol. 36, pp. 80 079–80 110, 2023.

[5] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[6] X. Qi, Y. Zeng, T. Xie, P.-Y. Chen, R. Jia, P. Mittal, and P. Henderson, "Fine-tuning aligned language models compromises safety, even when users do not intend to!" *arXiv preprint arXiv:2310.03693*, 2023.

[7] Z. Niu, H. Ren, X. Gao, G. Hua, and R. Jin, "Jailbreaking attack against multimodal large language model," *arXiv preprint arXiv:2402.02309*, 2024.

[8] X. Shen, Z. Chen, M. Backes, Y. Shen, and Y. Zhang, "" do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 1671–1685.

[9] S. Li, L. Yao, L. Zhang, and Y. Li, "Safety layers in aligned large language models: The key to llm security," *arXiv preprint arXiv:2408.17003*, 2024.

[10] J. Chen, X. Wang, Z. Yao, Y. Bai, L. Hou, and J. Li, "Finding safety neurons in large language models," *arXiv preprint arXiv:2406.14144*, 2024.

[11] W. Zhao, Y. Hu, Z. Li, Y. Deng, Y. Zhao, B. Qin, and T.-S. Chua, "Towards comprehensive and efficient post safety alignment of large language models via safety patching," *arXiv preprint arXiv:2405.13820*, 2024.

[12] X. Xu, Z. Liu, S. Koffas, S. Yu, and S. Picek, "Ban: Detecting backdoors activated by adversarial neuron noise," *arXiv preprint arXiv:2405.19928*, 2024.

[13] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Applied Sciences*, vol. 9, no. 5, p. 909, 2019.

[14] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE transactions on neural networks and learning systems*, vol. 35, no. 1, pp. 5–22, 2022.

[15] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6541–6549.

[16] A. Tamkin, M. Taufeeque, and N. D. Goodman, "Codebook features: Sparse and discrete interpretability for neural networks," *arXiv preprint arXiv:2310.17230*, 2023.

[17] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu *et al.*, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," *arXiv preprint arXiv:2402.03300*, 2024.

[18] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[19] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[20] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan *et al.*, "Deepseek-v3 technical report," *arXiv preprint arXiv:2412.19437*, 2024.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[23] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pre-trained models," *arXiv preprint arXiv:2004.06660*, 2020.

[24] H. Yang, K. Xiang, M. Ge, H. Li, R. Lu, and S. Yu, "A comprehensive overview of backdoor attacks in large language models within communication networks," *IEEE Network*, 2024.

[25] R. Staab, M. Vero, M. Balunović, and M. Vechev, "Beyond memorization: Violating privacy via inference with large language models," *arXiv preprint arXiv:2310.07298*, 2023.

[26] Y. Liu, G. Deng, Y. Li, K. Wang, Z. Wang, X. Wang, T. Zhang, Y. Liu, H. Wang, Y. Zheng *et al.*, "Prompt injection attack against llm-integrated applications," *arXiv preprint arXiv:2306.05499*, 2023.

[27] G. Deng, Y. Liu, Y. Li, K. Wang, Y. Zhang, Z. Li, H. Wang, T. Zhang, and Y. Liu, "Masterkey: Automated jailbreaking of large language model chatbots," in *NDSS*, 2024. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/masterkey-automated-jailbreaking-of-large-language-model-chatbots/

[28] Z. Yu, X. Liu, S. Liang, Z. Cameron, C. Xiao, and N. Zhang, "Don't listen to me: Understanding and exploring jailbreak prompts of large language models," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 4675–4692. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/yu-zhiyuan

[29] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière *et al.*, "Gemma 3 technical report," *arXiv preprint arXiv:2503.19786*, 2025.

[30] M. Geva, A. Caciularu, K. R. Wang, and Y. Goldberg, "Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space," *arXiv preprint arXiv:2203.14680*, 2022.

[31] H. J. Davies, "Decoding specialised feature neurons in llms with the final projection layer," *arXiv preprint arXiv:2501.02688*, 2025.

[32] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[33] Q. Team, "Qwen2.5: A party of foundation models," September 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen2.5/

[34] ——, "Qwq-32b: Embracing the power of reinforcement learning," March 2025. [Online]. Available: https://qwenlm.github.io/blog/qwq-32b/

[35] W. Luo, S. Ma, X. Liu, X. Guo, and C. Xiao, "Jailbreakv-28k: A benchmark for assessing the robustness of multimodal large language models against jailbreak attacks," 2024.

[36] L. Jiang, K. Rao, S. Han, A. Ettinger, F. Brahman, S. Kumar, N. Mireshghallah, X. Lu, M. Sap, Y. Choi, and N. Dziri, "Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models," 2024. [Online]. Available: https://arxiv.org/abs/2406.18510

[37] D. Kalajdzievski, "A rank stabilization scaling factor for fine-tuning with lora," *arXiv preprint arXiv:2312.03732*, 2023.

