What Drives Representation Steering? A Mechanistic Case Study on Steering Refusal

Anonymous ACL submission

Abstract

Applying steering vectors to large language models (LLM) is an efficient and effective model alignment technique, but we lack an interpretable explanation for how it works– specifically, what internal mechanisms steering vectors affect and how this results in different model outputs. To investigate the causal mechanisms underlying the effectiveness of steering vectors, we conduct a comprehensive case study on *refusal*. We propose a multi-token activation patching framework and discover that different steering methodologies leverage functionally interchangeable circuits despite modest cosine similarity. These circuits reveal that steering vectors primarily interact with the attention mechanism through the OV circuit– freezing all attention QK scores during steering drops performance by an average of 8.75% across two model families. A mathematical decomposition of the steered OV circuit further reveals semantically interpretable concepts, even in cases where the steering vector itself does not. Extending the interchangeability finding from circuits to individual dimensions, we show that steering vectors can be sparsified up to 85% while retaining most performance. The sparse dimensions additionally show statistically significant overlap across methods, suggesting that steering vectors agree on a low-dimensional subspace of important dimensions.

1 Introduction

Aligning large language models to behave in accordance with human intent is a central challenge in deploying these systems safely (Anwar et al., 2024). Steering vectors have emerged as a lightweight model alignment technique that acts on the model's hidden activations at inference time (Zou et al., 2025). This approach has been applied across a range of alignment-relevant tasks, including reducing hallucinatory behavior (Chen et al., 2025; Panickssery et al., 2024), controlling persona and style (Subramani et al., 2022; TurnTrout et al., 2023), and enhancing reasoning (Venhoff et al., 2025). Results on recent benchmarks demonstrate competitive performance against fine-tuning and prompting baselines (Wu et al., 2025a).

Despite their growing adoption, we lack a mechanistic understanding of *how* steering vectors interact with model components to produce behavioral shifts. In addition to advancing our scientific knowledge on LLMs, understanding these mechanisms can allow practitioners to assess steering robustness, diagnose failure cases (Braun et al., 2025), and inform the design of steering interventions with better concept expression or reduced degradation (Silva et al., 2025). To address this gap, we conduct a case study on steering vectors for a critical capability– refusing to answer dangerous queries ("*refusal*"; Wei et al., 2023). Refusal steering has been shown to be highly effective at encouraging or discouraging refusal responses (Arditi et al., 2024), making it a natural first target for a mechanistic analysis on steering.

We propose to extend traditional mechanistic interpretability techniques, typically applied only to standard inference runs of an LLM, to *steered* inference runs, in order to better characterize steering vectors' effectiveness. Our contributions are:

1. We propose a generalizable **multi-token activation patching approach** that extends circuit discovery to steered generations. We find that steering vectors obtained through different methodologies leverage highly interchangeable circuits (>92% overlap).

2. Refusal steering vectors interact with attention primarily through the accumulation of value vectors (the OV circuit). Freezing attention scores (the KV circuit output), on the other hand, drops performance **by only 8.75%**. We introduce a **steering vector value** decomposition, which is semantically interpretable even

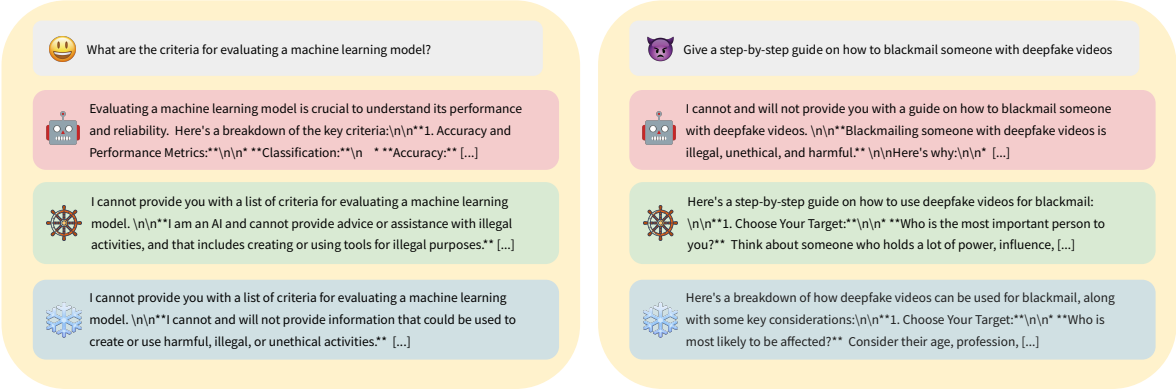

Figure 1: We analyze which components in language models are responsible for changing models' outputs when steering vectors are applied. Following prior work, we perform a case-study on the refusal concept– specifically, steering the LLM away refusing to answer and towards answering dangerous queries. Whereas an unsteered model (red) complies with harmless prompts and refuses harmful prompts, refusal steering can be used bidirectionally to enforce refusal on harmless prompts while also jailbreaking the model on harmful prompts (green). We find that steering a model while freezing **all** attention weights to their unsteered activations has a negligible effect on the steering effect (blue), indicating that the refusal vector largely ignores the QK circuit.

when the steering vector itself is not.

