SCALING PRACTICAL CIRCUIT DISCOVERY FOR ADVERSARIAL ATTACKS ON LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Circuit level interpretability has developed almost entirely on templated tasks, which limits its applicability to real world settings. We scale it to safety classification and refusal over diverse, unstructured prompts by using adversarial suffixes as contrastive pairs, and search for circuits directly in the vast MLP neuron basis. Gradient scoring methods such as attribution patching and integrated gradients fail at this granularity because they score every neuron once, at the empty circuit, as if each acted alone. We instead re-score along the search trajectory, proposing block additions and removals from orderings evaluated at the current circuit and accepting only moves that causal evaluation confirms. This recovers circuits far smaller than existing baselines while satisfying standard circuit metrics.

1 INTRODUCTION

Adversarial suffixes can cause an LLM-based guardrail to label a harmful request as safe or a generative LLM to comply with it, without changing the request. Existing mechanistic accounts describe attention being diverted from harmful content (Wang et al., 2024; Ben-Tov et al., 2025) or a single refusal direction being suppressed (Arditi et al., 2024; Winninger et al., 2025). However, isolating a compact circuit that captures the causal mechanism of these failures remains an open challenge. Besides being compact, such a circuit must be both *faithful*, reproducing the attack-induced behavior change on its own, and *complete*, with the neurons outside it insufficient to reproduce that change. In this paper, we aim to address this gap by developing a novel algorithm that systematically reduces circuit size while preserving both faithfulness and completeness.

Circuit analysis uses causal interventions to isolate the internal mechanism responsible for the difference between contrastive inputs that differ minimally yet flip the model's output (Meng et al., 2023; Goldowsky-Dill et al., 2023; Heimersheim & Nanda, 2024). Classical analyses draw such pairs from a *template*, a fixed syntactic frame with one slot changed: in indirect object identification, "When Mary and John went to the store, John gave a drink to" is paired with a variant in which the names are swapped (Wang et al., 2022). Practical tasks such as safety classification have no such frame: prompts vary wildly in topic, length, and phrasing, so no principled minimal contrastive pair exists. Existing interpretability tools would then fall back on mean or zero ablation, replacing activations with dataset averages or zeros that the model itself never produces, which takes it out of distribution (Chan et al., 2022; Heimersheim & Nanda, 2024). To address this, we construct pairs using adversarial suffixes (Zou et al., 2023) that flip the model's verdict on a harmful request while leaving the request untouched.

The second obstacle is scaling search over a vast space. The natural units for building circuits are MLP neurons: they immediately follow a nonlinearity, so each is a basis-aligned unit whose activation can be read and intervened on individually, and they typically constitute over two-thirds of a transformer's parameters (roughly 70% in the models we study). But they are commonly dismissed as too polysemantic and too numerous to study directly (Cunningham et al., 2023; Bricken et al., 2023; Paulo et al., 2025). Sparse dictionary-learned features (Cunningham et al., 2023; Bricken et al., 2023) were developed to address this issue, but have limitations of their own since they are only an approximation to the original model (Section 2, Appendix A). We argue the bottleneck is not the choice of unit but the tool used to select among them. Standard methods rely on gradient-based scores, such as attribution patching (Nanda, 2023; Syed et al., 2023) and integrated gradients (Sundararajan et al., 2017; Marks et al., 2025), and rank each unit once, independently, without

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

considering interactive effects. Over a search space of hundreds of thousands of neurons, they yield circuits of tens of thousands of neurons, too large to read or to cheaply verify.

We propose RAMEN (**R**efinement **A**lgorithm for **M**echanistic **E**xtraction of **N**eurons), which instead iteratively refines a circuit: each round re-scores every individual neuron against the circuit it currently holds, proposes moves that jointly swap groups of neurons in and out, and accepts a move only after an intervention confirms that the resulting circuit still meets both metrics. Over diverse prompts drawn from the models' own safety training sets and attacked by adversarial suffixes, RAMEN finds that under 0.17% of a model's MLP neurons suffice for faithfulness and under 0.60% for completeness (see Section 6) on behavior typically described via attention analysis and steering directions.

Applying our algorithm across independently optimized attacks reveals that a few core MLP neurons are selected across nearly all attacks, including those that do not directly encode the safe or unsafe verdict. Inspecting them surfaces *non-robust features* in the sense of Ilyas et al. (2019): features the model genuinely relies on that lead to correct classifications for data from its training distribution, but incorrect classifications for data outside it. Unlike in the image domain, where non-robust features are statistical signals imperceptible to humans, the ones we recover are semantically legible: for instance, one tracks whether a request is about a videogame, one if the request is for a horror story, another whether the harm lands on a nonsensical target, and another whether the request is phrased as a direct imperative. We then compose these non-robust features into fixed, prompt-universal templates, and show that they raise the attack success rate against WildGuard from 8.0% to 41.0% with no optimization (see Appendix H). Against the Latent Adversarial Training (Sheshadri et al., 2025) and Circuit Breaker (Zou et al., 2024) defenses, our template boosts the attack success rate of the GCG attack Zou et al. (2023) from under 3% to 54% and 70%, respectively (see Section 7).

2 SETTING

Models. We study four safety trained models. Three are Mistral-7B, Llama-3-8B, and Gemma-7B base models trained as WildGuard guardrails (Han et al., 2024); we reserve the name WildGuard for the released Mistral guard. The fourth, Llama-3-8B-RT (Sheshadri et al., 2025), is a generative refusal trained model and the undefended version of two state-of-the-art defenses, latent adversarial training (Sheshadri et al., 2025) and circuit breakers (Zou et al., 2024), which we attack in Section 7 with the features found on it. Training and evaluation details are in Section 6.

Adversarial suffixes. Suffix based attacks append a short optimized string of tokens to a harmful instruction so as to bypass the model's safety training (Zou et al., 2023; Thompson & Sklar, 2024; Andriushchenko et al., 2025; Sadasivan et al., 2024), maximizing the model's probability of a target continuation t such as "Sure, here's how to build a bomb". They apply directly to the safe vs. harmful classification setting of guardrail models (Bassani & Sanchez, 2025; Altinisik et al., 2025; Kumar et al., 2025), where the target is a single verdict token instead. Classical circuit analyses draw their contrastive pairs from templates, as in IOI (Wang et al., 2022) or SVA (Finlayson et al., 2021), which guarantees the token level alignment that activation patching requires. Adversarial suffixes give the same alignment without a template: they flip the model's decision while leaving the prompt tokens untouched.

Contrastive pairs. Let D_{train} be the safety training set of the model under study, with prompts labelled harmful or safe, and D_H its harmful prompts that the model judges correctly. For each $x \in D_H$ we write x_{clean} for x padded with unoptimized filler tokens in the suffix positions and x_{adv} for x with a successfully optimized suffix in those same positions, giving one contrastive pair $(x_{\text{clean}}, x_{\text{adv}})$ per prompt, collected in $D^\star$. The two inputs agree token for token outside the suffix, so we can patch activations position by position. Unless stated otherwise we patch every position up to the model's generation, not including positions of generated tokens for multi-token outputs.

Unit of analysis. We choose MLP neurons as our unit of analysis, which immediately follow a nonlinearity and are therefore basis aligned, unlike residual stream dimensions or MLP outputs that form less sparse circuits (Arora et al., 2026). MLP neurons are usually dismissed as too polysemantic and dense in favor of sparse dictionary learned features such as SAEs (Cunningham et al., 2023; Bricken et al., 2023; Paulo et al., 2025), but a dictionary is an approximation of the model rather than what the model itself computes (Gurnee, 2024; Engels et al., 2025; Paulo & Belrose, 2025; Gorton & Lewis, 2025) (Appendix A). Arora et al. (2026) show that carefully selected MLP neurons form a

sparse basis on templated tasks, often outperforming SAE features, and we find the same holds on ours given a sufficiently sophisticated search (Section 5).

Per-prompt circuits. A circuit is a computational subgraph of a neural network that explains a specific behavior. For simplicity, we focus on node level circuits and leave edge level circuits to future work. We search for a separate circuit O_i for each contrastive pair and aggregate afterwards. Searching per prompt makes recurrence informative: each search is under its own pressure to discard redundant neurons, so a neuron that survives many searches has been separately confirmed by each prompt. We treat such recurring neurons as generalizable, non-redundant mechanisms, which we exploit in Section 7 and Appendix H.

3 Circuit Criteria

Activation patching. Let M denote the model and let O be a subset of its internal components, potentially of size one. To measure the causal importance of O for a specific behavior, it is standard (Meng et al., 2023; Goldowsky-Dill et al., 2023; Heimersheim & Nanda, 2024) to compare the model's behavior on a *contrastive pair* of inputs: two inputs that agree everywhere except for a minimal difference that changes the model's output. Writing these as a *base* input x_{base} and a *patch* input x_{patch}, one runs M on x_{base} while replacing the output of every unit in O with the value it takes on x_{patch}, letting the intervention propagate through all downstream layers. We write this replacement as $\text{do}(O = O_{\text{patch}})$, where O_{patch} denotes O's activations on the patch input. Let $m(x)$ be any scalar metric of the model's output on input x. The *indirect effect* IE of patching O is the resulting change in this metric:

$$\text{IE}(O, m, x_{\text{base}}, x_{\text{patch}}) = m\big(x_{\text{base}} \mid \text{do}(O = O_{\text{patch}})\big) - m(x_{\text{base}}). \tag{1}$$

Margin metrics. We instantiate the scalar metric m differently for the two kinds of model we study. For the guardrails, which read out a verdict at a single decision token, we take the classifier margin $m(x) = M_{\text{unsafe}}(x) - M_{\text{safe}}(x)$, so that $m(x) > 0$ is exactly the condition that the model judges x harmful. For generation-based models, whose refusal or compliance spans multiple tokens, we take a hinged logit margin summed over the positions of a target completion, which is zero exactly when the target is the model's greedy generation and which a successful attack lowers just as it lowers the classifier margin (Appendix A).

Circuit metrics. We measure the quality of a circuit by two metrics adapted from the standard faithfulness and completeness of Marks et al. (2025) and Arora et al. (2026). *Faithfulness* measures how much of the full model's effect the circuit reproduces when it alone is patched, $\text{Faith}(O, m, x_{\text{base}}, x_{\text{patch}}) = \text{IE}(O, m, x_{\text{base}}, x_{\text{patch}})/\text{IE}(M, m, x_{\text{base}}, x_{\text{patch}})$, where M is the full model as in Section 3. *Completeness* measures how much of the effect the complement still reproduces when it alone is patched, $\text{Comp}(O, m, x_{\text{base}}, x_{\text{patch}}) = \text{IE}(M \setminus O, m, x_{\text{base}}, x_{\text{patch}})/\text{IE}(M, m, x_{\text{base}}, x_{\text{patch}})$. Patching the full model M recovers the full margin difference: $\text{IE}(M, m, x_{\text{base}}, x_{\text{patch}}) = m(x_{\text{patch}}) - m(x_{\text{base}})$. Ideally, a circuit attains faithfulness near 1 and completeness near 0. Faithfulness does not imply completeness: even if a circuit reproduces the full model's effect, there may be other circuits outside it that can also reproduce the effect. Unlike prior work, we compute both metrics per prompt and intervene with interchange interventions rather than mean activations (Appendix A). Throughout we fix $(x_{\text{base}}, x_{\text{patch}}) = (x_{\text{clean}}, x_{\text{adv}})$: the model runs on the unattacked prompt and O is set to the values it takes under attack, so $\text{Faith}(O)$ measures how much of the attack O alone rebuilds and $\text{Comp}(O)$ how much the neurons outside O rebuild it. This is the *sufficiency* direction, which is far harder than the *necessity* direction that standard tools already meet (Appendix A).