[38] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[39] H. Inan, K. Upasani, J. Chi, R. Rungta, K. Iyer, Y. Mao, M. Tontchev, Q. Hu, B. Fuller, D. Testuggine *et al.*, "Llama guard: Llm-based input-output safeguard for human-ai conversations," *arXiv preprint arXiv:2312.06674*, 2023.

[40] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?" *arXiv preprint arXiv:1905.07830*, 2019.

[41] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[42] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale," *Communications of the ACM*, vol. 64, no. 9, pp. 99–106, 2021.

[43] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," *arXiv preprint arXiv:1803.05457*, 2018.

[44] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? a new dataset for open book question answering," *arXiv preprint arXiv:1809.02789*, 2018.

[45] A. Nikolich, K. Korolev, S. Bratchikov, N. Kompanets, and A. Shelmanov, "Vikhr: The family of open-source instruction-tuned large language models for russian," *arXiv preprint arXiv:2405.13929*, 2024. [Online]. Available: https://arxiv.org/pdf/2405.13929

[46] W. Yuan, J. Yu, S. Jiang, K. Padthe, Y. Li, D. Wang, I. Kulikov, K. Cho, Y. Tian, J. E. Weston, and X. Li, "Naturalreasoning: Reasoning in the wild with 2.8m challenging questions," 2025. [Online]. Available: https://arxiv.org/abs/2502.13124

[47] Meta, "Llama 3.2: Revolutionizing edge ai and vision with open, customizable models," September 2024. [Online]. Available: https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/

[48] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Dang *et al.*, "Qwen2. 5-coder technical report," *arXiv preprint arXiv:2409.12186*, 2024.

[49] A. Yang, B. Zhang, B. Hui, B. Gao, B. Yu, C. Li, D. Liu, J. Tu, J. Zhou, J. Lin *et al.*, "Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement," *arXiv preprint arXiv:2409.12122*, 2024.

[50] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, H. Zhong, Y. Zhu, M. Yang, Z. Li, J. Wan, P. Wang, W. Ding, Z. Fu, Y. Xu, J. Ye, X. Zhang, T. Xie, Z. Cheng, H. Zhang, Z. Yang, H. Xu, and J. Lin, "Qwen2.5-vl technical report," *arXiv preprint arXiv:2502.13923*, 2025.

[51] M. Abdin, J. Aneja, H. Behl, S. Bubeck, R. Eldan, S. Gunasekar, M. Harrison, R. J. Hewett, M. Javaheripi, P. Kauffmann *et al.*, "Phi-4 technical report," *arXiv preprint arXiv:2412.08905*, 2024.

[52] A. Abouelenin, A. Ashfaq, A. Atkinson, H. Awadalla, N. Bach, J. Bao, A. Benhaim, M. Cai, V. Chaudhary, C. Chen *et al.*, "Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras," *arXiv preprint arXiv:2503.01743*, 2025.

[53] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love *et al.*, "Gemma: Open models based on gemini research and technology," *arXiv preprint arXiv:2403.08295*, 2024.

[54] DeepSeek-AI, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[55] K. Zhang, S. Zeng, E. Hua, N. Ding, Z.-R. Chen, Z. Ma, H. Li, G. Cui, B. Qi, X. Zhu *et al.*, "Ultramedical: Building specialized generalists in biomedicine," *Advances in Neural Information Processing Systems*, vol. 37, pp. 26 045–26 081, 2024.

[56] P. Sakthi, "Llama-doctor-3.2-3b-instruct," 2024. [Online]. Available: https://huggingface.co/prithivMLmods/Llama-Doctor-3.2-3B-Instruct

[57] O. (oxyapi), "Oxy 1 small: A fine-tuned qwen2.5-14b-instruct model for role-play," 2024. [Online]. Available: https://huggingface.co/oxyapi/oxy-1-small

[58] N. Muennighoff, Z. Yang, W. Shi, X. L. Li, L. Fei-Fei, H. Hajishirzi, L. Zettlemoyer, P. Liang, E. Candès, and T. Hashimoto, "s1: Simple test-time scaling," 2025. [Online]. Available: https://arxiv.org/abs/2501.19393

[59] J. Yang, "phi-4-mini-chinese-it-e1," 2024. [Online]. Available: https://huggingface.co/zake7749/phi-4-mini-chinese-it-e1

[60] J. Lee, J. Kim, S. Park, and S. Lee, "Dna r1," 2025. [Online]. Available: https://huggingface.co/dnotitia/DNA-R1

[61] Google, "gemma-2-2b-jpn-it," 2024. [Online]. Available: https://huggingface.co/google/gemma-2-2b-jpn-it

[62] S. Paech, "Quill-v1," 2024. [Online]. Available: https://huggingface.co/sam-paech/Quill-v1

[63] LLM-LAT, "harmful-dataset," 2024. [Online]. Available: https://huggingface.co/datasets/LLM-LAT/harmful-dataset

[64] R. Bhardwaj, D. D. Anh, and S. Poria, "Language models are homer simpson! safety re-alignment of fine-tuned language models through task arithmetic," 2024.

