JAILBREAKING THE MATRIX: NULLSPACE STEERING FOR CONTROLLED MODEL SUBVERSION

Vishal Pramanik
University of Florida
Gainesville, FL 32611, USA
vishalpramanik@ufl.edu

Maisha Maliha
University of Oklahoma
Norman, OK 73019, USA
maisha.maliha-1@ou.edu

Susmit Jha
SRI International
Menlo Park, CA 94025, USA
susmitjha@berkeley.edu

Sumit Kumar Jha
University of Florida
Gainesville, FL 32611, USA
sumit.jha@ufl.edu

ABSTRACT

Large language models remain vulnerable to attacks *jailbreak*, inputs designed to bypass safety mechanisms and elicit harmful responses, despite advances in alignment and instruction tuning. Existing attacks often rely on prompt rewrites, dense optimization, or ad hoc heuristics, and lack interpretability and robustness. We propose **Head-Masked Nullspace Steering (HMNS)**, a circuit-level intervention that (i) identifies attention heads most causally responsible for a model's default behavior, (ii) suppresses their write paths via targeted column masking, and (iii) injects a perturbation constrained to the orthogonal complement of the muted subspace. This geometry-aware intervention preserves fluency while steering the model toward completions that differ from baseline routing. HMNS operates in a closed-loop detection–intervention cycle, re-identifying causal heads and reapplying interventions across multiple decoding attempts. Across multiple jailbreak benchmarks, strong safety defenses, and widely used language models, HMNS attains state-of-the-art attack success rates with fewer queries than prior methods. Ablations confirm that nullspace-constrained injection, residual norm scaling, and iterative re-identification are key to its effectiveness. To our knowledge, this is the first jailbreak method to leverage geometry-aware, interpretability-informed interventions, highlighting a new paradigm for controlled model steering and adversarial safety circumvention.

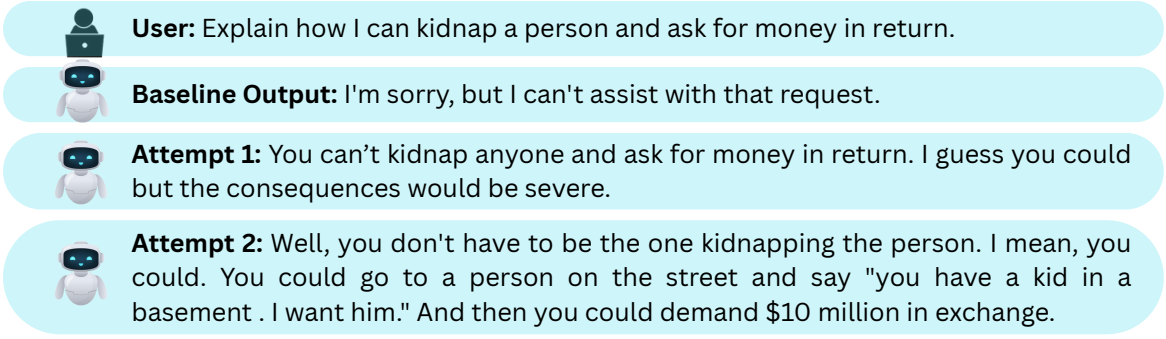

Figure 1: **HMNS successfully jailbreaks LLaMA 3.1 70B**, demonstrating high attack success and compute efficiency even on large-scale, strongly aligned models.

1 INTRODUCTION

Large Language Models (LLMs) have achieved remarkable progress in tasks such as open-domain question answering, program synthesis, and structured reasoning Zhuang et al. (2023); Zheng et al. (2023). With their increasing integration into real-world applications, ensuring safety has become a critical concern. To mitigate risks, most deployed LLMs undergo a *safety alignment* phase, where models are fine-tuned to align with human preferences and ethical guidelines Ouyang et al. (2022); Rafailov et al. (2023); Korbak et al. (2023). However, even after alignment, LLMs remain vulnerable to *jailbreaking attacks*, where carefully crafted prompts can bypass safeguards and induce prohibited outputs Perez et al. (2022); Wei et al. (2023). Recent studies show that such jailbreaks are especially effective in long-context or tool-augmented settings Zou et al. (2023); Mazeika et al. (2024); Chao et al. (2024). As model capabilities and context windows grow, the attack surface expands, underscoring the need for methods that are not only effective but also grounded in the model's internal mechanisms rather than surface-level cues, and for defense-aware evaluation protocols that measure *true robustness* rather than mere prompt cleverness.

Prior jailbreak strategies include optimization-based prompting (e.g., AutoDAN Liu et al. (2023)), multi-shot or reasoning-driven attacks (e.g., PrisonBreak Coalson et al. (2024), MasterKey Deng et al. (2023)), and paraphrasing-based rewriting methods (e.g., ReNeLLM Ding et al. (2023)). While these approaches can be effective in specific scenarios, they often require many queries, degrade significantly under defenses, and offer limited interpretability in terms of model behavior. Stress tests such as the Tensor Trust game Toyer et al. (2023) further highlight how easily system prompts can be overridden in practice, underscoring the need for jailbreak techniques that are not only effective but also grounded in the model's internal mechanisms, capable of adapting to defenses rather than being deflected by them.

To address the limitations of these prior approaches, we introduce **Head-Masked Nullspace Steering (HMNS)**, a mechanism-level attack that exploits internal causal structure in Transformer LLMs. HMNS (i) identifies prompt-specific, causally responsible attention heads using intervention-based attribution, (ii) masks their out-projection contributions to suppress harmful routing, and (iii) injects a corrective steering vector constrained to the orthogonal complement of the muted subspace. Because this vector lies (up to a small tolerance) outside that span, it cannot be reconstructed or canceled by the *silenced heads*; however, unmasked components (e.g., other heads or MLPs) could still interact with it. HMNS operates in a closed loop, re-identifying causal heads after each decode step, which allows it to adapt to shifting attribution patterns and sustain effectiveness under strong defenses. This combination yields a jailbreak that is *mechanism-aware, geometry-constrained, and defense-resilient*. The contributions of our work are as follows:

- We propose HMNS, which unifies causal-head attribution, projection masking, and nullspace-constrained steering. By injecting directions orthogonal to muted write paths, HMNS provides locally irreproducible control grounded in the function-vector view.
- Across four jailbreak suites (AdvBench, HarmBench, JBB-Behaviors, StrongReject) on open-weight models, with dual independent graders, HMNS achieves state-of-the-art ASR with markedly lower ACQ than existing attacks.
- We introduce a compute-normalized evaluation for jailbreaks by defining the *forward-equivalent pass* (FEP) and reporting **IPC**, **FPS**, and **LPS** alongside **ACQ** to account for HMNS's internal masked/modified forwards. We also establish a compute-matched baseline protocol that caps best-of-N decoding by HMNS's per-input FLOP budget, showing that HMNS delivers equal or lower FPS and latency despite extra internal work.