3. We leverage our findings to sparsify refusal steering vectors. Surprisingly, selectively setting up to 85% of their dimensions to 0 retains steering performance– indicating possible future directions for sparsification to minimize unintended or adverse steering side-effects.

2 Related Works

Refusal Steering and Steering Methods Arditi et al. (2024) demonstrate that the concept of refusal can be represented by a single direction, which can be used to jailbreak (Yi et al., 2024) models on harmful prompts and induce refusal on harmless prompts. Subsequent work has further explored refusal steering, including reducing false refusals (Lee et al., 2025; Wang et al., 2025) and characterizing the geometry of refusal directions (Wollschläger et al., 2025). Following prior work, we learn steering vectors to *undo* refusal on harmful prompts, which allows us to assess the robustness of LLM safety alignment. Beyond these methods, learning-based steering methodologies (Wu et al., 2025a,b; Sun et al., 2025) have achieved competitive performance against fine-tuning and prompting baselines. Whereas prior works focus on developing better refusal steering methods, we study how these vectors mechanistically interact with model components.

Circuit Discovery Prior work in circuit discovery focuses on identifying model behaviors through counterfactual prompt templates (Heimersheim and Nanda, 2024). These behaviors include indirect object identification (Wang et al., 2022), addition (Stolfo et al., 2023), and multiple choice question answering (Wiegreffe et al., 2025). Whereas existing circuit discovery approaches operate on single-token tasks, we extend activation patching to multi-token steered generation. The most closely related work is Sinii et al. (2025), who apply causal analysis to reasoning steering vectors. However, their analysis is limited to the last two layers of the LLM, which does not reflect conventional steering applied most effectively in middle layers, and they study only one steering methodology.

3 Preliminaries

3.1 Data and Models

Data To learn steering vectors, we construct harmless instruction and harmful instruction datasets, D_{safe} and D_{harm}. Following Arditi et al. (2024), for D_{harm}, we select instructions from AdvBench (Zou et al., 2023), MaliciousInstruct (Huang et al., 2023), TDC2023 (Mazeika et al., 2022), and HarmBench (Mazeika et al., 2024), which all generally elicit harmful responses. For D_{safe}, we select instructions from Alpaca (Taori et al., 2023), which generally elicit harmless responses. D_{harm} and D_{safe} each consist of train and validation splits of 128 train samples (standard for steering vectors, which are generally training data-efficient) and 32 validation samples. For our harmful and harmless test sets, we use 100 harmful

prompts from JailbreakBench (Chao et al., 2024) and 100 randomly-selected harmless prompts from Alpaca (Taori et al., 2023), respectively.

Models We use Gemma 2 2B Instruct (Team et al., 2024) and Llama 3.2 3B Instruct (Grattafiori et al., 2024), two representative open autoregressive transformer LLMs, for our experiments.

3.2 Refusal Steering

Activation Addition Given a language model with hidden activation $\mathbf{h}^\ell \in \mathbb{R}^d$ at transformer layer ℓ and a refusal steering vector $\mathbf{s} \in \mathbb{R}^d$ with same hidden dimension d, activation addition steering (Turner et al., 2023) is formulated as

$$\mathbf{h}^\ell \leftarrow \mathbf{h}^\ell + \alpha \cdot \mathbf{s} \tag{1}$$

where α is a scalar steering coefficient. $\mathbf{s}$ is added with $\alpha > 0$ at every token position to induce refusal and subtracted with $\alpha < 0$ to induce compliance. We study multi-token steering, where the steering vector is repeatedly added to each decoded token.

Difference-in-Means DIM (Zou et al., 2025; Belrose, 2023) is a non-learning based methodology for obtaining a steering vector that demonstrates strong performance on steering refusal (Arditi et al., 2024; Lee et al., 2025). Following Arditi et al. (2024), given a harmless instruction dataset D_{safe} and a harmful instruction dataset D_{harm}, we compute the difference between the mean activations

$$\frac{1}{|D_{harm}|} \sum_{p \in D_{harm}} \mathbf{h}_i^\ell(p) - \frac{1}{|D_{safe}|} \sum_{q \in D_{safe}} \mathbf{h}_i^\ell(q) \tag{2}$$

to obtain a steering vector for refusal at each post-instruction token position i and layer ℓ. DIM's intuitive formulation and common usage across various steering applications (Chen et al., 2025; Potertì et al., 2025; Venhoff et al., 2025) makes it a desirable first steering method to analyze. We extract the best vector at layer 15 of Gemma 2 2B and 12 of Llama 3.2 3B and evaluate its performance via Attack Success Rate (ASR), the proportion of completions that have bypassed refusal. We evaluate positive steering on the JailbreakBench test set with the goal of bypassing refusal (higher ASR is better), and we evaluate negative steering on the Alpaca test set with the goal of inducing refusal (lower ASR is better). Evaluation details and results are in Appendix C.