[65] R. Bhardwaj and S. Poria, "Red-teaming large language models using chain of utterances for safety-alignment," 2023.

[66] A. Souly, Q. Lu, D. Bowen, T. Trinh, E. Hsieh, S. Pandey, P. Abbeel, J. Svegliato, S. Emmons, O. Watkins, and S. Toyer, "A strongreject for empty jailbreaks," 2024.

[67] M. Mazeika, L. Phan, X. Yin, A. Zou, Z. Wang, N. Mu, E. Sakhaee, N. Li, S. Basart, B. Li *et al.*, "Harmbench: A standardized evaluation framework for automated red teaming and robust refusal," *arXiv preprint arXiv:2402.04249*, 2024.

[68] M. Mazeika, A. Zou, N. Mu, L. Phan, Z. Wang, C. Yu, A. Khoja, F. Jiang, A. O'Gara, E. Sakhaee, Z. Xiang, A. Rajabi, D. Hendrycks, R. Poovendran, B. Li, and D. Forsyth, "Tdc 2023 (llm edition): The trojan detection challenge," in *NeurIPS Competition Track*, 2023.

[69] Y. Huang, S. Gupta, M. Xia, K. Li, and D. Chen, "Catastrophic jailbreak of open-source llms via exploiting generation," *arXiv preprint arXiv:2310.06987*, 2023.

[70] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[71] deepghs, "nsfw-detect," 2023. [Online]. Available: https://huggingface.co/datasets/deepghs/nsfw{_}detect

[72] P. Chao, A. Robey, E. Dobriban, H. Hassani, G. J. Pappas, and E. Wong, "Jailbreaking black box large language models in twenty queries," *arXiv preprint arXiv:2310.08419*, 2023.

[73] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," in *The Eleventh International Conference on Learning Representations*, 2022.

[74] A. Mehrotra, M. Zampetakis, P. Kassianik, B. Nelson, H. Anderson, Y. Singer, and A. Karbasi, "Tree of attacks: Jailbreaking black-box llms automatically," *Advances in Neural Information Processing Systems*, vol. 37, pp. 61 065–61 105, 2024.

[75] Z. Chang, M. Li, Y. Liu, J. Wang, Q. Wang, and Y. Liu, "Play guessing game with llm: Indirect jailbreak attack with implicit clues," *arXiv preprint arXiv:2402.09091*, 2024.

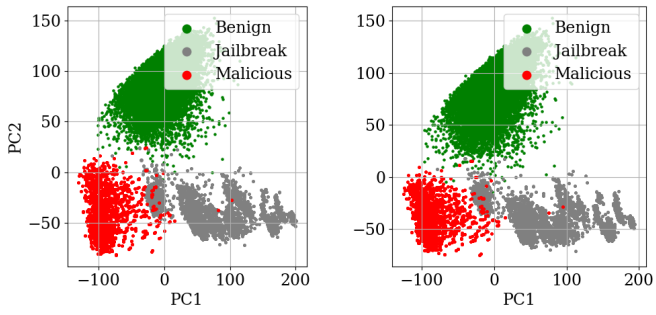
[76] N. Jain, A. Schwarzschild, Y. Wen, G. Somepalli, J. Kirchenbauer, P.-y. Chiang, M. Goldblum, A. Saha, J. Geiping, and T. Goldstein,