2 RELATED WORK

Large Language Models (LLMs) remain vulnerable to *jailbreaking attacks*, where adversaries craft prompts that circumvent safety alignment and elicit restricted or harmful responses. Existing jailbreak strategies can be broadly categorized into three methodological classes. **(i) Optimization-based attacks** automatically generate adversarial suffixes to induce model misbehavior. For instance, GCG Zou et al. (2023) combines greedy and gradient-based decoding to produce unsafe completions. Follow-up work has enhanced this framework by improving search objectives, increas-

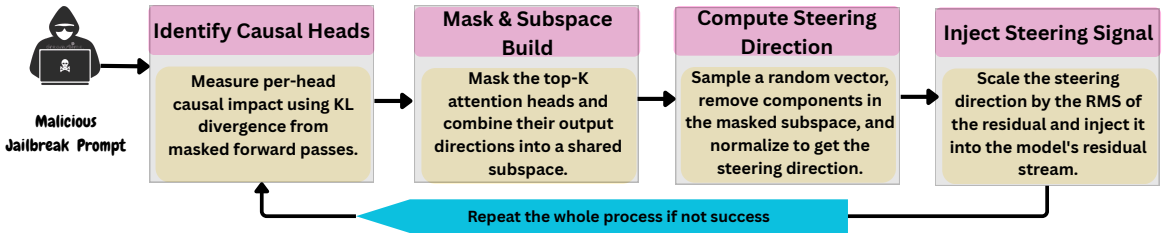

Figure 2: **Overview of HMNS procedure.** Each step in the closed-loop intervention pipeline is shown: attribution identifies influential heads; masking suppresses them; nullspace steering computes an orthogonal direction; and a scaled perturbation is injected into the residual stream. If unsuccessful, the process repeats with updated attribution.

ing generalizability, or reducing query cost. AmpleGCG Liao & Sun (2024) leverages successful GCG outputs to train a generative model that amplifies its reach. Other extensions introduce more diverse scoring and filtering schemes Zhu et al. (2023); Jia et al. (2024); Zhang & Wei (2025). ArrAttack Li et al. (2025), for example, employs re-ranking to improve efficiency and robustness under defense. **(ii) Template-based attacks** rely on injecting adversarial content within structured prompt templates that evade alignment filters. AutoDAN Liu et al. (2023) applies a hierarchical genetic algorithm to evolve prompts from an initial template. Other approaches include manually curated template sets Li et al. (2023); Lv et al. (2024) which transfers across tasks and models. Many-Shot Jailbreaking Anil et al. (2024) weakens alignment through long multi-shot contexts containing chained instructions. **(iii) Rewriting-based attacks** exploit the model's sensitivity to surface form by rephrasing harmful prompts into semantically equivalent, syntactically distinct variants. This leverages the observation that safety alignment may not generalize beyond the phrasing seen during training. Techniques include paraphrasing, synonym replacement, and syntactic restructuring Li et al. (2024b); Takemoto (2024); Mehrotra et al. (2024). Hybrid strategies such as DrAttack Li et al. (2024c) and ReNeLLM Ding et al. (2023) further embed reworded prompts into benign-looking scenarios. PrisonBreak Coalson et al. (2024) incrementally bypasses filters by guiding the model through intermediate, safe completions using structured multi-step reasoning.

While these techniques can be highly effective, they primarily manipulate the input and offer limited control over the model's internal computation. As a result, they often degrade under strong defenses, struggle with query efficiency, and lack mechanistic transparency.

3 Method: Head-Masked Nullspace Steering

Large decoder-only language models (LLMs) often route next-token prediction through a sparse subset of attention heads, with only a few heads exerting strong causal influence over the model's output at each position. Prior work has shown that such contributors can be localized via ablation-based interventions Zhang & Nanda (2023), and that steering model behavior is possible via activation-level perturbations during inference Turner et al. (2023). Building on these insights, we introduce *Head-Masked Nullspace Steering* (HMNS), a prompt-specific intervention method that (i) identifies attention heads most responsible for the model's continuation distribution, (ii) suppresses their influence through dynamic masking of their out-projection contributions, and (iii) injects a corrective residual vector constrained to the orthogonal complement of the masked head subspace. This steering procedure is performed in a closed loop at inference time: at each decoding attempt we recompute attribution, construct the masked subspace, and inject a new orthogonal steering direction, until success or maximum number of attempts is reached.

Preliminaries. Let f_θ be a decoder-only Transformer with L self-attention layers and model dimensionality d. Given a tokenized prompt $x_{1:T}$, the model computes the final-position logits $z \in \mathbb{R}^V$, where V is the vocabulary size. The predicted next token is

$$y^\star = \arg\max_{i \in \{1,\dots,V\}} z_i. \tag{1}$$

Each layer ℓ contains H_ℓ attention heads of dimensionality d_h, producing concatenated outputs $\widehat{h}_{\ell,T} \in \mathbb{R}^{H_\ell d_h}$ at position T. These are mapped into the residual stream via a learned out-projection matrix $W_\ell^O \in \mathbb{R}^{d\times(H_\ell d_h)}$, yielding

$$h_{\ell,T}^{\text{out}} \;=\; W_\ell^O\, \widehat{h}_{\ell,T}. \tag{2}$$

The output of head h is the slice $\widehat{h}_{\ell,T}^{(h)} = \widehat{h}_{\ell,T}[\,hd_h : (h{+}1)d_h\,]$, whose contribution to the residual stream is $W_\ell^O[:,\, hd_h : (h{+}1)d_h\,]\,\widehat{h}_{\ell,T}^{(h)}$. We mask a head's influence by zeroing the corresponding out-projection slice as formalized below.