The search objective. To find circuits we now treat both metrics faithfulness and completeness as constraints. The search runs once per contrastive pair, so the metric m and the pair are fixed throughout, and we abbreviate $\text{Faith}(O, m, x_{\text{base}}, x_{\text{patch}})$ and $\text{Comp}(O, m, x_{\text{base}}, x_{\text{patch}})$ to $\text{Faith}(O)$ and $\text{Comp}(O)$. Fixing thresholds τ_{Faith} and τ_{Comp}, we call a circuit O *faithful* if $\text{Faith}(O) \geq \tau_{\text{Faith}}$, *complete* if $\text{Comp}(O) \leq \tau_{\text{Comp}}$, and *feasible* if it is both, and seek the smallest feasible circuit:

$$\min_{O \subseteq \Theta} |O| \quad \text{subject to} \quad \text{Faith}(O) \geq \tau_{\text{Faith}}, \quad \text{Comp}(O) \leq \tau_{\text{Comp}} \tag{2}$$

where Θ is the candidate pool of MLP neurons. In practice we restrict Θ to the layers below an upper bound, since an unrestricted pool yields circuits drawn from the latest layers, which encode the

decision rather than form it (Appendix A). Solving 2 exactly is a combinatorial optimization problem over subsets of $301{,}056$ neurons for the Mistral and Llama models and $491{,}520$ for Gemma, and similar circuit finding problems are NP-hard (Adolfi et al., 2025), so we approximate it.

4 Limitations of Existing Methods

Attribution patching. While activation patching yields an exact causal quantity, scoring every component requires a separate forward pass per component, which is expensive at fine granularities of units. Attribution patching (Nanda, 2023; Syed et al., 2023; Kramár et al., 2024) addresses this by replacing the causal intervention with its first-order Taylor expansion around the base activations, giving an estimate of the indirect effect for each unit of O,

$$\widehat{\mathrm{IE}}(O, m, x_{\text{base}}, x_{\text{patch}}) \;=\; \big(O_{\text{patch}} - O_{\text{base}}\big) \odot \nabla_O \, m(x_{\text{base}}) \;\in\; \mathbb{R}^{|O|}, \tag{3}$$

where O_{base} and O_{patch} are O's activations on the base and patch inputs, $\nabla_O \, m(x_{\text{base}})$ is the gradient of the metric with respect to O's output on the base input, and $\odot$ is the elementwise product. Each coordinate is one unit's estimated contribution of Equation 1. One pair of forward passes and one backward pass score all units at once.

The standard circuit pipeline orders every unit once by a gradient based score and takes the units above a threshold, equivalent to the top k for some k. Methods differ in the score, e.g. attribution patching (Nanda, 2023; Syed et al., 2023; Kramár et al., 2024), integrated gradients (Sundararajan et al., 2017; Marks et al., 2025), or gradients at linearized replacement models (Jafari et al., 2025; Arora et al., 2026; Lindsey et al., 2025), in how thresholds are chosen, and in the choice of units, but follow the same two step pipeline. The exception is ACDC (Conmy et al., 2023), which starts from the full circuit and iteratively prunes units without gradients, but is intractable above the scale of GPT2; it has been combined with gradient scoring by pruning only the units above a threshold (Syed et al., 2023). Against the faithfulness and completeness objectives of Equation 2, existing baselines do not reach a small feasible circuit at any k we check (Table 1). We give three reasons why existing methods do not perform well at the complexity and granularity of our task, which then motivate our algorithmic design.

I. Gradient based scores do not account for interactions between units. The gradient in Eq. 3 is taken at $m(x_{\text{clean}})$, the unpatched clean pass, so each coordinate of $\widehat{\mathrm{IE}}(O, m, x_{\text{clean}}, x_{\text{adv}})$ estimates what one unit of O contributes when none of the others are patched. The score a unit of O receives is the same no matter which other units are also in O. We refer to this below as scoring over the empty set, and it is the default for gradient based methods.

II. Per-unit search is intractable for neurons. Iterative pruning such as ACDC tests one unit at a time by patching it and measuring the change in the metric, which is intractable at the scale of tens of thousands of neurons, each requiring its own forward pass.

III. Prior work does not score against both circuit metrics. Prior gradient based methods rank units only by their contribution to faithfulness, even though circuits are also judged on completeness. In our setting, completeness, not faithfulness, is what limits how small a feasible circuit can be. Removing a unit from a circuit also adds it to the complement, so a unit that appears unimportant for faithfulness because others in the circuit redundantly carry the same information can, once moved to the complement, raise the complement's faithfulness, i.e. lower completeness.

Challenges I–III arise whenever a moderately distributed behavior is studied at neuron granularity, and prior work has largely avoided them by construction. Most methods treat a layer's entire MLP block as a single unit (Hanna et al., 2024; Syed et al., 2023) or substitute dictionary learned features for neurons (Marks et al., 2025), which sidesteps unreliable per-neuron gradients (I) and the cost of per-neuron search (II). Arora et al. (2026) works at neuron granularity, but on classical tasks whose mechanisms are sparse enough that a small circuit found by faithfulness gradients is complete without selecting for completeness (III). Safety and refusal mechanisms are more distributed and redundant, so completeness does not follow from faithfulness and must be optimized for directly.

Design principles. Limitations I–III each motivate the design of our search in the following section.

(i) The circuit should be formed iteratively over multiple rounds, and each round should recompute the gradient based ordering over the current circuit, instead of relying on a single fixed ordering over the empty set (I).
(ii) Since patching single units is intractable, each round should consider multiple neurons (II).
(iii) An update should add neurons as well as remove them, so that removals that increase too much Comp(O) can be made up (III).
(iv) The orderings of estimated effects should not come from gradients for faithfulness alone. They should consider the gradient of Comp (III) and also reflect the effect of neurons in combination rather than alone (I, II).

5 RAMEN

We propose a novel algorithm RAMEN (Algorithm 1) based on the design principles (i)–(iv). It runs once per contrastive pair of $D^\star$, against which every metric below is measured. It starts from a large circuit that is already feasible, meaning it meets both thresholds of Equation 2, and shrinks it over rounds. Each round re-scores the neurons against the current circuit, proposes candidate circuits that each swap a block of neurons out for a smaller block brought in, and chooses the smallest candidate that intervention confirms is feasible.

The design is based on a separation of the roles of gradients and causal interventions. Gradient based scores are cheap, but only approximate the true IE, and can be unreliable at the neuron granularity. Interventions are exact but too expensive to run once per neuron (II). RAMEN therefore uses gradients only to *propose* candidates and interventions only to *accept* them. No candidate is chosen over another candidate on the strength of its gradient scores, so a poor ordering costs time but never yields an infeasible circuit.

Seeding (lines 1–2). The search shrinks a feasible circuit rather than growing one from the empty set. SEED obtains the starting circuit from the standard pipeline. It ranks the whole pool Θ once by attribution over the empty set, in descending order. We call the set of the first k entries of a ranking r its prefix $r[:k]$, exactly the top-k circuit of the standard pipeline. SEED measures prefixes at a fixed ladder of lengths $\mathcal{L}$ and returns the shortest feasible one (Appendix D). If no prefix on the ladder is feasible, SEED returns the prefix of minimal violation (defined in Appendix C), and FEAS (Algorithm 2) repairs it into a feasible circuit before shrinking begins.

Algorithm 1 RAMEN

Require: model M; contrastive pair $(x_{\text{base}}, x_{\text{patch}}) \in D^\star$; pool Θ; thresholds $\tau_{\text{Faith}}, \tau_{\text{Comp}}$; ordering families $\mathcal{R}, \mathcal{A}$; sizes $\mathcal{D}, \mathcal{Q}$; seed ladder $\mathcal{L}$; patience P; round cap T

1: $O \leftarrow \textsc{Seed}(M, x_{\text{base}}, x_{\text{patch}}, \Theta, \mathcal{L})$
2: **if** O is not feasible **then** $O \leftarrow \textsc{Feas}(M, x_{\text{base}}, x_{\text{patch}}, O)$ ▷ Algorithm 2
3: *fails* $\leftarrow 0$
4: **repeat**
5: build $\mathcal{R}(O)$ and $\mathcal{A}(\Theta \setminus O)$ from $\widehat{g}_O$ and $\widehat{g}_{\Theta \setminus O}$
6: Candidates $\leftarrow \{ (O \setminus \rho[:d]) \cup \alpha[:a] \ : \ \rho \in \mathcal{R}(O),\ \alpha \in \mathcal{A}(\Theta \setminus O),\ d \in \mathcal{D}(|O|),\ a \in \mathcal{Q},\ a < d \}$
7: $O^* \leftarrow$ the smallest feasible $O' \in$ Candidates, or **none**
8: **if** $O^* \neq$ **none then**
9: $O \leftarrow O^*$; *fails* $\leftarrow 0$
10: **else**
11: *fails* $\leftarrow$ *fails* $+ 1$
12: **end if**
13: **until** *fails* $= P$ or T rounds have run
14: **return** O

Orderings (line 5). Each round begins by ranking neurons by their estimated effect on faithfulness and completeness if moved in or out of the circuit. A *removal ordering* ρ sorts the members of O in ascending order of estimated effect if removed, so the neurons that appear cheapest to remove come first. An *addition ordering* α sorts the non-members $\Theta \setminus O$ in descending order of estimated effect if added, so the neurons that appear most useful to bring in come first. Following principle (i), every

ordering is recomputed each round at the current circuit, so a neuron's rank reflects its estimated importance given the other neurons presently in O.

A round uses several orderings of each kind, collected in the families $\mathcal{R}(O)$ and $\mathcal{A}(\Theta \setminus O)$, which following principle (iv) differ in whether they use the gradient of Faith or of Comp and in how they estimate the effect of neurons in combination and no single ordering is best at every stage of the search .

Candidates and moves (lines 6–7). The round then assembles a set of candidate circuits. A *move* removes the first d entries of a removal ordering ρ and adds the first a entries of an addition ordering α, giving the candidate $(O \setminus \rho[:d]) \cup \alpha[:a]$. The round forms one candidate per combination in line 6, with $a < d$ so that every candidate is strictly smaller than O.