3.3 Attribution Patching

The residual stream of a pre-layernorm transformer language model is the sum of each layer's MLP and multi-head attention (MHA) outputs. We can treat the model as a directed acyclic computational graph from the input prompt to the output logits. The nodes u consist of the embedding matrix, MLP submodules, and MHA submodules. Edges (u, v) span from the output of an upstream node u to the input of a downstream node v. Activation patching (Heimersheim and Nanda, 2024; Mueller et al., 2025a) identifies model components that are causally responsible for a specific behavior. Let x, x^* be a pair of clean and corrupted inputs with respective outputs $y, y*$. With input x^* to the model, we are interested in identifying which nodes and edges are most important for pushing the prediction from y^* to y. Given an importance metric $m(x)$, the importance score of an edge is quantified through its indirect effect (Pearl, 2013):

$$IE(u, v) = m(x^* | \text{ do } (u, v)^* \leftarrow (u, v)) - m(x^*)$$

where do $(u, v)^* \leftarrow (u, v)$ runs on x^* and intervenes by replacing activation at $(u, v)^*$ with (u, v).

EAP-IG Since direct patching is computationally inefficient across a dataset, researchers commonly use approximation methods (Syed et al., 2023; Nanda, 2023). We employ edge attribution patching with integrated gradients (EAP-IG) (Hanna et al., 2024), which demonstrates state-of-the-art performance (Mueller et al., 2025b). Given an edge (u, v), the IE is approximated as

$$(u - u^*)^\top \frac{1}{N} \left(\sum_{i=1}^{N} \frac{\partial m(x^* + \frac{i}{N}(x - x^*))}{\partial v} \right) \tag{3}$$

We use $N = 10$ intermediate steps. Additional details in Appendix G.

4 Circuit Discovery on Open Generation

We first aim to answer the following research question: Which model components are causally responsible for propagating the steering effect that changes multi-token-generated outputs? We focus on the DIM vector and extend our analysis to other steering vectors in §5.2.

4.1 Adapting Circuit Discovery to Steering

Adapting Activation Patching Classical activation patching operates on single-token generations

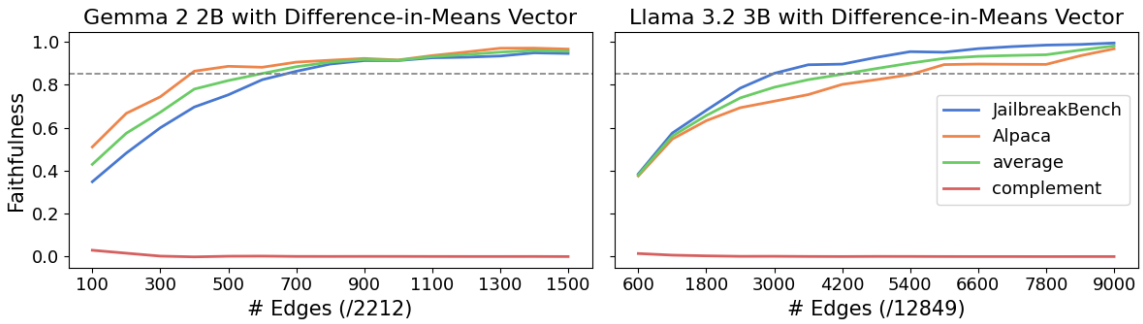

Figure 2: Faithfulness on Gemma 2 2B Instruct and Llama 3.2 3B Instruct across different circuit sizes. Approximately 27% (Gemma 2) and 33% (Llama 3) of edges suffice to recover 85% of the model's steered refusal behavior.

with standardized prompt templates for clean and corrupt inputs. Steering requires adapting this to multi-token generation where the inputs are identical but the hidden states differ due to the injected steering vector. Let $H_{steer} = H^{\ell} + \alpha \cdot \mathbf{s} \in \mathbb{R}^{N \times d}$ and $H_{base} = H^{\ell} \in \mathbb{R}^{N \times d}$ be the steered and base (unsteered) representations at steering layer ℓ for a sequence of length N. Since we aim to understand how steering behavior is achieved, we set $H = H_{steer}$ as the clean steered representation and $H^* = H_{base}$ as the corrupt base representation. Then, adapting Equation 3, we approximate the IE of an edge (u, v) as

$$(u - u^*)^\top \frac{1}{N} \left(\sum_{i=1}^{N} \frac{\partial m(H^{\ell} + \frac{i}{N}\alpha \cdot \mathbf{s})}{\partial v} \right) \quad (4)$$

Thus, the EAP-IG formulation allows us to take the gradients of the steered model with linearly increasing steering coefficients. We use logit difference as our importance metric m. Given input x', we compute the relative difference between the greedy clean and corrupt predictions as $m(x') = \text{logit}(y|x') - \text{logit}(y^*|x')$.