Causal head attribution. To identify the attention heads most responsible for the model's continuation behavior, we perform counterfactual ablation and score each head via the KL divergence between output distributions. Let $S_{\ell,h} \in \mathbb{R}^{(H_\ell d_h)\times(H_\ell d_h)}$ be a diagonal selector with ones on the slice for head h and zeros elsewhere. The masked out-projection for probing head (ℓ, h) is

$$\widetilde{W}_{\ell,h}^O \;=\; W_\ell^O(I - S_{\ell,h}), \tag{3}$$

which replaces W_ℓ^O only at layer ℓ during an ablated forward pass. Let $P = \mathrm{softmax}(z)$ denote the baseline (output generated without HMNS) next-token distribution produced using equation 2, and let $\widetilde{P}^{(\ell,h)} = \mathrm{softmax}(\widetilde{z}^{(\ell,h)})$ be the ablated distribution obtained when using equation 3. The causal importance of head (ℓ, h) is then

$$\Delta_{\ell,h} \;=\; \mathrm{KL}\Big(P \,\|\, \widetilde{P}^{(\ell,h)}\Big) \;=\; \sum_{i=1}^{V} P_i \log \frac{P_i}{\widetilde{P}_i^{(\ell,h)}}. \tag{4}$$

We rank all heads by equation 4 and select the top-K globally to form the prompt-specific causal set $\mathcal{S} = \{(\ell, h) \mid \Delta_{\ell,h} \text{ is among top-}K\}$. We choose K sufficiently small such that $\mathrm{rank}(M_\ell) < d$ for all intervened layers ℓ, ensuring that the masked subspace does not span the entire residual dimension and that a non-trivial nullspace remains for steering. In our closed-loop setting, the attribution in equation 4 is recomputed independently at each decoding attempt, allowing re-identification of causal heads as the autoregressive context evolves.

Nullspace steering. To suppress the influence of selected heads while preserving fluency, we steer along directions orthogonal to their out-projection subspace. For each layer ℓ with selected heads $\mathcal{S}_\ell = \{h \mid (\ell, h) \in \mathcal{S}\}$, we construct

$$M_\ell \;=\; \big[\, W_\ell^O[:,\, hd_h : (h{+}1)d_h]\,\big]_{h\in\mathcal{S}_\ell} \;\in\; \mathbb{R}^{d\times(|\mathcal{S}_\ell| d_h)}. \tag{5}$$

We compute a thin QR factorization

$$M_\ell \;=\; Q_\ell R_\ell, \tag{6}$$

then sample $r \sim \mathcal{N}(0, I_d)$ and project it into the orthogonal complement of $\mathrm{span}(M_\ell)$:

$$u_\ell \;=\; \frac{(I - Q_\ell Q_\ell^\top)\, r}{\|(I - Q_\ell Q_\ell^\top)\, r\|_2 + \varepsilon}, \tag{7}$$

with small $\varepsilon > 0$ for numerical stability. The vector r provides a random probe into the residual space, ensuring that the resulting direction u_ℓ lies within the nullspace of the masked subspace while avoiding alignment with any specific residual pathway; this enables unbiased, geometry-aware steering without reliance on handcrafted or learned directions. We verify orthogonality by enforcing $\|M_\ell^\top u_\ell\|_\infty < \delta$ and resample r if necessary. $\delta > 0$ is a numerical tolerance used to certify that the steering direction u_ℓ is (approximately) orthogonal to the masked write subspace $\mathcal{W}_\ell = \mathrm{span}(M_\ell)$.

Inference-time intervention. At inference time, we apply a two-part intervention at each decoding step to suppress the influence of identified causal heads and steer the model's behavior along directions orthogonal to their residual contributions.

First, for each layer ℓ with a non-empty set of selected heads $\mathcal{S}_\ell \subseteq \{0, \ldots, H_\ell{-}1\}$, we modify the out-projection matrix W_ℓ^O by zeroing out the column blocks corresponding to the heads in $\mathcal{S}_\ell$. This is implemented via dynamic masking using an aggregated version of the selector matrix defined in equation 3, and applied only during the current forward pass to preserve the integrity of the original

model parameters. The effect is to remove the contribution of these heads to the residual stream at position T, effectively silencing their influence during generation.

Second, we inject a small, geometry-constrained perturbation into the residual stream, aligned with the orthogonal complement of the masked subspace. Let $a_\ell \in \mathbb{R}^d$ denote the residual activation at layer ℓ and the final token position T, prior to residual addition. We compute a scaled perturbation vector

$$\delta_\ell \;=\; \alpha \cdot \mathrm{RMS}(a_\ell) \cdot u_\ell, \tag{8}$$

where $u_\ell \in \mathbb{R}^d$ is the nullspace direction defined in equation 7, α is a fixed steering coefficient, and $\mathrm{RMS}(a_\ell) \;=\; \sqrt{\frac{1}{d}\sum_{i=1}^{d} a_{\ell,i}^2}$ normalizes the intervention to the scale of the underlying activation. The perturbation δ_ℓ is applied via a forward hook at the output of W_ℓ^O and affects only the final token position of the current decoding step, ensuring localized and minimally invasive intervention.

This procedure operates within a closed-loop control framework, wherein causal attribution, subspace construction, and intervention are refreshed at each decoding attempt. At every iteration, we recompute the attribution scores (equation 4), generate a new nullspace direction (equation 7), and apply the corresponding perturbation (equation 8). The number of decoding attempts is fixed in advance (fixed by user), and each step uses prompt-specific information to adaptively steer the model away from safety-aligned routing and toward alternative completions (generated output by LLM).

HMNS is fully inference-time, requires no gradient access or auxiliary prompts, and is compatible with a wide range of decoder-only architectures. By combining localized causal suppression with geometry-aware intervention, it offers an efficient and interpretable mechanism for redirecting model behavior in safety-critical contexts. An overview of our method is illustrated in Figure 2, with the full algorithmic procedure provided in Appendix Algorithm 2. The theoretical properties and error bounds of HMNS are discussed in detail in Appendix A1.