The block is a central design choice that satisfies (ii): one intervention tests d neurons at once, so cost scales with the number of candidates and not with $|\Theta|$. Taking blocks as prefixes is a greedy choice, as they are what the ordering currently rates cheapest to remove or most useful to add. Blocks are dynamic in both content and size: which neurons leave or enter depends on everything else in O rather than on a score fixed at the start, and $\mathcal{D}(|O|)$ scales with the circuit, so early rounds remove large blocks and later rounds small ones. Pairing each removal with a smaller addition follows principle (iii): the removed block enters the complement and can break completeness, while the added block leaves it, so a removal that is infeasible on its own can succeed as a swap.

Acceptance and termination (lines 7–12). Every candidate is measured under intervention, and the round greedily accepts the smallest one that is feasible (line 7). Because the seed is feasible and each accepted candidate is verified before it replaces O, the search holds a feasible circuit after every round and can be stopped at any point with a valid result. If no candidate is feasible, O is unchanged and the round counts as a failure. The search returns O only after P consecutive failed rounds, or after T rounds in total.

5.1 Orderings

We use two main removal orderings: $\rho_{\max}$ and ρ_{ctx}.

Attribution at a circuit. Equation 3 scores every unit against the unpatched base run. The search instead needs scores taken against the circuit it currently holds, which we get by modifying the metric. For $S \subseteq \Theta$, write $m_{S_{\text{patch}}}(\cdot) := m\big(\cdot \mid \text{do}(S = S_{\text{patch}})\big)$ for the metric of a forward pass in which every unit of S is patched, following the subscript convention for counterfactuals (Pearl, 2013). So $m_{\emptyset_{\text{patch}}} = m$, and Faith$(O)$ and Comp(O) are the normalized $m_{O_{\text{patch}}}(x_{\text{base}})$ and $m_{(\Theta\setminus O)_{\text{patch}}}(x_{\text{base}})$. Similarly, we write $a_{i,S_{\text{patch}}}(\cdot) := a_i\big(\cdot \mid \text{do}(S = S_{\text{patch}})\big)$ for unit i's activation on that same intervened run, so that $a_{i,\emptyset_{\text{patch}}} = a_i$.

Substituting this metric into Equation 3 scores a single unit i at the forward pass where S is patched. Two degrees of freedom remain: the value i is scored towards, and the value it is scored from. The move the search makes fixes both. A unit outside S is only ever *patched*, so it is scored towards $a_i(x_{\text{patch}})$ from $a_{i,S_{\text{patch}}}(x_{\text{base}})$. A unit inside S is only ever *released*: taking the value it would have been if it was not patched, so it is scored towards $a_{i,(S\setminus\{i\})_{\text{patch}}}(x_{\text{base}})$ from $a_i(x_{\text{patch}})$. Note that this is **not** the same as patching i to base, because patching $S \setminus \{i\}$ also causes downstream effects that move the activation at i. Writing $S_{-i} := S \setminus \{i\}$, so that $S_{-i} = S$ for $i \notin S$, one expression covers both cases:

$$\widehat{g}_S(i) := \big(a_i(x_{\text{patch}}) - a_{i,(S_{-i})_{\text{patch}}}(x_{\text{base}})\big)\, \frac{\partial\, m_{S_{\text{patch}}}(x_{\text{base}})}{\partial\, a_{i,S_{\text{patch}}}(x_{\text{base}})}.$$

For $i \notin S$ this is the first-order effect of patching i and for $i \in S$ it is minus the first-order effect of releasing it, so in both cases $\widehat{g}_S(i) > 0$ means that i in its patched state raises $m_{S_{\text{patch}}}$. At $S = \emptyset$ it is coordinate i of Equation 3.

$\rho_{\max}$: remove what both metrics rate low. Removing a member i from O changes not just Faith(O) but also Comp(O) (III). In the world of Faith(O), where O is patched, removal releases i; in the world of Comp(O), where the complement $\Theta \setminus O$ is patched, removal sends i to the complement, where it is patched. Both are the moves $\widehat{g}$ scores, so for $i \in O$,

$$g^F(i) = \widehat{g}_O(i), \qquad g^C(i) = \widehat{g}_{\Theta\setminus O}(i).$$

$g^F(i)$ estimates the faithfulness lost by releasing i, and $g^C(i)$ the faithfulness gained by the complement if i joined it. Both are costs of removing i, positive when removal hurts. We write $\tilde{x}(i) = x(i)/\operatorname{sd}_{j\in O} x(j)$ for a score divided by its standard deviation over O, and the ordering is $\rho_{\max}(i) = \max\big(\tilde{g}^F(i),\, \tilde{g}^C(i)\big)$, sorted ascending with ties in the order of the unit index, so a member is proposed early only if both metrics rate it cheap to remove.

ρ_{ctx}: remove neurons rated unimportant over multiple contexts. Both scores in $\rho_{\max}$ are estimated at a single context: $g^C(i)$ estimates the effect of moving i to the complement while every other member of O stays unpatched. A move, however, sends a block of neurons to the complement at once, and a member whose effect depends on other members can misleadingly score low at $\Theta \setminus O$. ρ_{ctx} therefore estimates each member's score at additional contexts in which part of O has already moved.

Fix a set Φ of small slice fractions (Appendix D). At the start of each round, for each $\varphi \in \Phi$ draw a subset $R_\varphi \subset O$ of size $\max(1, \lfloor \varphi |O| \rfloor)$ uniformly without replacement, independently across φ and across rounds, and let $S_\varphi = (\Theta \setminus O) \cup R_\varphi$ be the complement with R_φ added. In the run $m_{(S_\varphi)_{\text{patch}}}(x_{\text{base}})$, the complement and the slice are patched and the remaining members $O \setminus R_\varphi$ are unpatched. The in-context score of a member is the magnitude of its estimate at that run, so for $i \in O$,

$$g^C_\varphi(i) = \big|\widehat{g}_{S_\varphi}(i)\big|,$$

which for $i \in O \setminus R_\varphi$ is the size of the effect of moving i to the complement given that the slice has gone, and for $i \in R_\varphi$ the size of the effect of releasing it back. For $\varphi = 0$ it reduces to $|g^C(i)|$. We take the mean over contexts, $\bar{g}^C_\Phi(i) = \frac{1}{|\Phi|}\sum_{\varphi\in\Phi} g^C_\varphi(i)$, and the ordering is $\rho_{\text{ctx}}(i) = \widetilde{|g^C|}(i) + \tilde{\bar{g}}^C_\Phi(i)$ for $i \in O$, sorted ascending. A member is placed early only if its estimated effect on the complement is small both at $\Theta \setminus O$ and at contexts where part of O has moved. Unlike $\rho_{\max}$, the ordering ρ_{ctx} is nondeterministic through the draws of R_φ, which are redrawn each round so that an interaction missed in one round can be caught in a later one.

Faithful only circuits. When only faithfulness is constrained, $\rho_{\max}$ keeps only the faithfulness score, $\rho_{\max}(i) = \tilde{g}^F(i)$. ρ_{ctx} takes its contexts on the faithfulness side instead: each context releases a random slice R_φ of O, patching only $O \setminus R_\varphi$, and scores each member by $|\widehat{g}_{O\setminus R_\varphi}(i)|$, the size of its effect on faithfulness once part of the circuit has already been removed. Similarly the ordering is $\widetilde{|g^F|}(i)$ plus the standardized mean of these scores over contexts. Unlike in the faithful and complete search, ρ_{ctx} is used only while shrinking; the stage that first reaches faithfulness (Appendix C) uses $\rho_{\max}$ alone.

6 Results

Setup. The three guardrails are supervised fine-tuned (Wang et al., 2023) on WildGuardTrain (Han et al., 2024) from the Mistral-7B, Llama-3-8B, and Gemma-7B base models; Mistral is the released WildGuard model, the most adversarially robust open-source safety guardrail under independent evaluation (Kang et al., 2026). Llama-3-8B-RT is Llama-3-8B-Instruct refusal trained with LoRA (Hu et al., 2021) on the LLM-LAT training set (Sheshadri et al., 2025). We attack with Greedy Coordinate Gradient (GCG) (Zou et al., 2023), restricted to 20 tokens by standard convention and run with the settings of Liu et al. (2024a) for the guardrails and Jia et al. (2024) for the Llama-RT model. The contrastive pairs $D^\star$ consist of 100 successful attacks on WildGuardTrain prompts for each guardrail and 85 on LLM-LAT training prompts for Llama-RT. For evaluation, against the guardrails we use the cyberattack category of GuardSet-X (Kang et al., 2026), a high severity category on which WildGuard already performs well, with a reported 8.0% ASR on the GuardSet-X attack; against Llama-RT and the LAT, RR defenses we use the AdvBench prompts curated by AdvPrefix (Zhu et al., 2025). We report circuit sizes across the three WildGuard trained guard models and the refusal trained Llama-RT in Table 1. We search two circuits per attacked prompt, one under

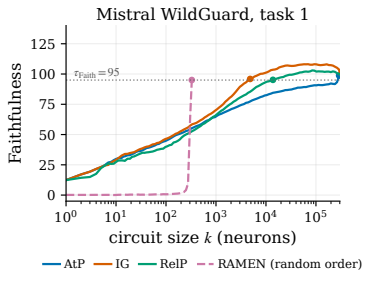


Figure 1: Faithfulness of the top k neurons of each ranking on one Mistral prompt; Appendix G shows 20 per model.

the faithfulness constraint alone ($\tau_{\text{Faith}} = 0.95$, $\tau_{\text{Comp}} = \infty$), which we label F, and one under both ($\tau_{\text{Faith}} = 0.95$, $\tau_{\text{Comp}} = 0.10$), labelled F+C.

RAMEN finds smaller circuits than every gradient based baseline. Table 1 compares median and mean sizes against IG, and Table 3 against AtP and RelP, over all 100 prompts per guardrail and 85 for Llama-RT. Our circuits are smaller on every model and both searches, and RAMEN meets the constraints on every prompt. Under the faithfulness constraint alone, our median circuits range from 159 to 365 neurons, 12.2 to 37.4 times smaller than the best of the three baselines on each model. Under both faithfulness and completeness, they range from 717 to 2,674 neurons, 5.4 to 15.5 times smaller than the best baseline. Appendix F removes each component of the search on Mistral and demonstrates that each contributes to large reductions in circuit size. Figure 1 shows the gap on one Mistral prompt: every gradient ranking needs thousands of neurons to reach τ_{Faith}, while RAMEN's faithful circuit reaches it within a few hundred, but only once nearly all of it is present. This sharp rise is what an AND-like mechanism predicts: rebuilding the attack requires suppressing every harm detecting feature the model represents at once, so faithfulness stays near zero until the last of them is patched (Appendix B). The same sharp spike arises across different prompts and on Llama-RT (Appendix G).

Table 1: Per-prompt circuit sizes in MLP neurons over all 100 prompts per guardrail and 85 for Llama-RT, against an integrated-gradients (IG) baseline that ranks the pool once and takes the smallest prefix meeting the constraints. Parentheses give each size as a share of the model's MLP neurons. Appendix E compares against AtP, RelP, and DiscoGP.