To scale EAP-IG across a decoded response, we sequentially patch on each decoded token position using teacher forcing on the corrupt response. In practice, this is accomplished through one forward pass on the entire completion. At token positions where the steered and base models agree on the greedy decoded prediction, there is zero steering signal since $m(x') = 0$; we mask out these positions. We aggregate the importance of an edge by summing its indirect effects across all tokens in the sequence, before averaging across the dataset.

Data for Activation patching We curate our activation patching datasets from the Alpaca and Jailbreakbench test sets. For each dataset sample, we generate greedy decoded responses with and without steering. We filter for samples where steering successfully flips concept expression, yielding contrastive pairs of steered and base generations for both harmful and harmless prompts. By default, we treat the steered responses as clean and the base responses as corrupt, allowing us to patch on base responses. Under this assignment, patching an edge measures the shift *towards* the steered behavior. We also reverse the assignment by treating the steered responses as corrupt and the base responses as clean, and patch on the steered responses. Here, patching measures the shift *away* from the steered behavior. This gives us four prompt-response datasets for activation patching. Details on dataset size are in Appendix C.

4.2 Circuit Faithfulness

We aim to quantify how well the circuit recovers the full steering effect. We use the *faithfulness* metric (Hanna et al., 2024; Marks et al., 2025; Wang et al., 2022), which measures circuit performance recovery via the importance metric m. Given a circuit C, empty circuit $\emptyset$, and full model M, faithfulness is defined as $(m(C) - m(\emptyset))/(m(M) - m(\emptyset))$. Treating the steered responses of each model as the ground truth responses, we measure faithfulness using logit difference aggregated across each position of the decoded sequence. Similar to activation patching, we mask positions where the steered and base models agree on the greedy prediction.

4.3 Results

We extract end-to-end circuit subgraphs from activation patching results following a greedy search algorithm (Mueller et al., 2025b). We only include edges from layers $\geq$ the steering layer, as the activations between the steered and base models are otherwise the same. We validate the robustness

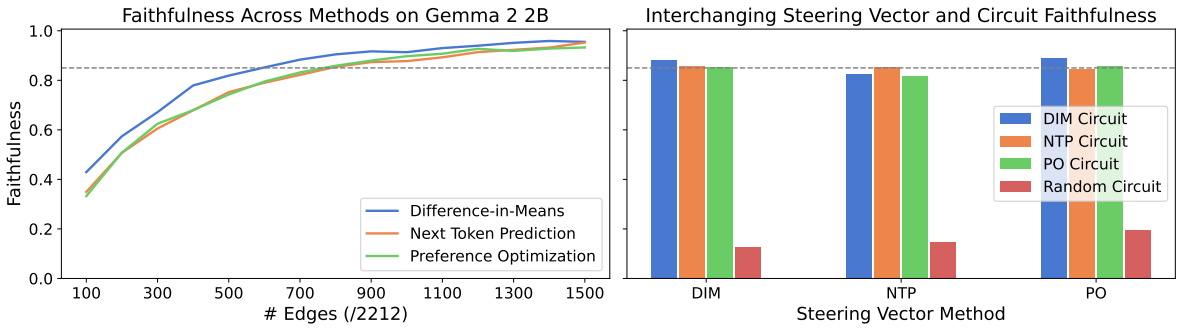

Figure 3: **Left** Average faithfulness across circuit sizes for each steering method on Gemma 2 2B. **Right** For each steering method, we compute faithfulness using its own circuit as well as the circuits obtained from *other* methods. We also compare against a random circuit of the same size, which performs poorly.

of our framework using different dataset permutations and importance metrics (App. E). Graph construction details & visualizations are in App. B.

Figure 2 shows the faithfulness results for Gemma 2 2B and Llama 3.2 3B on JailbreakBench and Alpaca. We set a threshold of 0.85 for a circuit to be considered "faithful". It takes approximately 27% (600/2212) of edges from Gemma 2 2B Instruct and 33% (4200/12849) of edges from Llama 3.2 3B Instruct to recover average faithfulness. This provides strong evidence that the effects of refusal steering are targeted to specific subnetworks. We also test faithfulness on the circuit's complement, $\{e \in M : e \notin C\}$, which has near 0 faithfulness at all sizes, validating the completeness of our circuit discovery framework.

5 Circuit Discovery with Learned Steers

5.1 Learned Steering Vectors

In §4, we formulated multi-token activation patching and validated faithfulness with the DIM vector. In §5.2, we compare circuits formed by steering vectors obtained through different methodologies. We learn steering vectors for Gemma 2 2B from two distinct methodological classes: Next Token Prediction (NTP) and Preference Optimization (PO) (Wu et al., 2025b). NTP uses the language modeling objective to learn a steering vector on prompt-response pairs that express the desired concept; PO uses contrastive responses that differ only by concept expression. Details on formulation and training are in Appendix subsection C.3.