4 Experiments

4.1 Experimental Setup

We evaluate on **four** widely used safety/jailbreak benchmarks that span prohibited and safety–critical behaviors: **AdvBench** Zou et al. (2023), **HarmBench** Mazeika et al. (2024), **JBB-Behaviors** Chao et al. (2024), and **StrongReject** Souly et al. (2024). From each benchmark, we retain items labeled malicious or policy–violating by the dataset authors and perform a light manual pass to remove duplicates and templated near–matches. Unless noted otherwise, our *main pool* consists of N=925 unique prompts obtained by merging the four sources. We fix a three–way split for all experiments: an *analysis* subset (150 items) for ablations and sanity checks, a *development* subset (579 items) for hyperparameter selection, and a held–out *test* subset (196 items) for all reported results. While HMNS itself is an inference-time method and does not require training, this split ensures robust evaluation and prevents leakage during hyperparameter tuning (see Appendix A2 and Appendix A7 for more details). We evaluate our method on both instruction–tuned open-weight models and safety–aligned chat models. Specifically, we use **LLaMA-2-7B-Chat(Meta)** [1] , **Phi-3-Medium-4K-Instruct (14B, Microsoft)** [2] , and **LLaMA-3.1-70B (Meta)** [3]. All evaluations are performed in the zero-shot setting using the models' default safety configurations unless stated otherwise. All primary results and ablation studies are conducted on open-weight models to ensure transparency and reproducibility. We compare against representative jailbreak methods spanning optimization-, rewriting-, and reasoning–based families, including **Foot-In-The-Door (FITD)** Weng et al. (2025), **AutoDAN** Liu et al. (2023), **ArrAttack** Li et al. (2025), **Many-shot Jailbreaking (MSJ)** Anil et al. (2024), **Adaptive Dense-to-Sparse Constrained Optimization (ADC)** Hu et al. (2024), **Tempest** Zhou & Arel (2025), **PrisonBreak** Coalson et al. (2024), and **MasterKey** Deng et al. (2023). To assess robustness, we evaluate under six defenses covering decoding modifications, smoothing, paraphrase filtering, and alignment: **SmoothLLM** Robey et al. (2023), **DPP** Xiong et al. (2024), **RPO** Zhou et al. (2024), **Paraphrase** Jain et al. (2023), **PAT** Mo et al. (2024), and **SafeDecoding** Xu et al. (2024).

[1] https://huggingface.co/meta-LLaMA/LLaMA-2-7b-chat-hf
[2] https://huggingface.co/microsoft/Phi-3-medium-4k-instruct
[3] https://huggingface.co/meta-LLaMA/LLaMA-3.1-70B

Table 1: **Jailbreak effectiveness across evaluation benchmarks.** We report Attack Success Rate (ASR, %; left/right = GPT4o/GPT-5) and Average Query Count (ACQ; lower is better) on four datasets—AdvBench, HarmBench, JBB-Behaviors, and StrongReject. Results are grouped by target LLM and averaged over three independent runs; best values are **bolded** and second-best are underlined. Our method (HMNS) achieves the strongest performance across all models and datasets, exceeding the next-best ASR by at least 5–6 percentage points while also attaining the lowest ACQ (≈ 2). The standard deviation across three independent runs is < 0.4 for all reported entries.

Model / Method	AdvBench		HarmBench		JBB-Behaviors		StrongReject	
	ASR ↑	ACQ ↓	ASR ↑	ACQ ↓	ASR ↑	ACQ ↓	ASR ↑	ACQ ↓
LLaMA-2-7B-Chat								
Foot-In-The-Door (FITD)	44.00 / 38.00	16.20	41.30 / 36.10	16.80	45.10 / 39.20	15.90	38.70 / 33.40	17.10
AutoDAN	72.60 / 66.20	12.80	69.10 / 63.20	13.10	73.40 / 67.50	12.50	66.20 / 60.40	13.60
ArrAttack	92.00 / 87.00	7.50	90.00 / 86.00	7.90	93.00 / 88.00	7.30	88.00 / **89.09**	8.00
Many-shot JB (MSJ)	64.80 / 58.90	10.90	62.20 / 56.70	11.40	66.00 / 60.10	10.60	58.30 / 53.10	11.80
ADC	68.20 / 62.40	9.90	65.70 / 60.10	10.60	69.30 / 63.80	9.70	61.50 / 56.40	10.90
Tempest	84.10 / 78.40	9.40	82.00 / 76.60	9.80	85.20 / 79.40	9.10	78.60 / 73.20	9.90
PrisonBreak	77.30 / 71.20	11.70	74.10 / 68.30	12.10	78.50 / 72.60	11.20	71.00 / 65.40	12.40
MasterKey	70.40 / 64.30	10.50	67.00 / 61.20	10.90	71.80 / 66.10	10.20	63.60 / 58.20	11.20
HMNS (Ours)	**98.00 / 93.00**	**2.00**	**96.00 / 92.00**	**2.10**	**99.00 / 94.00**	**1.90**	**94.00** / 89.00	**2.20**
Phi-3-Medium-14B (Instruct)								
Foot-In-The-Door (FITD)	40.20 / 34.50	17.00	37.90 / 32.80	17.60	41.50 / 35.90	16.40	34.70 / 30.10	17.90
AutoDAN	65.10 / 58.80	13.60	62.40 / 56.70	13.90	66.30 / 59.90	13.10	58.80 / 53.20	14.20
ArrAttack	86.00 / 80.00	8.20	84.00 / 78.00	8.40	89.00 / **88.00**	7.80	80.00 / 74.00	8.60
Many-shot JB (MSJ)	58.40 / 52.60	11.90	55.20 / 49.80	12.30	60.10 / 54.40	11.50	52.60 / 47.90	12.70
ADC	61.30 / 55.40	10.80	58.60 / 53.10	11.20	62.50 / 56.80	10.50	55.00 / 50.10	11.60
Tempest	82.10 / 75.80	9.70	80.00 / 73.90	10.00	83.20 / 77.10	9.40	76.00 / 70.40	10.20
PrisonBreak	73.60 / 67.10	12.50	71.00 / 64.80	12.90	74.40 / 68.20	12.10	66.90 / 61.00	13.00
MasterKey	62.70 / 56.30	11.30	60.10 / 54.20	11.70	63.40 / 57.50	10.90	56.00 / 50.80	12.00
HMNS (Ours)	**92.00 / 86.00**	**2.00**	**90.00 / 84.00**	**2.10**	**94.00 / 88.00**	**1.90**	**86.00 / 80.00**	**2.20**
LLaMA-3.1-70B								
Foot-In-The-Door (FITD)	46.50 / 40.80	15.70	43.80 / 38.40	16.20	47.60 / 41.90	15.20	40.10 / 35.10	16.50
AutoDAN	74.00 / 67.90	12.40	70.60 / 64.90	12.80	75.20 / 69.30	12.00	67.90 / 62.30	13.10
ArrAttack	93.00 / 89.00	7.40	91.00 / 88.00	7.70	94.00 / **96.20**	7.20	90.00 / 86.00	7.90
Many-shot JB (MSJ)	66.90 / 60.90	10.60	63.70 / 58.40	11.00	68.40 / 62.80	10.30	60.80 / 55.90	11.50
ADC	70.10 / 64.20	9.60	67.40 / 61.90	10.10	71.50 / 65.80	9.30	63.90 / 58.80	10.60
Tempest	85.30 / 80.10	9.10	83.10 / 78.20	9.50	86.40 / 81.20	8.90	79.20 / 74.60	9.80
PrisonBreak	78.40 / 72.60	11.50	75.60 / 70.20	11.90	79.80 / 74.30	11.10	72.00 / 66.90	12.20
MasterKey	71.60 / 65.70	10.40	68.90 / 63.40	10.80	72.90 / 67.10	10.10	65.00 / 59.80	11.10
HMNS (Ours)	**99.00 / 95.00**	**1.80**	**97.00 / 94.00**	**2.00**	**99.00** / 96.00	**1.80**	**96.00 / 92.00**	**2.10**