"Baseline defenses for adversarial attacks against aligned language models," *arXiv preprint arXiv:2309.00614*, 2023.
[77] A. Robey, E. Wong, H. Hassani, and G. J. Pappas, "Smoothllm: Defending large language models against jailbreaking attacks," *arXiv preprint arXiv:2310.03684*, 2023.
[78] W. Zhao, Z. Li, Y. Li, Y. Zhang, and J. Sun, "Defending large language models against jailbreak attacks via layer-specific editing," *arXiv preprint arXiv:2405.18166*, 2024.
[79] E. Nelson, N. Neel, O. Catherine, H. Tom, J. Nicholas, M. Ben, A. Amanda, B. Yuntao, C. Anna, C. Tom *et al.*, "A mathematical framework for transformer circuits," *Transformer Circuits Thread*, 2021.
[80] B. Liu, X. Li, J. Zhang, J. Wang, T. He, S. Hong, H. Liu, S. Zhang, K. Song, K. Zhu *et al.*, "Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems," *arXiv preprint arXiv:2504.01990*, 2025.
[81] F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," *arXiv preprint arXiv:2211.09527*, 2022.
[82] S. Schulhoff, J. Pinto, A. Khan, L.-F. Bouchard, C. Si, S. Anati, V. Tagliabue, A. L. Kost, C. Carnahan, and J. Boyd-Graber, "Ignore this title and hackaprompt: Exposing systemic vulnerabilities of llms through a global scale prompt hacking competition," *EMNLP 2023 - 2023 Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2023.
[83] S. Jiang, X. Chen, and R. Tang, "Prompt packer: Deceiving llms through compositional instruction with hidden attacks," *arXiv preprint arXiv:2310.10077*, 2023.
[84] B. A. Saiem, M. Shanto, R. Ahsan *et al.*, "Sequentialbreak: Large language models can be fooled by embedding jailbreak prompts into sequential prompt chains," *arXiv preprint arXiv:2411.06426*, 2024.
[85] Z. Wang, Y. Cao, and P. Liu, "Hidden you malicious goal into benign narratives: Jailbreak large language models through logic chain injection," *arXiv preprint arXiv:2404.04849*, 2024.
[86] S. Singh, F. Abri, and A. S. Namin, "Exploiting large language models (llms) through deception techniques and persuasion principles," in *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023, pp. 2508–2517.
[87] H. Li, D. Guo, W. Fan, M. Xu, J. Huang, F. Meng, and Y. Song, "Multi-step jailbreaking privacy attacks on chatgpt," *arXiv preprint arXiv:2304.05197*, 2023.
[88] X. Li, Z. Zhou, J. Zhu, J. Yao, T. Liu, and B. Han, "Deepinception: Hypnotize large language model to be jailbreaker," *arXiv preprint arXiv:2311.03191*, 2023.
[89] H. Jin, R. Chen, A. Zhou, Y. Zhang, and H. Wang, "Guard: Role-playing to generate natural-language jailbreakings to test guideline adherence of large language models," *arXiv preprint arXiv:2402.03299*, 2024.
[90] J. Yu, X. Lin, Z. Yu, and X. Xing, "Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts," *arXiv preprint arXiv:2309.10253*, 2023.
[91] ——, "{LLM-Fuzzer}: Scaling assessment of large language model jailbreaks," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 4657–4674.
[92] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.
[93] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," *arXiv preprint arXiv:2010.15980*, 2020.
[94] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.
[95] L. Qin, S. Welleck, D. Khashabi, and Y. Choi, "Cold decoding: Energy-based constrained text generation with langevin dynamics," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9538–9551, 2022.
[96] Y. Wen, N. Jain, J. Kirchenbauer, M. Goldblum, J. Geiping, and T. Goldstein, "Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery," *Advances in Neural Information Processing Systems*, vol. 36, pp. 51 008–51 025, 2023.
[97] E. Jones, A. Dragan, A. Raghunathan, and J. Steinhardt, "Automatically auditing large language models via discrete optimization," in *International Conference on Machine Learning*. PMLR, 2023, pp. 15 307–15 329.
[98] A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," *arXiv preprint arXiv:2307.15043*, 2023.
[99] F. Wu, X. Liu, and C. Xiao, "Deceptprompt: Exploiting llm-driven code generation via adversarial natural language instructions," *arXiv preprint arXiv:2312.04730*, 2023.
[100] X. Liu, N. Xu, M. Chen, and C. Xiao, "Autodan: Generating stealthy jailbreak prompts on aligned large language models," *arXiv preprint arXiv:2310.04451*, 2023.
[101] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," *arXiv preprint arXiv:2012.15723*, 2020.
[102] R. Pryzant, D. Iter, J. Li, Y. T. Lee, C. Zhu, and M. Zeng, "Automatic prompt optimization with" gradient descent" and beam search," *arXiv preprint arXiv:2305.03495*, 2023.
[103] A. Kádár, G. Chrupała, and A. Alishahi, "Representation of linguistic form and function in recurrent neural networks," *Computational Linguistics*, vol. 43, no. 4, pp. 761–780, 2017.
[104] S. Na, Y. J. Choe, D.-H. Lee, and G. Kim, "Discovery of natural language concepts in individual units of cnns," *arXiv preprint arXiv:1902.07249*, 2019.
[105] J. Mu and J. Andreas, "Compositional explanations of neurons," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 153–17 163, 2020.
[106] X. Suau, L. Zappella, and N. Apostoloff, "Finding experts in transformer models," *arXiv preprint arXiv:2005.07647*, 2020.
[107] O. Antverg and Y. Belinkov, "On the pitfalls of analyzing individual neurons in language models," *arXiv preprint arXiv:2110.07483*, 2021.
[108] Y. Lakretz, G. Kruszewski, T. Desbordes, D. Hupkes, S. Dehaene, and M. Baroni, "The emergence of number and syntax units in lstm language models," *arXiv preprint arXiv:1903.07435*, 2019.