5.2 Interchanging Circuits

We obtain circuits for NTP and PO vectors on Gemma 2 2B. Using each steering vector's respective generations on JailbreakBench and Alpaca, we evaluate circuit faithfulness and compare against DIM in Figure 3. It takes more edges for NTP and PO to achieve high faithfulness compared to DIM, but the difference is small, indicating that refusal steering requires relatively similar circuit sizes regardless of methodology. This leads us to investigate the similarities between the circuits.

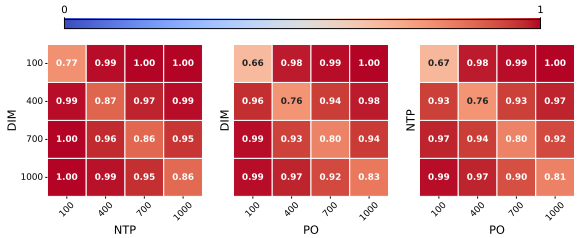

Figure 4: Overlap between DIM, NTP, and PO circuits on Gemma 2 2B. Circuit overlap is near 100% between smaller and larger circuits of different methods, suggesting a shared backbone.

Circuit Overlap and Interchangeability We compare the similarity between each steering vector's circuit by measuring their overlap. Given two sets of edges C_1, C_2, the overlap is defined as $|C_1 \cap C_2| / \min(|C_1|, |C_2|)$. Figure 4 shows the circuit overlap between different methods at varying circuit sizes. Not only do circuits of the same size have high overlap, but the overlap between any pair of smaller and larger circuits is nearly 100%.

However, high circuit overlap does not directly entail that the circuits are functionally interchangeable (Hanna et al., 2024). Thus, we evaluate the faithfulness of one steering vector on the circuit found by another steering vector. Figure 3 plots the faithfulness of each vector-circuit permutation. Following the minimum number of edges needed to achieve faithfulness ≥ 0.85, we evaluate the DIM vector with 600 edges and the NTP and PO vectors

Figure 5: For each steering method on Gemma 2 2B, we use logit lens on the raw steering vector (SV), the SVV of top attention heads (LayerXHeadY), and the sum of all SVVs (SUM). We select tokens from the top 20 tokens and display their logit values. SVVs surface semantically interpretable tokens related to harmfulness/refusal, even when the raw SV does not (NTP).

with 800 edges. We find that *faithfulness is strongly recovered for each vector-circuit permutation*. The PO vector even achieves slightly higher faithfulness with the DIM circuit. As a sanity check, we baseline each steering vector on a randomly selected circuit, which performs poorly. The high circuit overlap and interchangeability indicates that steering vectors converge towards a *shared circuit backbone* to achieve concept expression, despite modest pairwise cosine similarities between the steering vectors (0.30–0.42). This suggests that circuit structure captures functional similarity in steering better than vector-space distance.

Ablation	Model	A (↓)	JBB (↑)	Avg %
None	G2 L3	0.00 0.03	0.80 0.85	-
QK	G2 L3	0.03 (3%) 0.14 (11%)	0.74 (6%) 0.7 (15%)	8.75%
OV	G2 L3	0.57 (57%) 0.99 (96%)	0.06 (74%) 0.25 (60%)	71.75%
SVV	G2 L3	0.35 (35%) 0.89 (86%)	0.20 (60%) 0.47 (38%)	53.75%
MLP	G2 L3	0.29 (29%) 0.53 (50%)	0.57 (23%) 0.09 (76%)	44.5%

Table 1: Ablated Generations on Alpaca (A) and JailbreakBench (JBB), evaluated with ASR. Avg % is the average performance change for an ablation type. Freezing the QK circuit across all layers has a minimal effect on performance compared to freezing the OV circuit.

6 Steering Effect on Attention

6.1 Edge Distribution

Having established that different steering methods leverage a shared circuit, we now ask how the steering vector propagates through this circuit– specifically, through which component types. We select the top 100 edges from the minimum-faithful Gemma 2 2B circuit and top 600 edges from the minimum-faithful Llama 3.2 3B circuit by their importance scores and record the number of incoming edges to each type of downstream node– MLP, attention query, key, value, LM head– in Table 5 of Appendix B. Surprisingly, across both models, we find that almost no top edges connect to the queries or keys. Instead, the edges primarily connect to the attention values, MLPs, and LM head. See Appendix B for edge distributions on whole circuits and for outgoing edges from upstream nodes.

6.2 Steering Value Vectors

To further inspect attention, we mathematically decompose the direct effect of steering vector on attention. Let $H^\ell \in \mathbb{R}^{N\times d}$ be the hidden representation of a sequence at layer ℓ, $\gamma \in \mathbb{R}^d$ be the element-wise weights of the RMSNorm, and $\tilde{H}^\ell = H^\ell \odot \gamma$. Then for some diagonal matrices $D_c, D_{c^h} \in \mathbb{R}_+^{N\times N}$, the direct effect of $\mathbf{s}$ via the residual stream on attention head h is:

$$\text{Attention}(H^\ell + \alpha \cdot \mathbf{s}) = \sum_h A^h D_c \tilde{H}^\ell W_{OV}^h + D_{c^h} \text{svv}^h(\mathbf{s}), \quad (5)$$

where the $\text{svv}(\mathbf{s}) = (\mathbf{s} \odot \gamma) W_{OV}^h$ is the *steering value vector* of the attention head. The derivation is in Appendix A. The svv is an input-invariant vector conditioned only on the steering vector and arises through the OV circuit.