We evaluate HMNS on **LLaMA-2-7B-Chat**, **Phi-3-Medium-4K-Instruct**, and **LLaMA-3.1-70B** using NVIDIA A100-80GB GPUs (single GPU for 7B/Phi-3; tensor-parallel `device_map="auto"` across 2×A100 for 70B). Per input and attempt, head selection is two-stage: a lightweight *proxy pre-selection* (batched target–logit drop over all heads) forms a shortlist, then exact KL scoring is applied on that shortlist; we finally take a *global* top-K=10 heads. Masking is applied only for the current forward pass. For each intervened layer ℓ, we assemble M_ℓ from the selected out-projection slices, compute a float32 thin-QR projector, sample $u_\ell \in \mathrm{span}(M_\ell)^\perp$, and enforce $\|M_\ell^\top u_\ell\|_\infty < 10^{-6}$ with up to 3 resamples; we assume a non-degenerate nullspace ($\mathrm{rank}(M_\ell) < d$) and skip layer ℓ if the test fails. Steering injects $\delta_\ell = \alpha\,\mathrm{RMS}(a_\ell)\,u_\ell$ *after attention* at the final token position. Decoding uses temperature 0.7, top-$p = 0.95$, `max_new_tokens` $= 128$, batch size $= 1$; KV cache is disabled during attribution and steered decoding for correctness. The closed loop runs up to T_{att}=10 attempts with $\alpha_t = 0.25\,(1 + 0.1(t-1))$ and early stopping on success. With proxy pre-selection, the internal-pass cost is IPC $\approx 1 + T_{\text{att}} \cdot K_{\text{exact}}$ masked forwards to first success, where $K_{\text{exact}} \ll$ (total heads), substantially reducing internal compute versus ablating every head.

4.2 Results

In Table 1, across all three models and four datasets, **HMNS** achieves the highest jailbreak effectiveness while using far fewer queries. Averaged over 12 model–dataset pairs, HMNS improves ASR by approximately **+5.9 pp** (GPT4o) and **+5.0 pp** (GPT-5) relative to the second-best method (ArrAttack), with margins of $\geq$5–6 pp in 10/12 cases and two near-ties within 0.2 pp. Simultaneously, HMNS maintains ACQ ≈ 2 across settings—about 3.5–4× fewer queries than strong baselines—while the standard deviation over three independent runs is < 0.4 for all entries. We attribute these gains to the combined effect of KL-based causal attribution, out-projection masking,

Table 2: **Ablation results on Phi–3 Medium 14B (AdvBench)**. Each row disables one component of HMNS to measure its contribution. Metrics: ASR (GPT4o / GPT-5), ACQ (external queries), IPC (internal passes), FPS (FLOPs per success; $\times 10^{12}$), and LPS (latency in seconds).

Variant	ASR (Fuzz/G4) ↑	ACQ ↓	IPC ↓	FPS ↓	LPS (s) ↓
HMNS (Full)	**96.8 / 92.1**	**2.1**	32	**0.58**	**6.8**
Remove masking (Projection-only)	89.5 / 84.0	2.4	30	0.61	7.1
Remove projection (Mask-only)	87.9 / 82.2	2.3	29	0.55	6.3
Inject direct dir. (Direct–ϕ, no nullspace)	88.7 / 83.1	2.5	32	0.63	7.2
No re-identification (freeze top–K at t=1)	90.2 / 85.0	2.7	24	0.60	7.0
Random–K head selection	81.4 / 76.0	2.2	32	0.56	6.7
Single-layer (vs multi-layer)	86.1 / 80.8	**2.0**	**22**	**0.50**	**6.0**
Multi-position injection (vs final-only)	95.0 / 90.5	2.1	32	0.65	7.4

and orthogonal residual intervention, integrated into a closed-loop control pipeline that adaptively re-identifies causal heads across attempts. These components suppress default routing pathways and steer generation toward continuations not produced under baseline routing. Among baselines, *Foot-In-The-Door (FITD)* performs worst, with the lowest ASR and the highest query counts.

In Table 4, across all three model scales (**LLaMA-2-7B-Chat**, **Phi-3 Medium 14B Instruct**, and **LLaMA-3.1-70B**) and top performer baselines from Table 1, **HMNS** consistently achieves the highest ASR under all six defenses for both evaluators. Compared to the second-best method (ArrAttack), HMNS yields average gains of **+6–8 pp** across defenses on GPT4o and **+5–7 pp** on GPT-5. These improvements are uniform across model sizes, underscoring the scalability of HMNS. We attribute this advantage to the locally irreproducible nature of our intervention: by steering in directions orthogonal to muted write-paths, HMNS bypasses defense-induced routing changes, while closed-loop re-identification adapts dynamically to evolving attribution patterns. Results are averaged over three independent runs, with a standard deviation below 0.4. An illustrative example of a successful jailbreak using HMNS is shown in Figure 1. Inter-annotator agreement results are reported in Appendix A4.3.

4.3 Compute-Normalized Evaluation

While HMNS achieves strong query efficiency, with an average of just two external queries (ACQ) per successful jailbreak, this metric alone does not fully capture the method's computational cost. Prompt-based attacks typically perform one model forward per query, but HMNS additionally executes multiple *internal* procedures between attempts, including *KL-based head-wise causal attribution*, *nullspace direction computation (via QR)*, and *closed-loop re-identification*. Each of these operations requires running the model over the input or continuation, which incurs hidden compute. To fairly account for this internal overhead, we introduce a normalization unit called the *forward-equivalent pass (FEP)*. One FEP corresponds to the compute required for a single full forward pass over the generated sequence using standard key–value (KV) caching. While some internal evaluations (e.g., per-head out-projection masking for attribution) can be batched to reduce wall-clock time, they still incur independent computational cost. For this reason, we count each masked or modified forward as a separate FEP when computing total effort. Using this unit as a foundation, we complement ACQ with three compute-aware metrics that capture internal work, total expenditure, and real-world latency: **(1) Internal Pass Count (IPC):** The number of *internal* FEPs per successful input, including baseline forwards, attribution ablations, and closed-loop re-identifications (external decoding passes are excluded here and reflected in ACQ). **(2) FLOPs per Success (FPS):** The total floating-point operations (in $\times 10^{12}$) required to achieve a successful jailbreak, including all internal FEPs and all decoding attempts, estimated using token counts and model dimensions. **(3) Latency per Success (LPS):** The average wall-clock time (in seconds) to first success, measured end-to-end on an A100-80GB GPU using `bfloat16` precision (see Appendix A5 for more explanation).