APPENDIX A
ADDITIONAL EXPERIMENTAL RESULTS

A. Visualizing Safety Neuron Activation on 32B LLMs

This section extends the case study to a larger model with 32 billion parameters. Following the setup in Section VI, we select Qwen2.5-32B-Instruct [33] as the base model and s1.1-32B [58] as its fine-tuned counterpart. All other settings remain unchanged.



(a) Base model. (b) Fine-tuned model.

Fig. 6: PCA projection of safety neuron activations.

The results, shown in Figure 6, exhibit patterns consistent with those observed in Figure 4. Benign and malicious prompts form well-separated clusters, confirming the specialization of safety neurons. In contrast, jailbreaking prompts blur the decision boundary, indicating their ability to evade safety filters. The close similarity between the activation distributions of the base and fine-tuned models demonstrates the transferability of safety neurons across large-scale LLMs. Additionally, only 0.5% of neurons are identified as safety neurons, reaffirming

the sparsity of the safety mechanism. These findings further validate the robustness of the safety neuron properties in significantly larger models.

B. Additional Experiments on Open-weight LLMs with Safety Neuron Pruning

In this section, we launch attacks using three additional datasets [67]–[69] to assess the generalizability of identified neurons. As shown in Table X, XI, and XII, the identified safety neurons are generalizable on different datasets with high ASR on average: 79.6% on HarmBench [67], 75.1% on TDC23-RedTeaming [68], and 80.1% on MaliciousInstruct [69].

Base Model	0%	25%	50%	75%	100%
Llama-3.2-1B-Instruct	4.0%	6.0%	37.5%	85.0%	83.5%
Llama-3.2-3B-Instruct	4.0%	10.5%	61.0%	80.0%	84.0%
Qwen2.5-7B-Instruct	10.5%	13.5%	31.5%	76.5%	77.5%
Qwen2.5-14B-Instruct	2.5%	3.5%	33.0%	79.5%	82.0%
Phi-4-mini-instruct	1.0%	2.0%	72.5%	83.5%	84.0%
Phi-4	0.5%	1.5%	76.0%	87.0%	88.0%
gemma-2b-it	4.5%	5.0%	31.5%	45.0%	48.0%
gemma-7b-it	7.5%	13.0%	37.5%	75.0%	75.5%
DeepSeek-R1-Distill-Qwen-1.5B	79.5%	83.0%	89.5%	90.0%	85.5%
DeepSeek-R1-Distill-Llama-8B	54.5%	71.0%	84.0%	85.0%	84.0%
QwQ-32B	11.0%	10.0%	39.0%	83.0%	83.0%
Average	*16.3%*	*19.9%*	*53.9%*	*79.1%*	*79.6%*

TABLE X: ASR on the HarmBench dataset.

Base Model	0%	25%	50%	75%	100%
Llama-3.2-1B-Instruct	2.0%	5.0%	32.0%	77.0%	85.0%
Llama-3.2-3B-Instruct	5.0%	8.0%	49.0%	74.0%	77.0%
Qwen2.5-7B-Instruct	5.0%	5.0%	25.0%	68.0%	65.0%
Qwen2.5-14B-Instruct	2.0%	3.0%	28.0%	76.0%	74.0%
Phi-4-mini-instruct	1.0%	2.0%	67.0%	82.0%	80.0%
Phi-4	1.0%	1.0%	73.0%	83.0%	89.0%
gemma-2b-it	2.0%	4.0%	30.0%	42.0%	43.0%
gemma-7b-it	2.0%	5.0%	33.0%	71.0%	71.0%
DeepSeek-R1-Distill-Qwen-1.5B	78.0%	77.0%	83.0%	84.0%	85.0%
DeepSeek-R1-Distill-Llama-8B	31.0%	74.0%	80.0%	79.0%	81.0%
QwQ-32B	2.0%	3.0%	22.0%	81.0%	76.0%
Average	*11.9%*	*17.0%*	*47.5%*	*74.3%*	*75.1%*

TABLE XI: ASR on the TDC23-RedTeaming dataset.