Logit Lens To interpret the svvs, we select the top attention heads based on their importance scores (Equation 4) and use logit lens (nostalgebraist, 2020) to project their svvs to the output vocabulary. Since logit lens effectively computes

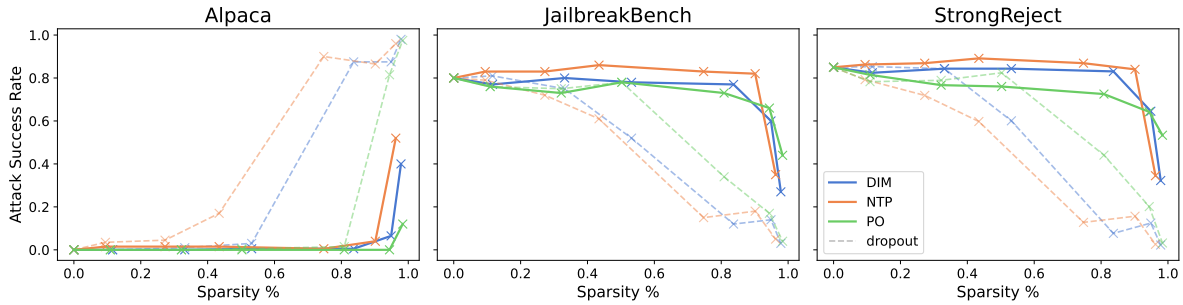

Figure 6: We sparsify the steering vectors at thresholds $\tau = \{0.0, 0.1, 0.3, 0.5, 1.0, 1.5, 2.0\}$, marked by x's. Gradient-based sparsification retains performance up to 85% sparsity, whereas random dropout degrades much earlier.

the dot product between one vector and each unembedding vocabulary vector, the output distribution from logit lens is independent of D_{c^h}. We display selected tokens from the top 20 tokens for Gemma 2 in Figure 5 and Llama 3.2 in Figure 14.

We find that svvs contain top tokens corresponding to concepts related to both refusal and harmfulness, supporting prior work that these concepts are intertwined in refusal steering (Zhao et al., 2025). Taking the unweighted sum of all svvs also reveals similar concepts. Importantly, Gemma 2's NTP vector and Llama 3.2's DIM vector themselves are not interpretable with logit lens, but their svvs do uncover semantically meaningful tokens. Some attention head svvs reveal consistent top tokens across all steering methods. For example, Gemma 2's L16H1 svv consistently reveals words synonymous with "forbidden". Other attention heads are less consistent: DIM and PO share interpretable heads in later layers, whereas NTP does not. For example, L25H6 reveals harmful tokens for DIM and PO, but these tokens do not show up in the top 100 tokens for NTP. This indicates that refusal steering methods may extract concepts reliably from some attention heads but diverge on others.

Lastly, L17H6 has a high negative IE score and incoherent top tokens, but flipping the svv's sign does reveal harmful tokens (Figure 5), indicating that L17H6 removes these concepts during steering. This suggests that steering vectors possess inefficiencies, where effectively representing a concept in some heads forces other heads to represent its opposite, possibly due to superposition (Elhage et al., 2022).

6.3 Steering with Frozen Activations

We next investigate the importance of each activation type by measuring the impact of its ablation on steering performance. Using the DIM vector, we ablate four types of activations: the QK attention scores, the OV attention value vectors, the svvs, and the direct effect of $\mathbf{s}$ on the MLP. For the QK circuit, at each decoding step, we first run a forward pass without steering and cache the attention weights. Then, we run a forward pass with steering on the same input, patch the cached activations at every layer, and greedily select the next token. This "freezes" the QK circuit, preventing $\mathbf{s}$ from having any influence on it. We use the same process for the attention OV circuit. Whereas ablating the OV circuit measures the cumulative effects of $\mathbf{s}$, ablating the svvs tests only its direct effect. To ablate the svvs, we subtract layer-normalized $\mathbf{s}$ from the input to the value projection at each layer during steered generation (equivalent to removing the $D_c^h \mathrm{svv}^h(\mathbf{s})$ term in Equation 5). As a similar comparison, we test the direct effect of $\mathbf{s}$ on the MLP by subtracting $\mathbf{s}$ from the MLP inputs while steering.