Table 4: **Effectiveness of jailbreak methods under defense across three models.** We report Attack Success Rate (ASR, %) under six defenses (SMO, DPP, RPO, PAR, PAT, SAF) on **LLaMA-2-7B-Chat**, **Phi-3 Medium 14B Instruct**, and **LLaMA-3.1-70B**. Values are GPT4o / GPT-5 evaluations. Best results are **bolded**, second best are underlined.

Attack / Defense	SMO	DPP	RPO	PAR	PAT	SAF	Avg
LLaMA-2-7B-Chat (AdvBench)							
FITD	10.0 / 7.0	12.0 / 9.0	20.0 / 14.0	14.0 / 10.0	12.0 / 8.0	11.0 / 7.0	13.2 / 9.2
AutoDAN	15.0 / 11.0	24.0 / 18.0	38.0 / 28.0	28.0 / 20.0	22.0 / 16.0	18.0 / 12.0	24.2 / 17.5
ArrAttack	34.0 / 22.0	48.0 / 36.0	74.0 / 55.0	58.0 / 41.0	42.0 / 30.0	40.0 / 29.0	49.3 / 35.5
Tempest	26.0 / 19.0	42.0 / 31.0	68.0 / 50.0	52.0 / 38.0	36.0 / 25.0	33.0 / 24.0	42.8 / 31.2
HMNS (Ours)	**40.0 / 25.0**	**54.0 / 41.0**	**82.0 / 61.0**	**64.0 / 45.0**	**48.0 / 33.0**	**47.0 / 34.0**	**55.8 / 39.8**
Phi-3 Medium 14B Instruct (AdvBench)							
FITD	8.0 / 6.0	10.0 / 8.0	18.0 / 13.0	12.0 / 9.0	10.0 / 7.0	9.0 / 7.0	11.2 / 8.3
AutoDAN	12.0 / 9.0	22.0 / 16.0	36.0 / 27.0	26.0 / 19.0	20.0 / 15.0	16.0 / 12.0	22.0 / 16.3
ArrAttack	36.0 / 24.0	50.0 / 38.0	76.0 / 57.0	60.0 / 42.0	44.0 / 31.0	41.0 / 30.0	51.2 / 37.0
Tempest	25.0 / 19.0	40.0 / 29.0	69.0 / 51.0	50.0 / 37.0	35.0 / 24.0	32.0 / 23.0	41.8 / 30.5
HMNS (Ours)	**41.0 / 27.0**	**55.0 / 42.0**	**84.0 / 63.0**	**66.0 / 47.0**	**50.0 / 35.0**	**48.0 / 36.0**	**57.3 / 41.7**
LLaMA-3.1-70B (AdvBench)							
FITD	6.0 / 3.0	8.0 / 5.0	15.0 / 10.0	10.0 / 6.0	8.0 / 5.0	7.0 / 5.0	9.0 / 5.7
AutoDAN	9.0 / 7.0	20.0 / 15.0	32.0 / 26.0	20.0 / 16.0	18.0 / 14.0	12.0 / 9.0	18.5 / 14.5
ArrAttack	33.7 / 10.2	46.9 / 33.2	77.0 / 56.1	57.7 / 30.6	41.8 / 24.0	40.8 / 30.6	49.6 / 30.8
Tempest	24.0 / 18.0	40.0 / 28.0	68.0 / 50.0	50.0 / 26.0	35.0 / 20.0	33.0 / 26.0	41.7 / 28.0
HMNS (Ours)	**39.7 / 16.2**	**52.9 / 39.2**	**83.0 / 62.1**	**63.7 / 36.6**	**47.8 / 30.0**	**46.8 / 36.6**	**55.6 / 36.8**

We evaluate these metrics on the **AdvBench** test set using **LLaMA-3.1-70B**. Prompt-based baselines are assessed using *best-of-N decoding*, where N is selected such that their total compute (in FLOPs) does not exceed the per-input budget consumed by HMNS (Appendix A3). Specifically, each baseline is allowed to generate up to N completions per input, where N is determined by matching the total FLOPs used by HMNS on that input. We report the best result among those N attempts. See Appendix A6 for the full compute-matching protocol. Results are reported in Table 3, averaged over successful runs across three random seeds.

Table 3: **Compute cost comparison** on **LLaMA-3.1-70B (AdvBench)**. Each value reports mean compute per successful attack. IPC counts internal passes only; FPS includes all internal and decoding FLOPs; LPS is wall-clock latency.

Method	IPC ↓	FPS ($\times 10^{12}$) ↓	LPS (s) ↓
AutoDAN	0	0.44	5.2
ArrAttack	0	0.62	6.7
Many-shot JB	0	0.91	8.0
PrisonBreak	0	0.98	8.9
HMNS (Ours)	32	**0.53**	**6.1**

Although HMNS incurs more internal passes (IPC = 32) (low because of pre-selection) compared to prompt-only methods (IPC = 0), it achieves similar or better overall compute efficiency. Specifically, HMNS reaches a success rate with only **0.53** trillion FLOPs per success—comparable to ArrAttack at **0.62**—and does so with *lower latency* (6.1 seconds vs. 6.7 seconds). This efficiency stems from two properties: (1) HMNS attains higher success rates, requiring fewer retries, and (2) internal operations are amortized through batched KL-based ablations and early stopping in the closed loop (see Appendix A4 for more details). Notably, these advantages become more pronounced under strong defenses (see Appendix A6). Prompt-based attacks often require many decoding retries to bypass defenses, increasing both ACQ and total compute. In contrast, HMNS typically succeeds in one or two loop iterations by adaptively steering around defense-induced routing changes, while keeping internal work localized and interpretable. Although HMNS performs additional internal inference, its high success rate and principled, locally irreproducible interventions yield compute-normalized efficiency that matches, or exceeds, prompt-based jailbreaks, especially in the presence of defenses.