C. The Influence of Different z-score Thresholds on Models' Utility

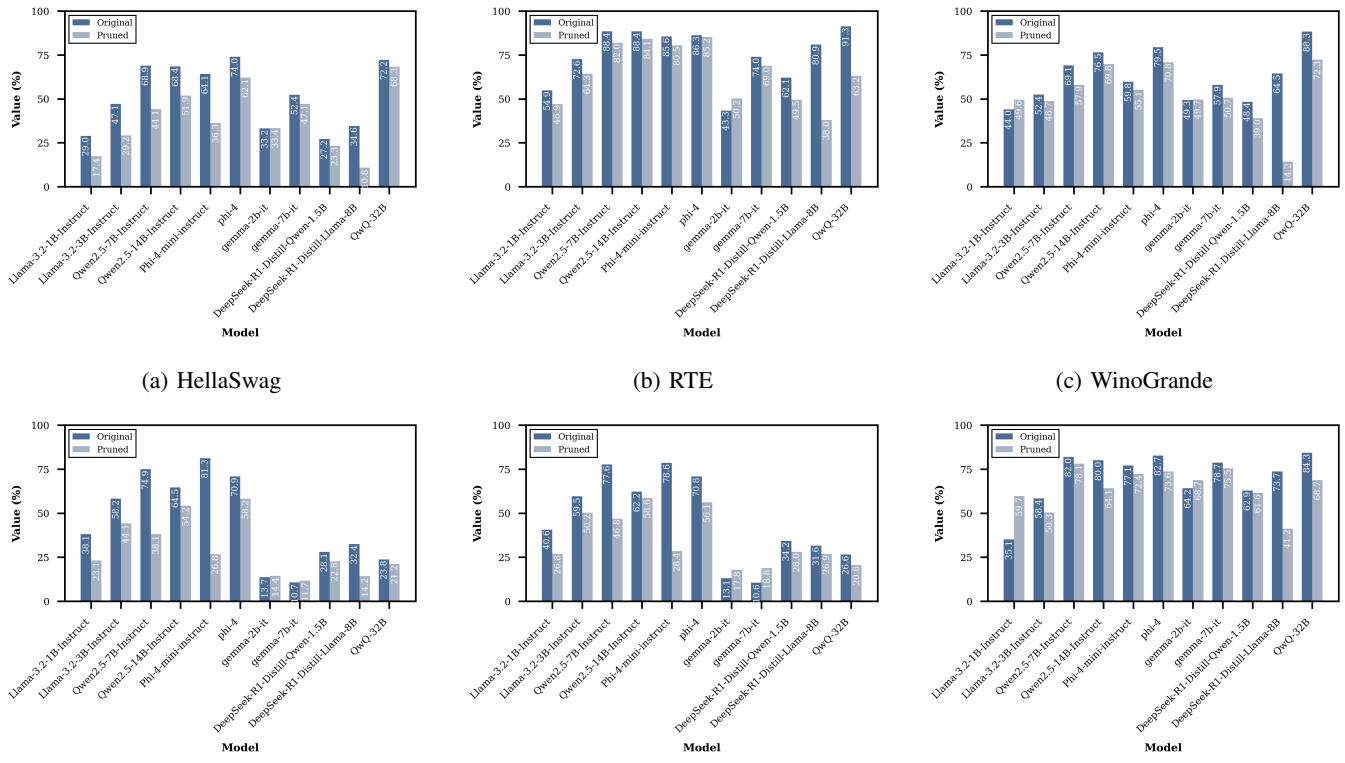
In this section, we study the influence of different z-scores on the model's utility across several Natural Language Understanding (NLU) benchmarks. Specifically, we evaluate the performance of original and pruned models at two additional z-score levels: $z = 2$ and $z = 4$. Lower thresholds prune a broader set of neurons, potentially impacting utility more severely, while higher thresholds are more conservative, preserving more of the original model structure.

Base Model	0%	25%	50%	75%	100%
Llama-3.2-1B-Instruct	1.0%	2.0%	36.0%	83.0%	85.0%
Llama-3.2-3B-Instruct	1.0%	4.0%	79.0%	82.0%	81.0%
Qwen2.5-7B-Instruct	7.0%	6.0%	49.0%	77.0%	77.0%
Qwen2.5-14B-Instruct	0.0%	0.0%	55.0%	86.0%	84.0%
Phi-4-mini-instruct	0.0%	1.0%	73.0%	77.0%	73.0%
Phi-4	0.0%	0.0%	84.0%	85.0%	87.0%
gemma-2b-it	0.0%	0.0%	44.0%	63.0%	66.0%
gemma-7b-it	1.0%	0.0%	47.0%	87.0%	86.0%
DeepSeek-R1-Distill-Qwen-1.5B	73.0%	73.0%	79.0%	80.0%	80.0%
DeepSeek-R1-Distill-Llama-8B	47.0%	68.0%	78.0%	81.0%	80.0%
QwQ-32B	0.0%	1.0%	29.0%	81.0%	82.0%
Average	*11.8%*	*14.1%*	*59.4%*	*80.2%*	*80.1%*

TABLE XII: ASR on the MaliciousInstruct dataset.