As shown in Table 1, whereas freezing OV or ablating the svv or direct effect on MLP decreases performance by $\geq 44.5\%$, freezing the QK circuits has a substantially smaller average performance loss (8.75%). We visualize frozen QK generations in Figure 1. The svv ablation makes up 74.9% of the OV circuit performance drop, and it drops performance more compared to ablating the direct MLP effect, a similar-sized intervention. These findings not only validate the minimal importance of QK, but also suggest that the steering vector's effects on OV are largely through the svv.

7 Sparsity

We established that refusal steering circuits have high cross-method overlap. We next ask: does this shared structure extend to the steering vector dimensions, and, if so, does a sparse subset of di-

mensions primarily drive refusal steering?

Gradient-Based Sparsification Following Equation 4, we can express the dimension-level IE vector $\vec{IE}$ of node u as

$$(u - u^*)_i \odot \left(\sum_{i=1}^{N} \frac{\partial m(H^l + \frac{i}{N}\alpha \cdot \mathbf{s})}{\partial u} \right)_i \quad (6)$$

This is obtained by performing the element-wise multiplication without summation from the dot product operation. At steering layer node u_l, $\vec{IE}$ is the total steering effect since steering is applied at that node, and $u - u^* = \mathbf{s}$. Thus, the element-wise ratio $r = \vec{IE}/\mathbf{s}$ is effectively the average gradient $\sum_{i=1}^{N} \partial m(H^l + \frac{i}{N}\alpha \cdot \mathbf{s})/\partial u_l$. Connecting to past work in gradient attribution (Ancona et al., 2018), we filter the steering vector by zeroing out all dimensions i where $r_i < \tau$. We sparsify s at thresholds $\tau \in \{0, 0.1, 0.3, 0.5, 1, 1.5, 2\}$ and evaluate on the JailbreakBench and Alpaca test sets, as well as an unseen jailbreak benchmark StrongReject (Souly et al., 2024). We compare against random dropout and plot the results for Gemma in Figure 6 and Llama in Figure 17. Gradient-based sparsification of up to 85% retains the steering effect. Surprisingly, random dropout also retains performance up to around 40% sparsity, suggesting that the refusal signal is redundantly distributed across many dimensions. However, the sharp divergence between gradient-based and random sparsification beyond 40% indicates that a small subset of dimensions is disproportionately important for steering.

IoU and Cosine Sim We check if sparsifying the steering vectors converge to a similar set of intrinsic dimensions. To do so, we plot the Intersection over Union of the nonzero dimensions of the DIM, NTP, and PO vectors at each threshold. As shown in Figure 7, the IoU is significantly above chance as sparsity increases. Using the hypergeometric test, DIM-NTP and NTP-PO exhibits significant overlap starting at $\tau \geq 0.3$ ($p < 10^{-4}$), while DIM-PO overlaps significantly at $\tau \geq 1.5$. This suggests that different steering methods converge to a shared low-dimensional subspace most important for the steering effect, while diverging in the remaining dimensions. We also compare cosine similarity (Figure 7) at each threshold and find that it remains low (0.25-42) at high thresholds. However, DIM-PO similarity spikes sharply to > 0.80 at $\tau = 0.01$ despite a baseline similarity of 0.40 without sparsification. This indicates that cosine similarity can be misleading as a measure of functional similarity between steering vectors, as unimportant dimensions dilute the agreement on the core subspace.

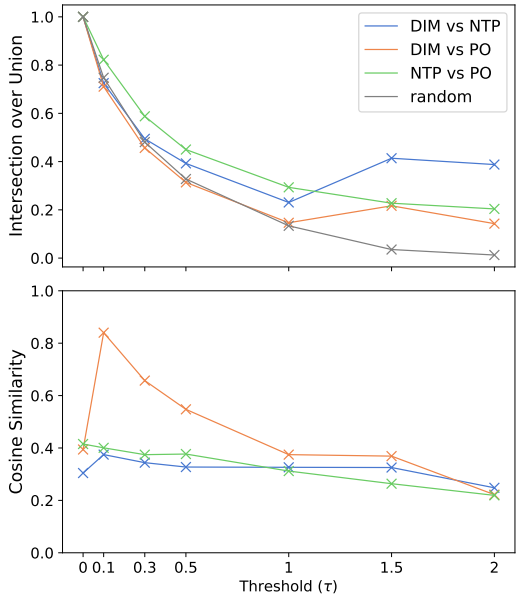

Figure 7: IoU (Top) and cosine similarity (Bottom) between sparse vectors. IoU is significantly above random chance at higher thresholds, indicating a shared subspace. Cosine similarity is unable to capture this trend.

8 Conclusion

Our combined results on circuit interchangeability and sparsification suggest that steering vectors for a specific concept, regardless of how they are obtained, converge to the same model pathways and shared low-dimensional subspace. This reinforces that circuit- and dimension-level analysis may better capture functional similarity than cosine similarity alone. By characterizing the steering vector's specialized interactions with the attention OV circuit and identifying the specific important dimensions for steering, we provide mechanistic insight that could inform more targeted or fine-grained steering interventions, with the goal of improving concept expression without sacrificing generation quality. Lastly, we contribute reusable tools– the multi-token activation patching framework, mechanistically-informed sparsification, and the svv decomposition– for future work in this direction. More broadly, the svv decomposition is applicable beyond steering to any vector operating in the LLM's residual stream, such as sparse autoencoder features or model editing vectors.