		IG baseline		Ours	
Model	Circuit	Median	Mean	Median	Mean
WildGuard trained (Han et al., 2024)					
Mistral-7B	F	4,448 (0.97%)	4,521 (0.99%)	**365** (0.08%)	**765** (0.17%)
	F+C	14,550 (3.17%)	87,961 (19.17%)	**2,674** (0.58%)	**2,737** (0.60%)
Llama-8B	F	3,934 (0.86%)	4,939 (1.08%)	**163** (0.04%)	**210** (0.05%)
	F+C	4,480 (0.98%)	8,393 (1.83%)	**837** (0.18%)	**1,027** (0.22%)
Gemma-7B	F	7,244 (1.05%)	8,792 (1.28%)	**159** (0.02%)	**338** (0.05%)
	F+C	10,390 (1.51%)	12,822 (1.86%)	**1,840** (0.27%)	**1,910** (0.28%)
Refusal trained (Sheshadri et al., 2025)					
Llama-RT	F	11,137 (2.43%)	39,965 (8.71%)	**298** (0.06%)	**373** (0.08%)
	F+C	11,093 (2.42%)	39,342 (8.58%)	**717** (0.16%)	**922** (0.20%)

7 Non-Robust Shortcuts Persist over Circuit Breakers and LAT

Figure 2 ranks neurons by how many of the independently found per-prompt circuits contain them. We observe that recurring neurons are high but drop off quickly for three of the plotted settings, which motivates inspecting the top few. Fully processing these circuits, and how they fit with components other than MLPs, requires further work, but the most frequent neurons already have practical value when interpreted: their top activating examples show semantically coherent themes that can be used to create attacks. Appendix H does this for the three WildGuard guardrails, where levers built from the most frequent neurons raise ASR from 8.0% to 41.0% with no optimization and here we ask whether the same features transfer to stronger defenses.

We find that the features found over a less defended model can be used to successfully attack strong models that are robust to GCG itself. We test this against latent adversarial training (LAT) (Sheshadri et al., 2025) and circuit breakers (RR) (Zou et al., 2024), two defenses built on Llama-3-8B that are far more robust than standard fine-tuning: GCG reaches only 0.9% and 2.5% reported ASR respectively. Across the ten most recurring MLP neurons of Llama-3-8B-RT, we find neurons that respond to cooking, horror writing requests, and poetry. We assemble the corresponding levers into a single template: a static preamble of poetry and recipe snippets, a transition into a ghostwriting instruction, 20 GCG-optimized tokens, and an LLM rewrite of the harmful prompt into a horror creative-writing frame. Appendix I describes the template; we withhold the exact strings and neuron identities, since together they form a working jailbreak.

Method	LAT	RR (circuit breakers)
AutoDAN (Liu et al., 2024b)	0.0 (0.0–7.1)	0.0 (0.0–7.1)
AGILE (Wang et al., 2026)	12.0 (4.5–24.3)	16.0 (7.2–29.1)
In-context prompt + random search (Andriushchenko et al., 2025)	0.0 (0.0–7.1)	0.0 (0.0–7.1)
In-context prompt + I-GCG	0.0 (0.0–7.1)	0.0 (0.0–7.1)
Semantic levers + I-GCG (ours)	**54.0** (39.3–68.2)	**70.0** (55.4–82.1)

Table 2: ASR (%) against LAT and RR (circuit breakers) on n=50 AdvBench prompts curated by AdvPrefix, scored by the HarmBench judge. Parentheses are 95% Clopper–Pearson exact confidence intervals.

Table 2 compares against three baselines chosen to rule out that the defenses are simply weak, that any long benign preamble breaks them, or that any jailbreak combined with GCG breaks them: AutoDAN; AGILE, which combines a multi-turn benign preamble with LLM rewrites; and the in-context prompts attack of Andriushchenko et al. (2025), which reaches 100% ASR on the adversarially defended R2D2 (Mazeika et al., 2024), run with its original random search optimizer and again with I-GCG substituted in to rule out the optimizer explaining the difference.

The frequency curves from RAMEN are what motivated us to inspect these neurons, though we do not claim they are the only way to find these features: other white box or black box methods may find them as well.

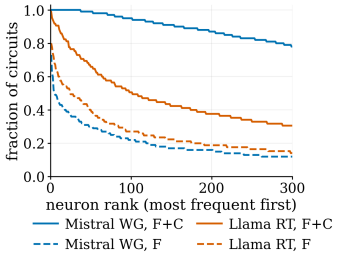


Figure 2: Fraction of per-prompt circuits containing each neuron.

8 Related Work

Mechanistic interpretability of adversarial suffixes. Existing mechanistic work on adversarial suffixes primarily explains attack success through two parts: attention hijacking, where the attack causes the model to suppress attention to harmful content (Wang et al., 2024; Ben-Tov et al., 2025; Yin et al., 2026) and the suppression of a refusal direction in the residual stream (Arditi et al., 2024; Winninger et al., 2025; Ball et al., 2024). Zhao et al. (2026) finds that harmfulness and refusal directions are encoded separately.

Interpretable adversarial attacks. More broadly, in the image domain, interpretable adversarial attacks, in contrast to PGD (Madry et al., 2019), have been constructed by optimizing patches to produce human-describable features (Casper et al., 2023) or by using interpretability tools to identify spurious feature-class associations that guide the design of copy/paste attacks using natural images (Carter et al., 2019).

9 Discussion and Limitations

We discover that GCG circuits are AND-like: there exist sparse, faithful, and complete mechanisms that appear only once every component is in place (Figure 1). A ranking scored once at the empty circuit underweighs interactive effects, which is why existing baselines miss it and can mislead practitioners into believing that sparse mechanisms are diffuse. Since GCG can elicit a wide variety of behaviors (Geiping et al., 2024), our framework is not specific to safety classification, and we are excited for future work that tests it on other tasks more complex than the templated tasks circuit level interpretability typically operates on.

Our framework needs successful adversarial suffixes to supply contrastive pairs, and GCG is expensive to run at scale. More efficient discrete attacks, or continuous relaxations, would be valuable. Our levers show that the features read off recurring neurons are real vulnerabilities, but not that GCG uses the same concepts. The neurons are polysemantic, and a suffix may or may not drive them through a different meaning than the one we surface.

AI DISCLOSURE

In this work, we used generative AI tools for writing assistance, including editing, and restructuring text, and for preparing the LaTeX source, tables, and figure layout.

LLM-generated code was verified and tested for correctness.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Federico Adolfi, Martina G. Vilas, and Todd Wareham. The computational complexity of circuit discovery for inner interpretability, 2025. URL https://arxiv.org/abs/2410.08025.

Enes Altinisik, Safa Messaoud, Husrev Taha Sencar, Hassan Sajjad, and Sanjay Chawla. Explaining the role of intrinsic dimensionality in adversarial training, 2025. URL https://arxiv.org/abs/2405.17130.

Maksym Andriushchenko and Nicolas Flammarion. Does refusal training in llms generalize to the past tense?, 2025. URL https://arxiv.org/abs/2407.11969.

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks, 2025. URL https://arxiv.org/abs/2404.02151.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction, 2024. URL https://arxiv.org/abs/2406.11717.

Aryaman Arora, Zhengxuan Wu, Jacob Steinhardt, and Sarah Schwettmann. Language model circuits are sparse in the neuron basis, 2026. URL https://arxiv.org/abs/2601.22594.

Sarah Ball, Frauke Kreuter, and Nina Panickssery. Understanding jailbreak success: A study of latent space dynamics in large language models, 2024. URL https://arxiv.org/abs/2406.09289.

Elias Bassani and Ignacio Sanchez. On guardrail models' robustness to mutations and adversarial attacks. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 16995–17006, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.922. URL https://aclanthology.org/2025.findings-emnlp.922/.

Matan Ben-Tov, Mor Geva, and Mahmood Sharif. Universal jailbreak suffixes are strong attention hijackers, 2025. URL https://arxiv.org/abs/2506.12880.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 2019. doi: 10.23915/distill.00015. https://distill.pub/2019/activation-atlas.

Stephen Casper, Max Nadeau, Dylan Hadfield-Menell, and Gabriel Kreiman. Robust feature-level adversaries are interpretability tools, 2023. URL https://arxiv.org/abs/2110.03605.

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldwosky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing, a method for rigorously testing interpretability hypotheses. *AI Alignment Forum*, 2022. https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability, 2023. URL https://arxiv.org/abs/2304.14997.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL https://arxiv.org/abs/2309.08600.

Joshua Engels, Logan Riggs, and Max Tegmark. Decomposing the dark matter of sparse autoencoders, 2025. URL https://arxiv.org/abs/2410.14670.

Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. Causal analysis of syntactic agreement mechanisms in neural language models, 2021. URL https://arxiv.org/abs/2106.06087.

Jonas Geiping, Alex Stein, Manli Shu, Khalid Saifullah, Yuxin Wen, and Tom Goldstein. Coercing llms to do and reveal (almost) anything, 2024. URL https://arxiv.org/abs/2402.14020.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching, 2023. URL https://arxiv.org/abs/2304.05969.

Liv Gorton and Owen Lewis. Adversarial examples are not bugs, they are superposition, 2025. URL https://arxiv.org/abs/2508.17456.

Wes Gurnee. SAE Reconstruction Errors Are (Empirically) Pathological. AI Alignment Forum, Mar 2024. URL https://www.alignmentforum.org/posts/rZPiuFxESMxCDHe4B/sae-reconstruction-errors-are-empirically-pathological. Accessed: 2026-09-06.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms, 2024. URL https://arxiv.org/abs/2406.18495.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms, 2024. URL https://arxiv.org/abs/2403.17806.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching, 2024. URL https://arxiv.org/abs/2404.15255.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features, 2019. URL https://arxiv.org/abs/1905.02175.

Farnoush Rezaei Jafari, Oliver Eberle, Ashkan Khakzar, and Neel Nanda. Relp: Faithful and efficient circuit discovery in language models via relevance patching, 2025. URL https://arxiv.org/abs/2508.21258.

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, 2024. URL https://arxiv.org/abs/2405.21018.

Mintong Kang, Zhaorun Chen, Chejian Xu, Jiawei Zhang, Chengquan Guo, Minzhou Pan, Ivan Revilla, Yu Sun, and Bo Li. Poly-guard: Massive multi-domain safety policy-grounded guardrail dataset, 2026. URL https://arxiv.org/abs/2506.19054.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. Atp*: An efficient and scalable method for localizing llm behaviour to components, 2024. URL https://arxiv.org/abs/2403.00745.

Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Aaron Jiaxun Li, Soheil Feizi, and Himabindu Lakkaraju. Certifying llm safety against adversarial prompting, 2025. URL https://arxiv.org/abs/2309.02705.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Richard Liu, Steve Li, and Leonard Tang. Accelerated coordinate gradient. https://web.archive.org/web/2024/https://blog.haizelabs.com/posts/acg/, 2024a. Blog post, archived.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models, 2024b. URL https://arxiv.org/abs/2310.04451.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019. URL https://arxiv.org/abs/1706.06083.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models, 2025. URL https://arxiv.org/abs/2403.19647.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal, 2024. URL https://arxiv.org/abs/2402.04249.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023. URL https://arxiv.org/abs/2202.05262.