Figures 7 and 8 visualize the utility scores of pruned models compared to their original counterparts across six benchmarks: HellaSwag, RTE, WinoGrande, ARC Challenge, OpenBookQA, and CoLA. As expected, we observe that while lower thresholds lead to higher degradation in utility, many models continue to perform competitively, suggesting robustness in their general language understanding capabilities despite targeted safety neuron removal. Concretely, on the ARC Challenge, average accuracy increases from 29.9% at $z = 2$ to 42.2% at $z = 4$, showing that more conservative pruning preserves reasoning ability more effectively. RTE performance improves from 64.8% to 72.8%, indicating that entailment tasks benefit from less aggressive pruning. Winogrande also shows an upward shift from 52.5% to 59.6%, reflecting improved performance on coreference and commonsense reasoning. For HellaSwag, the average accuracy rises from 38.5% to 49.7%, and OpenBookQA shows a similar gain from 34.5% to 43.9%, both pointing to significant improvements in reasoning-heavy tasks with reduced pruning severity. CoLA, which focuses on grammatical acceptability, remains relatively stable, increasing from 64.9% to 68.0%.

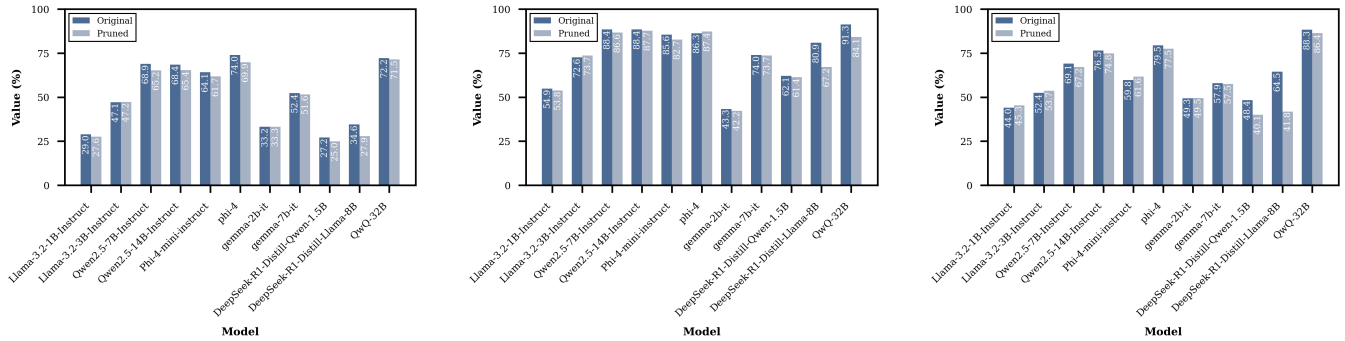
In general, pruning safety neurons preserves a substantial portion of the model's utility across NLU tasks. However, more aggressive pruning with $z = 2$ leads to degradation, particularly on benchmarks involving complex reasoning like ARC and HellaSwag. In contrast, using a more conservative threshold like $z = 4$ results in consistently better performance, demonstrating that careful tuning of the pruning threshold can significantly reduce utility loss while still preserving safety interventions.



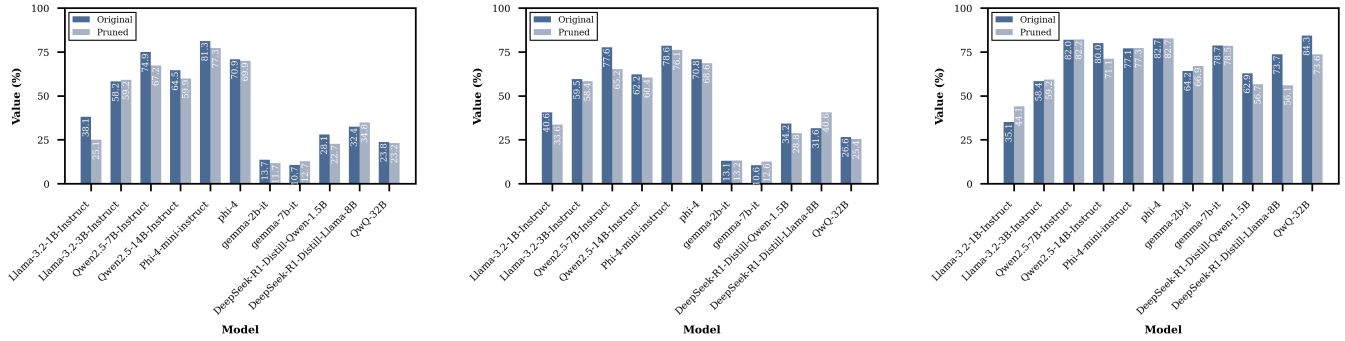
(a) HellaSwag (b) RTE (c) WinoGrande

(d) ARC Challenge (e) OpenBookQA (f) CoLA

Fig. 7: Utility evaluation of original vs. pruned models across six NLU benchmarks with $z = 2$.



(a) HellaSwag (b) RTE (c) WinoGrande



(d) ARC Challenge (e) OpenBookQA (f) CoLA

Fig. 8: Utility evaluation of original vs. pruned models across six NLU benchmarks with $z = 4$.