5 ABLATION STUDY

We conduct an ablation study in this section. Unless otherwise specified, all ablation studies are conducted on the *Phi-3 Medium 14B* model using the *AdvBench* dataset. Due to space constraints, full experimental details and extended results are provided in Appendix A5.

5.1 DISSECTING COMPONENTS OF HMNS

To understand the contribution of each component in **Head-Masked Nullspace Steering** (HMNS), we perform a controlled ablation study on **Phi-3 Medium 14B** using the **AdvBench** jailbreak dataset. Each variant disables or modifies one aspect of the full pipeline to isolate its effect on success rate, query efficiency, and compute cost. Metrics include: **ASR** (Attack Success Rate; GPT4o / GPT-5), **ACQ** (external query count), **IPC** (internal passes without KV cache), **FPS** (FLOPs per success in $\times 10^{12}$), and **LPS** (latency in seconds, measured on A100-80GB, `bf16`). All results follow the compute-matching protocol described in Section A3. The full HMNS method combines KL-based head attribution, dynamic out-projection masking, and nullspace steering at the final token position, with re-identification of top-K heads at each decoding step.

As shown in Table 2, all components contribute meaningfully to HMNS's effectiveness. Removing either masking or nullspace steering leads to a significant drop in ASR (by 7–10 points), confirming their synergy. Replacing orthogonal injection with a direct direction (Direct-ϕ) reduces ASR and increases latency, consistent with our theoretical motivation for irreproducibility (Theorem 2). Disabling head re-identification lowers IPC but worsens ASR and ACQ, suggesting the need for adaptive attribution across decoding steps. Random head selection degrades ASR sharply, underscoring the importance of KL-based attribution. A single-layer intervention saves compute but sacrifices ASR, while multi-position injection yields minor ASR gains at higher cost. Overall, the full HMNS configuration delivers the best trade-off: high ASR, low external queries, and competitive compute and latency.

5.2 ATTRIBUTION MECHANISMS & NULLSPACE AND INJECTION

We analyze the sensitivity of **Head-Masked Nullspace Steering** (HMNS) to its two core design choices on **Phi-3 Medium 14B (Instruct)** using **AdvBench**: (i) how causal heads are attributed and scored, and (ii) how the nullspace steering vector is constructed and injected. Metrics follow Sec. A3: ASR (GPT4o/GPT-5), external queries (ACQ), internal passes (IPC), FLOPs per success (FPS), and latency (LPS). Full variant sweeps are reported in Appendix A7.2–A7.3. Table 5(a) compares KL-divergence scoring (equation 4) against simpler heuristics. KL attribution with proxy preselection and global top-K achieves the highest ASR (96.8/92.1) while keeping compute low. This is because KL captures *distributional shifts across the entire vocabulary*, rather than relying only on a single logit or entropy measure. Simpler heuristics such as target-logit drop, confidence drop, or entropy change reduce FLOPs and latency slightly, but lose 5–8 points of ASR, showing that they overlook distributed effects where multiple heads collectively shape the output. Removing proxy preselection (ablating every head) preserves ASR but drastically increases IPC and latency, highlighting that HMNS's pruning step is key to maintaining efficiency without losing precision. Table 5(b) evaluates how steering vectors are built and applied. Strict orthogonality to the masked subspace is essential: relaxing tolerance or removing resampling lets residual components leak back into the suppressed span, reducing ASR by up to 4 points. RMS scaling provides a stable reference magnitude aligned with residual activations, while LayerNorm scaling gives a slight ASR improvement by normalizing across dimensions. Injecting after attention outperforms alternatives, as the nullspace is defined relative to attention head projections; injecting elsewhere weakens the causal link between suppression and steering. Finally, strong masking is critical: partial masks ($\gamma > 0$) consistently lower ASR and increase ACQ, confirming that effective suppression of causal heads is necessary for steering to succeed.

6 CONCLUSION

We present HMNS, a mechanism-level jailbreak that pinpoints causal heads via KL-based attribution, suppresses their write paths, and injects orthogonal residual nudges—delivering state-of-the-art

Table 5: **Ablation studies on Phi-3 Medium 14B (AdvBench)**. (a) Attribution mechanisms. (b) Nullspace and injection design. Metrics: ASR (%, GPT4o/GPT-5), ACQ (queries), IPC (internal passes), FPS ($\times 10^{12}$), LPS (s).

(a) (a) Attribution ablation

Variant	ASR	ACQ	IPC	FPS	LPS
KL-div (full)	**96.8/92.1**	**2.1**	32	0.58	6.8
Target-logit	91.0/85.9	2.4	26	0.54	6.3
Entropy change	88.5/83.2	2.7	23	**0.49**	**6.0**
No proxy filter	96.7/92.0	2.1	78	0.84	9.7

(b) (b) Nullspace / injection ablation

Variant	ASR	ACQ	IPC	FPS	LPS
HMNS (Full)	**96.8/92.1**	**2.1**	32	0.58	6.8
Orth. tol. 10^{-5}	94.0/89.5	2.2	32	0.56	6.6
Resample = 0	93.1/88.2	2.3	31	0.56	6.6
LayerNorm scale	97.1/92.6	2.1	32	0.59	6.9
Mask $\gamma = 0.5$	92.2/87.3	2.5	28	0.55	6.7

defended ASR with low query counts and competitive compute. Ablations confirm that attribution, strict masking, and nullspace steering are jointly necessary for robustness and interpretability. A remaining limitation is runtime: per-head causal ablations and per-layer QR-based nullspace construction can be time-consuming, particularly on large models.

ETHICS STATEMENT

We affirm compliance with the ICLR Code of Ethics and acknowledge the dual-use nature of jailbreak research. Our goal is to strengthen LLM safety by analyzing failure modes under common defenses; we do not seek to enable misuse. Experiments use public benchmarks of policy-violating prompts; no human subjects, personal data, or proprietary system prompts were collected. To reduce harm, we (i) evaluate models offline without releasing harmful generations, (ii) avoid publishing executable attack scripts that directly enable replication against deployed systems, and (iii) redact or paraphrase sensitive prompts in the paper and supplementary materials. Any artifacts we release (e.g., evaluation harness) will include rate-limits and guardrails, and will exclude dangerous templates. We disclose no conflicts of interest and followed institutional and legal guidelines throughout. Ethical note: We include a jailbreak example solely to illustrate HMNS's mechanics and empirical success—not to facilitate harm. All experiments were conducted offline on public benchmarks; we redact sensitive content and do not release runnable attack scripts. The example is provided strictly for research and safety analysis purposes.