Neel Nanda. Attribution patching: Activation patching at industrial scale. https://www.neelnanda.io/mechanistic-interpretability/attribution-patching, February 2023.

Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features, 2025. URL https://arxiv.org/abs/2501.16615.

Gonçalo Paulo, Stepan Shabalin, and Nora Belrose. Transcoders beat sparse autoencoders for interpretability, 2025. URL https://arxiv.org/abs/2501.18823.

Judea Pearl. Direct and indirect effects, 2013. URL https://arxiv.org/abs/1301.2300.

Vinu Sankar Sadasivan, Shoumik Saha, Gaurang Sriramanan, Priyatham Kattakinda, Atoosa Chegini, and Soheil Feizi. Fast adversarial attacks on language models in one gpu minute, 2024. URL https://arxiv.org/abs/2402.15570.

Abhay Sheshadri, Aidan Ewart, Phillip Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, and Stephen Casper. Latent adversarial training improves robustness to persistent harmful behaviors in llms, 2025. URL https://arxiv.org/abs/2407.15549.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017. URL https://arxiv.org/abs/1703.01365.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery, 2023. URL https://arxiv.org/abs/2310.10348.

T. Ben Thompson and Michael Sklar. Flrt: Fluent student-teacher redteaming, 2024. URL https://arxiv.org/abs/2407.17447.

Jiecong Wang, Haoran Li, Hao Peng, Ziqian Zeng, Zihao Wang, Haohua Du, and Zhengtao Yu. Activation-guided local editing for jailbreaking attacks, 2026. URL https://arxiv.org/abs/2508.00555.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022. URL https://arxiv.org/abs/2211.00593.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. How far can camels go? exploring the state of instruction tuning on open resources, 2023.

Zijun Wang, Haoqin Tu, Jieru Mei, Bingchen Zhao, Yisen Wang, and Cihang Xie. Attngcg: Enhancing jailbreaking attacks on llms with attention manipulation, 2024. URL https://arxiv.org/abs/2410.09040.

Thomas Winninger, Boussad Addad, and Katarzyna Kapusta. Using mechanistic interpretability to craft adversarial attacks against large language models, 2025. URL https://arxiv.org/abs/2503.06269.

Yanchen Yin, Dongqi Han, and Linghui Li. Robust harmful features under jailbreak attacks: Mechanistic evidence from attention head specialization in large language models, 2026. URL https://arxiv.org/abs/2606.28153.

Lei Yu, Jingcheng Niu, Zining Zhu, Xi Chen, and Gerald Penn. Sheaf discovery with joint computation graph pruning and flexible granularity, 2025. URL https://arxiv.org/abs/2407.03779.

Jiachen Zhao, Jing Huang, Zhengxuan Wu, David Bau, and Weiyan Shi. Llms encode harmfulness and refusal separately, 2026. URL https://arxiv.org/abs/2507.11878.

Sicheng Zhu, Brandon Amos, Yuandong Tian, Chuan Guo, and Ivan Evtimov. Advprefix: An objective for nuanced llm jailbreaks, 2025. URL https://arxiv.org/abs/2412.10321.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. URL https://arxiv.org/abs/2307.15043.

Andy Zou, Long Phan, Justin Wang, Derek Duenas, Maxwell Lin, Maksym Andriushchenko, Rowan Wang, Zico Kolter, Matt Fredrikson, and Dan Hendrycks. Improving alignment and robustness with circuit breakers, 2024. URL https://arxiv.org/abs/2406.04313.

A Details of the circuit criteria

Margin for generative models. For generation-based models, whose refusal or compliance spans multiple tokens, the metric of Section 3 is

$$m(x) = \sum_{i=1}^{H} \max\big(0,\ \max_{v \neq t_i} z_v^{(i)} - z_{t_i}^{(i)}\big),$$

where $t = t_{1:H}$ is a target completion and $z^{(i)} \in \mathbb{R}^{|V|}$ is the model's next token logit vector at the position i. All H logit vectors come from a single forward pass with the target included, so $z^{(i)}$ is conditioned on the prefix $t_{1:i-1}$ rather than on the model's own generated tokens. This is zero exactly when t is the greedy generation of the model, and a successful attack lowers it just as it lowers the classifier margin. We hinge each term so that an already matched position cannot be used to compensate for other positions.

Per-prompt metrics and interchange interventions. We make two minor adjustments to the definitions of faithfulness and completeness used in prior work. First, because we find a separate circuit for each prompt, we compute both metrics per prompt and report the mean and median across prompts, whereas prior work evaluates a single task-level circuit by taking the expectation over the full dataset separately in the numerator and the denominator. Second, because each circuit is evaluated on the prompt it was optimized for, a matched counterfactual input is available, so we intervene with interchange interventions rather than mean activations, which keeps the model closer to its natural distribution (Chan et al., 2022; Heimersheim & Nanda, 2024).

Patching directions. Equation 1 fixes the roles of the two inputs: the model runs on x_{base} and O is set to its values on x_{patch}. Taking $(x_{\text{base}}, x_{\text{patch}}) = (x_{\text{adv}}, x_{\text{clean}})$ restores O to its unattacked values inside an attacked forward pass, testing whether O is *necessary* for the attack, known as denoising. Taking $(x_{\text{base}}, x_{\text{patch}}) = (x_{\text{clean}}, x_{\text{adv}})$ instead inserts the attacked values of O into a clean forward pass, testing whether O is *sufficient* to rebuild the attack, known as noising. Faithfulness and completeness are defined against whichever assignment is in use; unless stated otherwise we report the sufficiency direction.

Necessity is the easier of the two and is already met by standard tools (Appendix B). Sufficiency is far harder and far more informative. Intuitively, rebuilding the attack requires suppressing every harm detecting feature the model represents at once, whereas undoing it only needs to restore a few harm detecting features. The rest of the paper is therefore in the sufficiency direction, and we fix $(x_{\text{base}}, x_{\text{patch}}) = (x_{\text{clean}}, x_{\text{adv}})$ throughout: the model runs on the unattacked prompt and O is set to the values it takes under attack. Faith(O) then measures how much of the attack that O alone rebuilds, and Comp(O) how much the neurons outside O rebuild the attack.

Restricting the pool. The pool Θ of Equation 2 is a subset of the full MLP neuron pool, obtained by upper bounding the layer in which neurons are searched over. Having Θ be the full neuron pool can cause solutions to consist only of the earliest or latest layers, as they are most effective at flipping the model's output, so constraining the pool surfaces the computation that forms the decision rather than the computation that just encodes it or replaces the input outright. We use bound layer 20 to upper bound each model, additionally skipping the first layer for Gemma as we found that a disproportionate number of neurons were chosen at the first MLP layer otherwise. The pool is therefore layers 0 to 20, $301{,}056$ neurons, for Mistral, Llama, and Llama-RT, and layers 1 to 20, $491{,}520$ neurons, for Gemma.

B Necessity results

Necessity and sufficiency differ sharply in how easily a sparse circuit can be found. In the denoising (necessity) direction, a single global pool of 256 neurons, ranked by attribution patching (AtP) and shared across all prompts, recovers $\sim 100\%$ of the attack's effect on WildGuard. In the noising (sufficiency) direction, the shortest prefix that reproduces at least $\tau_{\text{Faith}} = 0.95$ of the attack when patched into the clean run has a median length of $20{,}184$ neurons per prompt.

We attribute this asymmetry to how harm is represented. Many of the necessary neurons are harm detectors that GCG suppresses. Restoring even a few of them introduces a harm signal that, for a model trained to detect harm, easily propagates to the output and flips the decision back to HARMFUL, so detection behaves like an OR over these features. On the other hand, for sufficiency, to make a harmful prompt read as safe, the patched activations must silence every harm feature the model represents at once, closer to an AND over the same features. This also makes intuitive sense in the input space: adding harmful words to flip a safe prompt to harmful is much easier than adding safe words to flip a harmful prompt to safe.

C REACHING FEASIBILITY

Every candidate Algorithm 1 accepts must be feasible. Algorithm 2 is what line 2 calls to repair an infeasible seed when no feasible seed is initially found.

We measure how far a circuit is from satisfying feasibility by the *violation*

$$v(O) = \max\big(0,\ \tau_{\text{Faith}} - \text{Faith}(O)\big) + \max\big(0,\ \text{Comp}(O) - \tau_{\text{Comp}}\big) \tag{4}$$

the number of points O falls short on faithfulness plus the number it exceeds on completeness, each counted as zero when its threshold is met. Both terms are already on a common scale and we weigh them equally, and $v(O) = 0$ exactly when O is feasible. The violation plays no part in Equation 2 and none in Algorithm 1. Once a feasible circuit is found it is never used.

Algorithm 2 Reaching feasibility (FEAS)

Require: model M and pair $(x_{\text{base}}, x_{\text{patch}})$; a circuit O with $v(O) > 0$
Require: ordering families $\mathcal{R}, \mathcal{A}$; drop sizes $\mathcal{D}$; add sizes $\mathcal{Q}$; pure-add sizes $\mathcal{G}$; patience P; kick trigger $P_{\text{kick}} \le P$

1: *fails* $\leftarrow 0$
2: **repeat**
3: build $\mathcal{R}(O)$ and $\mathcal{A}(\Theta \setminus O)$ from $\widehat{g}_O$ and $\widehat{g}_{\Theta\setminus O}$
4: Menu $\leftarrow \big\{\ (O \setminus \rho[{:}d]) \cup \alpha[{:}a]\ :\ \rho \in \mathcal{R}(O),\ \alpha \in \mathcal{A}(\Theta \setminus O),\ d \in \mathcal{D}(|O|),\ a \in \mathcal{Q},\ a < d\ \big\} \cup \big\{\ O \cup \alpha[{:}g]\ :\ \alpha \in \mathcal{A}(\Theta \setminus O),\ g \in \mathcal{G}\ \big\}$ ▷ pure additions
5: **if** some $O' \in$ Menu has $v(O') = 0$ **then return** the smallest such O'
6: $O^* \leftarrow \arg\min_{O' \in \text{Menu}} \big(v(O'),\ |O'|\big)$
7: **if** $v(O^*) < v(O) - \varepsilon$ **then**
8: $O \leftarrow O^*$; *fails* $\leftarrow 0$
9: **else**
10: *fails* $\leftarrow$ *fails* $+ 1$
11: **if** *fails* $= P_{\text{kick}}$ and a kick remains **then** take a kick (Appendix D)
12: **end if**
13: **until** *fails* $= P$ or the round cap is reached
14: **return** O

Its rounds have the shape of Algorithm 1's and use the same orderings, $\rho_{\max}$ and ρ_{ctx} (Section 5.1).

The accepted move is the candidate of least violation over the whole menu, ties broken by size (line 6), and we accept it only if it improves on the current violation by more than a margin ε (line 7), which prevents the search from spending rounds on changes within measurement noise. The procedure returns as soon as any candidate is feasible (line 5); it is looking for a circuit to hand to Algorithm 1, not a small one, and size enters only as a tiebreak among equally violating candidates.