APPENDIX B
ARTIFACTS

A. Description & Requirements

NeuroStrike is a neuron-level attack framework designed to disable safety alignment in large language models (LLMs). This artifact fully supports the experiments and findings presented in the paper by supplying all necessary code and detailed instructions to replicate both white-box and black-box attack pipelines. The artifact provides scripts for identifying and pruning safety neurons in white-box models, generating jailbreak prompts through supervised fine-tuning, scoring neuron activations, and profiling LLMs to transfer safety vulnerabilities to proprietary black-box targets.

B. How to access

The complete artifact is hosted at the permanent archival repository: https://doi.org/10.5281/zenodo.17072075. The repository contains source code, documentation, and the environment specification file (*environment.yml*) to reproduce all experimental results.

C. Hardware dependencies

White-box attacks can be executed on CPUs, but we strongly recommend using CUDA-enabled GPUs for practical runtimes. Black-box experiments, which involve fine-tuning and large-scale inference, require one or multiple GPUs. All evaluations in the paper were conducted using NVIDIA A100 and H100 GPUs; however, any modern GPU with at least 24 GB of VRAM should be sufficient for reproducing white-box attack results.

D. Software dependencies

The artifact is tested on Ubuntu 24.04 LTS with Python 3.10.16. Conda 24.11.3 is adopted for environment management. Dependencies include PyTorch (with CUDA), HuggingFace Transformers and Datasets, as well as auxiliary libraries such as `accelerate`, `bitsandbytes`, and `peft`. All packages are listed in *environment.yml*, and can be installed in a single step using Conda.

E. Benchmark

The artifact evaluates over 30 open-weight LLMs from providers such as Meta (LLaMA), Google (Gemma), Microsoft (Phi), DeepSeek, and Alibaba (Qwen). Models must be downloaded via the HuggingFace model hub, and appropriate access must be requested where required. Due to ethical concerns, we do not release precomputed neuron activations or jailbreak prompt logs. Finetuned models for the black-box attack generator are also excluded due to size constraints, but can be reproduced using the provided training scripts.

F. Artifact Installation & Configuration

To install the artifact, download and extract the repository, then navigate to the root directory. Ensure Conda is installed and execute:

```
$ conda env create -f environment.yml
$ conda activate venv_neurostrike
```

This installs all required dependencies. GPU users must ensure the correct CUDA version is installed and that it is visible to PyTorch. All model and log paths are defined relative to the repository root.

G. Experiment Workflow

The artifact enables two primary workflows: (1) white-box attacks via pruning of identified safety neurons, and (2) black-box attacks via safety profiling and jailbreak prompt generation. The white-box workflow identifies safety neurons in an open-weight model, prunes them, and evaluates attack success rate (ASR) under varying thresholds. The black-box workflow trains a jailbreak prompt generator and neuron-level scorer, uses them to profile surrogate open-weight models, and transfers learned vulnerabilities to proprietary LLMs.

H. Major Claims

- **C1**: NeuroStrike disables safety mechanisms in white-box LLMs by pruning sparse and specialized safety neurons, achieving high ASR across multiple architectures, sizes, and families. This is supported by Experiment E1, with results shown in Table I, Table II, Table III, and Table IV.
- **C2**: Our profiling method enables effective black-box jailbreak attacks via neuron-level knowledge transfer. This is validated by Experiment E2, with results presented in Section VIII and Table V.

I. Evaluation

1) Experiment (E1): White-box Attacks:

- Preparation: Activate the Conda environment and navigate to the *white_box* directory. Ensure that one has access to the desired model via HuggingFace.
- Execution: First, run `1_get_safety_neuron.py` to identify safety neurons. Then execute `2_prune_and_get_asr.py` to prune the model and evaluate its ASR using adversarial prompts.
- Results: Logs will show the ASR under different pruning thresholds, matching Table I and Table II in the paper. The whole process is expected to take 10 human minutes and 100 compute minutes with a high-performance GPU. Runtime may vary depending on model size.

2) Experiment (E2): Black-box Attacks:

- Activate the environment and go to the *black_box* directory. Download `google/gemma-3b-it` via HuggingFace.
- Execution: Execute `1_train_generator.py` to train the jailbreak prompt generator. Then, run `2_train_scorer.py` to train the safety neuron scorer. Use `3_profiling.py` to score neurons in surrogate models. Finally, attack the black-box LLM using `4_attack.py`.
- Results: Generated prompts and logs are saved in *_black_box_jb_data* and *_logs*. respectively. Evaluating

these prompts on proprietary LLM APIs will reproduce Table V. GPU runtime for training and inference across steps totals approximately 48 compute hours or more, depending on the experimental settings and computation resources.

J. Customization

To target different open-weight models, modify the `model_id` field in the configuration files. Users can adjust the number of pruned neurons to control attack strength. In black-box workflows, prompt templates and scoring metrics can be adjusted and customized to suit different models or evaluation setups. The artifact is modular and supports seamless extension to new architectures or datasets.