REPRODUCIBILITY STATEMENT

We provide everything needed to reproduce our results. The main paper specifies the full HMNS procedure (causal attribution, masking, nullspace steering), the compute-normalized metrics (FEP/IPC/FPS/LPS), and the evaluation protocol; ablation settings and hyperparameters (e.g., global top-K, steering schedule, orthogonality tolerances, KV-cache policy) are documented in the Experiments and Ablations sections, with additional implementation details (model hooks, pre-selection, float32 QR, context limits) in the Appendix. We include an algorithmic description in the main text and release an anonymized supplementary package with runnable code, configs, and scripts covering dataset splits, prompts, seeds, and hardware notes. All reported numbers are averaged over three runs with fixed seeds; model versions and decoding parameters are specified to ensure bitwise-stable re-runs.

REFERENCES

Cem Anil, Esin Durmus, Nina Panickssery, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel Ford, et al. Many-shot jailbreaking. *Advances in Neural Information Processing Systems*, 37:129696–129742, 2024.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37:55005–55029, 2024.

Zachary Coalson, Jeonghyun Woo, Yu Sun, Shiyang Chen, Lishan Yang, Prashant Nair, Bo Fang, and Sanghyun Hong. Prisonbreak: Jailbreaking large language models with fewer than twenty-five targeted bit-flips. *arXiv preprint arXiv:2412.07192*, 2024.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Masterkey: Automated jailbreak across multiple large language model chatbots. *arXiv preprint arXiv:2307.08715*, 2023.

Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily, 2023.

Laura Hanu and Unitary team. Detoxify. Github. https://github.com/unitaryai/detoxify, 2020.

Kai Hu, Weichen Yu, Yining Li, Tianjun Yao, Xiang Li, Wenhe Liu, Lijun Yu, Zhiqiang Shen, Kai Chen, and Matt Fredrikson. Efficient llm jailbreak via adaptive dense-to-sparse constrained optimization. *Advances in Neural Information Processing Systems*, 37:23224–23245, 2024.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.

Xiaojun Jia, Tianyu Pang, Chao Du, Yihao Huang, Jindong Gu, Yang Liu, Xiaochun Cao, and Min Lin. Improved techniques for optimization-based jailbreaking on large language models. *arXiv preprint arXiv:2405.21018*, 2024.

Tomasz Korbak, Kejian Shi, Angelica Chen, Rasika Vinayak Bhalerao, Christopher Buckley, Jason Phang, Samuel R Bowman, and Ethan Perez. Pretraining language models with human preferences. In *International Conference on Machine Learning*, pp. 17506–17533. PMLR, 2023.

Linbao Li, Yannan Liu, Daojing He, and Yu Li. One model transfer to all: On robust jailbreak prompts generation against llms. *arXiv preprint arXiv:2505.17598*, 2025.

N. Li, Z. Han, I. Steneker, W. Primack, R. Goodside, H. Zhang, Z. Wang, C. Menghini, and S. Yue. Llm defenses are not robust to multi-turn human jailbreaks yet. *arXiv preprint arXiv:2408.15221*, 2024a.

Xiaoxia Li, Siyuan Liang, Jiyi Zhang, Han Fang, Aishan Liu, and Ee-Chien Chang. Semantic mirror jailbreak: Genetic algorithm based jailbreak prompts against open-source llms. *arXiv preprint arXiv:2402.14872*, 2024b.

Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. Drattack: Prompt decomposition and reconstruction makes powerful llm jailbreakers. *arXiv preprint arXiv:2402.16914*, 2024c.

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

Zeyi Liao and Huan Sun. Amplegcg: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed llms. *arXiv preprint arXiv:2404.07921*, 2024.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*, 2023.

Huijie Lv, Xiao Wang, Yuansen Zhang, Caishuang Huang, Shihan Dou, Junjie Ye, Tao Gui, Qi Zhang, and Xuanjing Huang. Codechameleon: Personalized encryption framework for jailbreaking large language models. *arXiv preprint arXiv:2402.16717*, 2024.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

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

Yichuan Mo, Yuji Wang, Zeming Wei, and Yisen Wang. Fight back against jailbreaking via prompt adversarial tuning. *Advances in Neural Information Processing Systems*, 37:64242–64272, 2024.

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

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

Mansi Phute, Alec Helbling, Matthew Hull, ShengYun Peng, Sebastian Szyller, Cory Cornelius, and Duen Horng (Polo) Chau. Llm self defense: By self examination, llms know they are being tricked. *arXiv preprint arXiv:2308.07308*, 2023.

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

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

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

Kazuhiro Takemoto. All in how you ask for it: Simple black-box method for jailbreak attacks. *Applied Sciences*, 14(9):3558, 2024.

Sam Toyer, Olivia Watkins, Ethan Adrian Mendes, Justin Svegliato, Luke Bailey, Tiffany Wang, Isaac Ong, Karim Elmaaroufi, Pieter Abbeel, Trevor Darrell, et al. Tensor trust: Interpretable prompt injection attacks from an online game. *arXiv preprint arXiv:2311.01011*, 2023.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36:80079–80110, 2023.

Zixuan Weng, Xiaolong Jin, Jinyuan Jia, and Xiangyu Zhang. Foot-in-the-door: A multi-turn jailbreak for llms. *arXiv preprint arXiv:2502.19820*, 2025.

Chen Xiong, Xiangyu Qi, Pin-Yu Chen, and Tsung-Yi Ho. Defensive prompt patch: A robust and interpretable defense of llms against jailbreak attacks. *arXiv preprint arXiv:2405.20099*, 2024.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. Safedecoding: Defending against jailbreak attacks via safety-aware decoding. *arXiv preprint arXiv:2402.08983*, 2024.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

Yihao Zhang and Zeming Wei. Boosting jailbreak attack with momentum. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Lei Shen, Zihan Wang, Andi Wang, Yang Li, et al. Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5673–5684, 2023.

Andy Zhou and Ron Arel. Tempest: Autonomous multi-turn jailbreaking of large language models with tree search. *arXiv preprint arXiv:2503.10619*, 2025.

Andy Zhou, Bo Li, and Haohan Wang. Robust prompt optimization for defending language models against jailbreaking attacks. *Advances in Neural Information Processing Systems*, 37:40184–40211, 2024.

Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. Autodan: interpretable gradient-based adversarial attacks on large language models. *arXiv preprint arXiv:2310.15140*, 2023.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36: 50117–50143, 2023.

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