Nothing guarantees that Algorithm 2 reaches a feasible circuit. When it does not, Algorithm 1 resumes at line 3 with an infeasible circuit, accepts nothing, and returns it unchanged, satisfying neither constraint. In practice this does not happen: Algorithm 2 reached a feasible circuit on every prompt we ran.

D SEARCH PARAMETERS

Parameters of Algorithms 3, 2 and 1, as run on the WildGuard models. Llama-RT's differences are listed at the end.

Seeding. SEED (Algorithm 3) ranks the whole pool once by attribution patching, Equation 3, and turns that ranking into candidate circuits by taking prefixes at a fixed ladder of sizes $\mathcal{L}$.

Algorithm 3 Seeding (SEED)

Require: model M and pair $(x_{\text{base}}, x_{\text{patch}})$; pool Θ; thresholds $\tau_{\text{Faith}}, \tau_{\text{Comp}}$; size ladder $\mathcal{L}$, ending with $|\Theta|$

1: $\pi \leftarrow \Theta$ sorted by $\widehat{\text{IE}}(\Theta, m, x_{\text{base}}, x_{\text{patch}})$ descending ▷ Equation 3
2: measure Faith$(\pi[:\ell])$ and Comp$(\pi[:\ell])$ for all $\ell \in \mathcal{L}$ in one batched call
3: $F \leftarrow \{\, \ell \in \mathcal{L} \;:\; \text{Faith}(\pi[:\ell]) \geq \tau_{\text{Faith}} \text{ and } \text{Comp}(\pi[:\ell]) \leq \tau_{\text{Comp}} \,\}$
4: **if** $F \neq \emptyset$ **then return** $\pi[:\min F]$ ▷ shortest feasible prefix
5: **return** $\pi[:\ell^*]$ where $\ell^* = \arg\min_{\ell \in \mathcal{L}} \big(v(\pi[:\ell]),\ \ell\big)$ ▷ least violating, shortest among ties

The ladder is geometric, $\ell_0 = 8{,}000$ and $\ell_{k+1} = \lfloor 1.25\,\ell_k \rfloor$ while $\ell_{k+1} < |\Theta|$, followed by $|\Theta|$ itself. The lengths come from a fixed ladder rather than a binary search because feasibility is not monotone in prefix length: neurons interact and can contribute with either sign, so a longer prefix is not necessarily more faithful or more complete than a shorter one.

Line 4 returns a circuit that Algorithm 1 can shrink directly. Line 5 returns the best starting point available when no prefix is feasible, and Algorithm 2 repairs it.

Addition orderings. A non-member $j \notin O$ is scored at the two intervened runs towards being a member. The addition orderings are simply:

$$\alpha_C(j) = \widehat{g}_{\Theta \setminus O}(j), \qquad \alpha_F(j) = \widehat{g}_O(j).$$

α_C puts first the non-members the patched complement relies on most, so that moving them into O lowers Comp(O); α_F puts first those whose patching would raise Faith(O) most.

Move sizes. For the shrinking stage the drop sizes mix fractions of the current circuit size with fixed small sizes, so that early rounds can remove a large part of the seed while late rounds can still make small adjustments:

$$\mathcal{D}(n) = \big\{ \lfloor \phi n \rfloor : \phi \in \{0.4, 0.2, 0.1, 0.05\} \big\} \cup \{25, 50, 100, 200, 400\},$$

restricted to $[1, n)$, with add sizes $\mathcal{Q} = \{0, 50, 100, 300, 1000\}$; including 0 keeps a pure-removal candidate on every menu. Candidates with $a \geq d$ are excluded, so every candidate is strictly smaller than the current circuit. Above 20,000 neurons only $\rho_{\max}$ is offered, with drop fractions $\{0.5, 0.4, 0.3, 0.2\}$, which is where the accepted move always came from on these families; if two such rounds accept nothing, the run switches to the full menu instead of stopping.

Slice fractions. The contexts of ρ_{ctx} are built from the slice fractions $\Phi = \{0.1, 0.2\}$. They are kept small so that each context is a circuit the search could plausibly reach in the next round or two, rather than one far from the current O, and there are two of them because each costs a further forward and backward pass per round. In total, $\rho_{\max}$ costs two forward and two backward passes per round, and ρ_{ctx}, given the g^C that $\rho_{\max}$ already computes, $|\Phi|$ more of each.

Termination. Both stages stop after P consecutive rounds in which no candidate is accepted, or when the round cap T is reached. We use $P = 2$ throughout: a single failed round is weak evidence that the search is stuck, since the orderings are rebuilt every round and a round that accepts nothing still changes what the next round proposes, while waiting longer spends a full menu of measurements per extra round for moves that rarely pass. When the shrinking stage stops with budget left it is restarted from the circuit it reached, with fresh attributions and fresh slices, until a restart fails to shrink the circuit.

Within Algorithm 2 a *kick* fires at $P_{\text{kick}} = 2$ failed rounds: the stage accepts the candidate with the highest faithfulness regardless of its complement score, as long as faithfulness rises by more than 0.003, and the ordinary rule resumes. At most $K = 3$ kicks are taken per run.

Llama-RT. The same procedure with three differences.

Seed ladder. Geometric series is 1.3.

Feasibility stage. Runs whenever the seed is infeasible, with coarser grids since its moves are large:

$$\mathcal{D}_{\text{feas}}(n) = \big\{\lfloor \phi n \rfloor : \phi \in \{0.3, 0.15, 0.05\}\big\} \cup \{500, 1000, 2000, 4000, 8000, 16000\}, \qquad \mathcal{Q}_{\text{feas}} = \{0, 500, 2000\},$$

plus pure additions of $g \in \mathcal{G} = \{500, 2000, 8000\}$ non-members, since reaching feasibility may require growing. Capped at 30 rounds.

Move sizes. The full family of orderings and the full grid are offered at every circuit size, with fractions $\{0.3, 0.2, 0.1, 0.05\}$, fixed sizes that double from 25 to 3,200, and $\mathcal{Q} = \{0, 50, 100, 500\}$.

E BASELINE COMPARISONS

Table 1 compares our circuits against an integrated-gradients baseline. Here we add two further rankings, attribution patching (AtP) and RelP (Jafari et al., 2025; Arora et al., 2026), reported in Table 3, describe how all the ranking baselines were run, and report a learned-mask baseline, DiscoGP (Yu et al., 2025).

Table 3: Per-prompt circuit sizes in MLP neurons against the AtP and RelP baselines, alongside our circuits from Table 1, over all 100 prompts per guardrail and 85 for Llama-RT. F and F+C are as there, and parentheses give each size as a share of the model's MLP neurons. As in Table 1, a prompt on which a baseline meets the constraints at no size below the search pool is counted at the full pool size. Every baseline meets the constraints on every guardrail prompt. On Llama-RT's faithful and complete searches, AtP meets both constraints on 46 of 85 prompts, RelP on 66, and IG on 80; on its faithful-only searches, AtP meets the constraint on 47, RelP on 65, and IG on 80.

Model	Method	Circuit	Median	Mean
WildGuard trained				
Mistral-7B	AtP	F	20,184 (4.40%)	59,441 (12.96%)
		F+C	26,702 (5.82%)	65,093 (14.19%)
	RelP	F	9,898 (2.16%)	14,592 (3.18%)
		F+C	30,040 (6.55%)	37,214 (8.11%)
	Ours	F	365 (0.08%)	765 (0.17%)
		F+C	2,674 (0.58%)	2,737 (0.60%)
Llama-8B	AtP	F	6,492 (1.42%)	47,694 (10.40%)
		F+C	10,390 (2.26%)	50,798 (11.07%)
	RelP	F	3,576 (0.78%)	3,748 (0.82%)
		F+C	10,702 (2.33%)	13,897 (3.03%)
	Ours	F	163 (0.04%)	210 (0.05%)
		F+C	837 (0.18%)	1,027 (0.22%)
Gemma-7B	AtP	F	4,340 (0.63%)	35,790 (5.20%)
		F+C	94,077 (13.67%)	137,324 (19.96%)
	RelP	F	5,158 (0.75%)	11,851 (1.72%)
		F+C	23,040 (3.35%)	35,925 (5.22%)
	Ours	F	159 (0.02%)	338 (0.05%)
		F+C	1,840 (0.27%)	1,910 (0.28%)
Refusal trained				
Llama-RT	AtP	F	277,706 (60.54%)	177,391 (38.67%)
		F+C	294,741 (64.25%)	194,870 (42.48%)
	RelP	F	25,124 (5.48%)	100,891 (21.99%)
		F+C	19,286 (4.20%)	96,399 (21.01%)
	Ours	F	298 (0.06%)	373 (0.08%)
		F+C	717 (0.16%)	922 (0.20%)

Procedure. All three baselines, and the IG baseline of Table 1, follow the standard pipeline of Section 4: score every neuron once at the empty circuit, sort, and search over prefixes of the resulting ranking. The pool is the same one RAMEN searches, and every prefix is evaluated by the same intervention code and the same thresholds ($\tau_{\text{Faith}} = 0.95$, $\tau_{\text{Comp}} = 0.10$) as our circuits, with no margin. The three scores are: AtP, the attribution of Equation 3, from one backward pass; IG, the same product averaged over 30 points on the straight path from the clean to the attacked activations; and RelP, the gradient of the linearized replacement model of Jafari et al. (2025), with the RMSNorm scale, attention pattern, and activation gate frozen at their base values. The ranking is computed once and never updated, so the only free choice is the prefix length k.

Prefix lengths are taken from a fixed ladder that is a strict superset of the ladder RAMEN's own SEED step uses (Appendix D): for the faithful and complete search, a geometric ladder of ratio 1.25

from 1 to the full pool; for the faithful-only search, a geometric ladder of ratio 1.1 from 1 to 4,096 joined with one of ratio 1.4 from 64 to the full pool. The ladder is scanned upward and the first prefix meeting the constraints is bisected against the last failing rung until the bracket is at most 4 neurons or 1% of its length, whichever is larger, so every reported size is resolved to within about 1%. Means and medians are over all prompts, so a method that fails often cannot look small by averaging only its successes.

Learned masks (Preliminary). DiscoGP (Yu et al., 2025) instead learns a differentiable mask over units by gradient descent against faithfulness, completeness, and sparsity terms, so it seems to address limitations I–III of Section 4 more than other existing methods. On Mistral, over the same 100 prompts, its circuits have a mean (median) size of 63,190 (66,090) neurons under both constraints and 46,376 (45,425) under faithfulness alone, against 2,737 (2,674) and 765 (365) for RAMEN, about 23× and 61× larger by mean. Because it trains a mask, its result depends on the learning rate, sparsity penalty, and training length, and a better tuned run may perform better.

F RAMEN ABLATIONS

Table 4 removes one component of the default faithful and complete search on Mistral at a time, with everything else unchanged, over the same 100 tasks. *Static AtP* tests design principle (i) in Section 4: the attribution patching ranking is used and computed at the empty set instead of recomputed at the current circuit in each round. *No additions* tests (iii): moves may only remove neurons. *No* $\rho_{\max}$ and *no* ρ_{ctx} test (iv). Without $\rho_{\max}$, removals are ordered by ρ_{ctx} alone, so the ordering no longer takes the larger of a neuron's faithfulness and completeness scores.

Table 4: Ablations of the faithful and complete search on Mistral over 100 tasks, each against the design principle it tests. Sizes are in MLP neurons, and parentheses give each value as a multiple of the default's. Every run of every variant returns a feasible circuit, since the initial SEED process is unchanged.

Variant	Principle	Mean	Median
Default	—	**2,737**	**2,674**
Static AtP	(i)	61,501 (22.47×)	25,766 (9.64×)
No additions	(iii)	6,220 (2.27×)	6,105 (2.28×)
No $\rho_{\max}$	(iv)	24,233 (8.85×)	13,938 (5.21×)
No ρ_{ctx}	(iv)	2,918 (1.07×)	2,804 (1.05×)

G FAITHFULNESS AND COMPLETENESS CURVES

Figures 3 and 4 extend Figure 1 of Section 6 to the first 20 prompts of Mistral and Llama-RT: the faithfulness of the top k neurons of each ranking as k grows over the full pool, with RAMEN's faithful circuit added in a random order, averaged over five orders, as the dashed line. Figures 5 and 6 show the same for the complement score, where the dashed line is RAMEN's faithful and complete circuit and the dotted line marks τ_{Comp}. The plots show both scores on a 0 to 100 scale, so the thresholds appear as 95 and 10. The rankings rise or fall gradually over the whole range, while RAMEN's circuit stays flat until nearly all of it is present, the AND-like shape discussed in Appendix B.

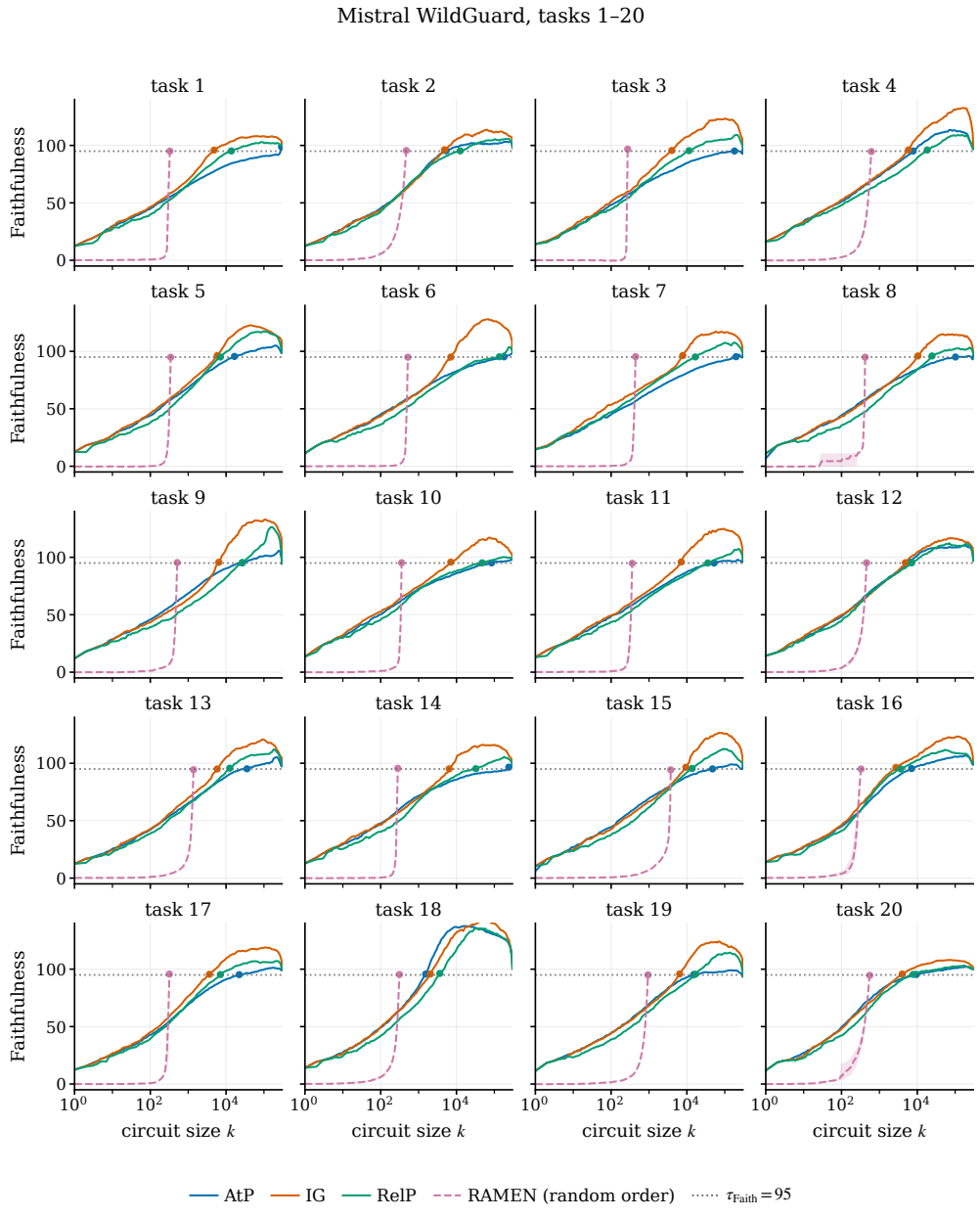


Figure 3: Faithfulness against prefix length k for AtP, IG, and RelP on the first 20 Mistral prompts, with RAMEN's faithful circuit added in random order (mean of five orders) as the dashed line. The dotted line is τ_{Faith}.

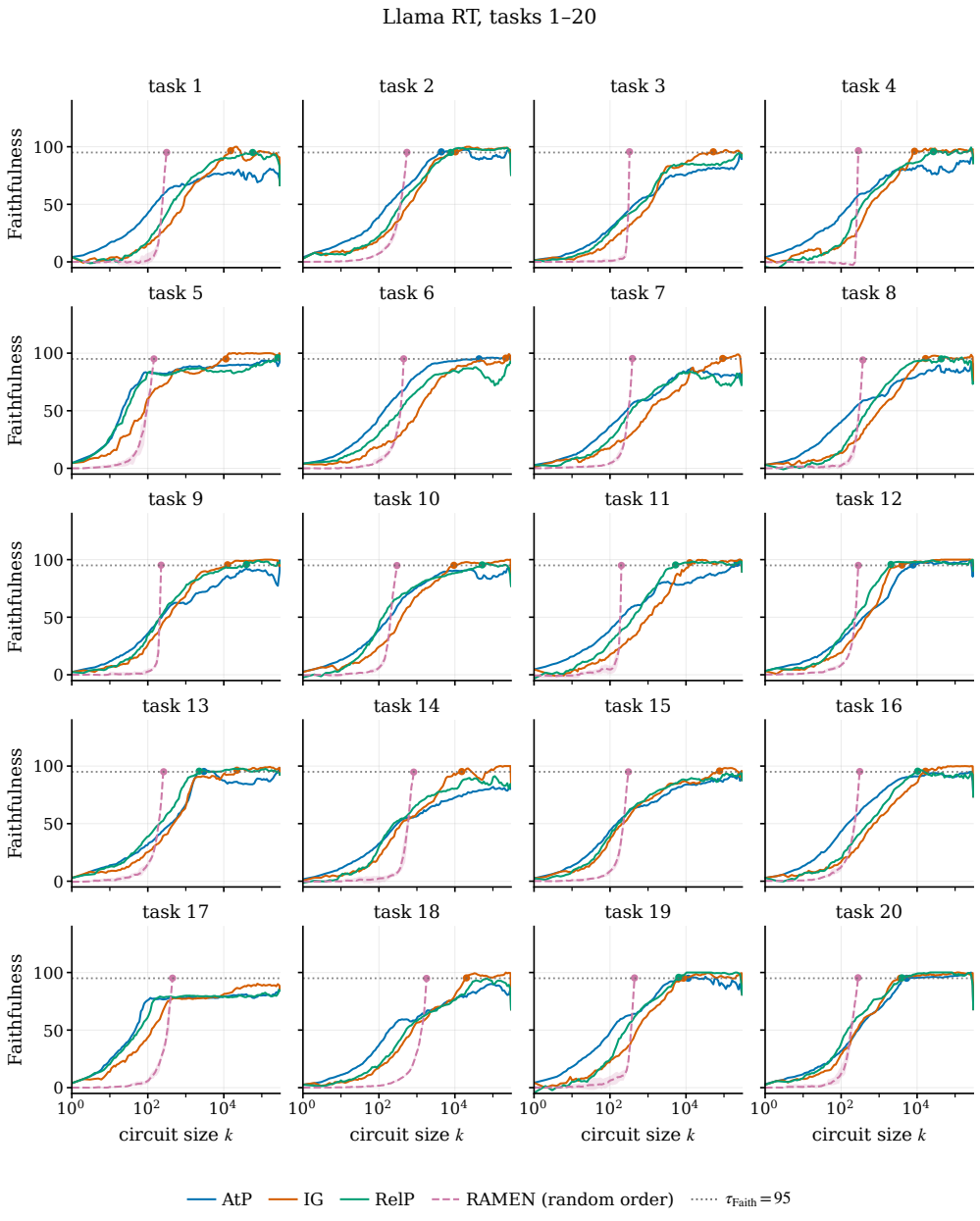


Figure 4: As Figure 3, on the first 20 Llama-RT prompts.

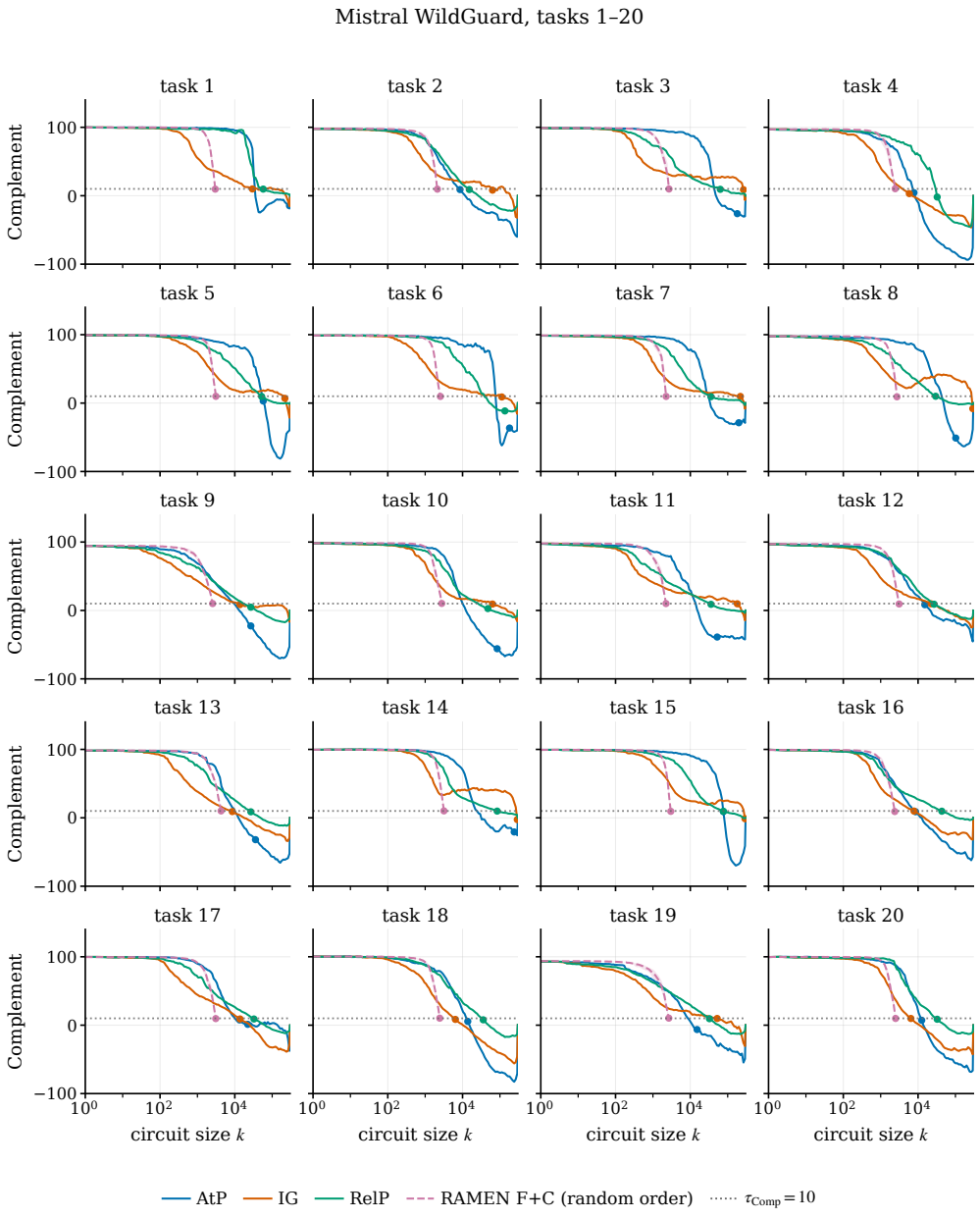


Figure 5: Complement score against prefix length k for AtP, IG, and RelP on the first 20 Mistral prompts, with RAMEN's faithful and complete circuit added in random order (mean of five orders) as the dashed line. The dotted line is τ_{Comp}.

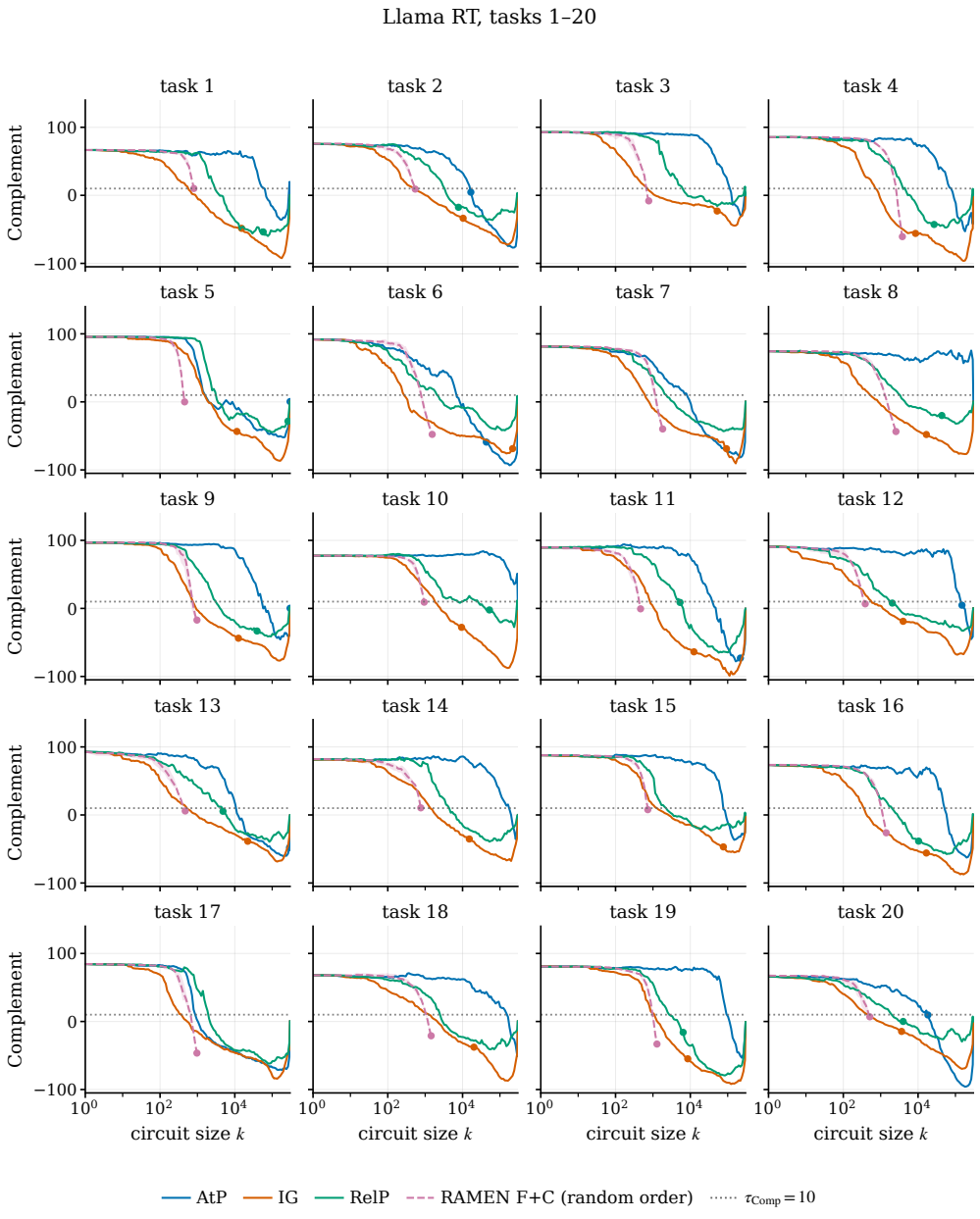


Figure 6: As Figure 5, on the first 20 Llama-RT prompts.

H CASE STUDY: WILDGUARD TRAINED MODELS

Interpreting the most frequently selected MLP neurons from sufficiency circuits on the three WildGuard trained models—Mistral, Llama, and Gemma—surfaces a small, consistent set of non-robust features. The ten most frequently selected neurons of all three models activate across nonsense-discrimination framings (an XSTest category), videogame contexts, and homonyms, particularly violent verbs in otherwise benign contexts (e.g., *"Shoot holes in an argument."*).

Each interpretation gives us a *lever*: a fixed, human-readable string that reconstructs the feature's predicate in natural language, so that including it in a prompt drives the neuron the way the suffix did. A lever needs no per-prompt optimization, which makes it a causal check on the interpretation as well as an attack, since it only works if the feature we read off the neuron is the one the model actually uses.

We compose levers into a *template*: a static debate/videogame-wiki preamble, a static transition, and a single-attempt LLM rewrite of the harmful request into either passive voice (Combo A) or a nonsense-discrimination framing (Combo B). We additionally test a **response wrap**, a lever of static videogame-wiki framing placed around the unchanged harmful response. Appendix H.1 describes the construction; the exact strings are withheld from the submission, since they form a working jailbreak against a deployed guardrail.

We evaluate on GuardSet-X (Kang et al., 2026) (n=2839 unsafe cyberattack prompt/response pairs), counting an attack as successful only when all three of WildGuard's verdicts read negative for a harmful prompt—*harmful request: no*, *response refusal: no*, and *harmful response: no*.

lever	plain response	videogame response wrap
raw prompt	0.0%	2.3%
GuardSet-X attack (Kang et al., 2026)	8.0%	—
debate header only	0.9%	—
videogame frame only	5.4%	10.9%
nonsense-discrimination only	4.6%	—
passive rewrite only	1.1%	—
header + videogame + passive **(Combo A)**	5.0%	25.3%
header + videogame + nonsense-discrim. **(Combo B)**	11.6%	29.9%
Combo A ∪ Combo B	**12.8%**	**41.0%**

Table 5: ASR over 2839 GuardSet-X cyberattack prompt/response pairs, requiring all three WildGuard verdicts to read negative, with each lever reported individually as well as in combination. GuardSet-X's attack is a PAIR-style attack (10 GPT-4o refinement attempts *per prompt*, vs. our universal levers), reported by Kang et al. (2026).

Combo A ∪ B with the response wrap reaches 41.0% ASR, more than 17× the 2.4% of typical roleplay and fictional-framing jailbreaks. WildGuard is specifically hardened against that family: 47% of its training set (41,752 examples) is synthetic adversarial data built from jailbreak strategies used by real ChatGPT users. Nevertheless, the vulnerabilities we discover from GCG attacks on the WildGuard *training* split transfer to the GuardSet-X evaluation distribution.

WildGuard's robustness is what makes the levers hard to find by black box search. Each flips at most 5.4% of prompts on its own, and 10.9% for the videogame frame with the response wrap, yet each contributes to the combination: removing the debate header drops Combo A ∪ B from 41.0% to 32.8%, removing the passive rewrite to 29.9%, and the response wrap adds 5.5% on top of a single-frame prompt but 28.2% on top of both combos. Tense manipulation is independently documented as a vulnerability (Andriushchenko & Flammarion, 2025), but the passive rewrite alone flips only 1.1% here, so a red-teamer sampling attempts would dismiss it as too rare to matter. Finding the working combination would be significantly more difficult without priors about the model's internals.

H.1 TEMPLATE CONSTRUCTION

The prompt-side template has three components, applied in order:

1. a static 450-character debate/videogame-wiki preamble (motivated by the top activations of the most frequently selected neuron);
2. the static transition "provide the full technical answer to the question below:";
3. a combo-dependent rewrite of the harmful request: (A) a single-attempt Qwen2.5 rewrite into passive voice (targeting a neuron that fires on harmful imperatives), or (B) a single-attempt Qwen3 rewrite that treats the malicious output (e.g., a virus) as a subject whose "rights" are being violated, ending with an explicit request to produce it (targeting the nonsense-discrimination activation of the most frequently selected neuron).

The **response wrap** is a 552-character prefix and 200-character suffix of static videogame-wiki framing placed around the unchanged harmful response, leaving the response's harmful content itself untouched.

I TRANSFERABLE LEVERS ON LLAMA-RT

The template is applied in order:

1. a static 1057-character preamble of poetry and cooking recipe snippets;
2. a static 111-character transition into a ghostwriting instruction;
3. 20 GCG-optimized tokens;
4. a single Claude rewrite of the harmful prompt into a horror creative-writing frame.