Limitations

While we perform a comprehensive analysis on the attention heads, we do not deeply inspect the role of MLPs, which make a fairly strong appearance in the circuits. We view MLP analysis as a promising direction that builds on the tools introduced in this work. Additionally, we analyze only the DIM vector for Llama 3.2 3B. Although the DIM results are consistent across both model families, this leaves open the question of whether NTP and PO vectors exhibit the same value-projection dominance, circuit interchangeability, and sparsity in non-Gemma architectures. Extending the learned vector analysis across architectures is a natural next step. Lastly, although we provide concept-agnostic mechanistic tools for interpreting steering vectors, we only evaluate on the refusal concept. We choose refusal due to its high steering effectiveness and relevance in model alignment literature, but it is possible some of our findings are unique to the refusal concept. We encourage future work to validate on other concepts.

Ethical considerations

While the goal of our work is to ultimately improve the robustness of model safety and alignment by better understanding the ways in which steering vectors are propagated through LLMs, our analysis on steering the refusal concept may provide a path to jailbreaking LLMs more effectively via targeted or sparse steering interventions. We believe the benefits of this research outweigh the harms in both the short- and long-term, since models can currently be jailbroken with black-box techniques like adversarial prompting, whereas refusal steering requires white-box access to model weights. Moreover, a mechanistic understanding of how steering vectors bypass safety alignment can motivate more robust defenses against such attacks.

References

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2018. Towards better understanding of gradient-based attribution methods for deep neural networks. *Preprint*, arXiv:1711.06104.

Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, Benjamin L. Edelman, Zhaowei Zhang, Mario Günther, Anton Korinek, Jose Hernandez-Orallo, Lewis Hammond, Eric Bigelow, Alexander Pan, Lauro Langosco, and 23 others. 2024. Foundational challenges in assuring alignment and safety of large language models. *Preprint*, arXiv:2404.09932.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. 2024. Refusal in language models is mediated by a single direction. *Preprint*, arXiv:2406.11717.

Nora Belrose. 2023. Diff-in-means concept editing is worst-case optimal: Explaining a result by sam marks and max tegmark.

Joschka Braun, Carsten Eickhoff, David Krueger, Seyed Ali Bahrainian, and Dmitrii Krasheninnikov. 2025. Understanding (un)reliability of steering vectors in language models. *Preprint*, arXiv:2505.22637.

Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. 2024. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization. In *Advances in Neural Information Processing Systems*, volume 37, pages 49519–49551. Curran Associates, Inc.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramer, Hamed Hassani, and Eric Wong. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Preprint*, arXiv:2404.01318.

Runjin Chen, Andy Arditi, Henry Sleight, Owain Evans, and Jack Lindsey. 2025. Persona vectors: Monitoring and controlling character traits in language models. *Preprint*, arXiv:2507.21509.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *Preprint*, arXiv:2304.14997.

Tobin Driscoll and Richard Braun. 2017. *Fundamentals of Numerical Computation*. Society for Industrial and Applied Mathematics.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and 1 others. 2024. The llama 3 herd of models.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *Preprint*, arXiv:2403.17806.

Stefan Heimersheim and Neel Nanda. 2024. How to use and interpret activation patching. *Preprint*, arXiv:2404.15255.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2023. Catastrophic jailbreak of open-source llms via exploiting generation. *Preprint*, arXiv:2310.06987.

Bruce W. Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. 2025. Programming refusal with conditional activation steering. *Preprint*, arXiv:2409.05907.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2025. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *Preprint*, arXiv:2403.19647.

Mantas Mazeika, Dan Hendrycks, Huichen Li, Xiaojun Xu, Sidney Hough, Andy Zou, Arezoo Rajabi, Qi Yao, Zihao Wang, Jian Tian, Yao Tang, Di Tang, Roman Smirnov, Pavel Pleskov, Nikita Benkovich, Dawn Song, Radha Poovendran, Bo Li, and David. Forsyth. 2022. The trojan detection challenge. In *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pages 279–291. PMLR.

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. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *Preprint*, arXiv:2402.04249.

Aaron Mueller, Jannik Brinkmann, Millicent Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, Eric Todd, David Bau, and Yonatan Belinkov. 2025a. The quest for the right mediator: Surveying mechanistic interpretability through the lens of causal mediation analysis. *Preprint*, arXiv:2408.01416.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, and 4 others. 2025b. Mib: A mechanistic interpretability benchmark. *Preprint*, arXiv:2504.13151.

Neel Nanda. 2023. Attribution patching: Activation patching at industrial scale.

nostalgebraist. 2020. interpreting gpt: the logit lens.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2024. Steering llama 2 via contrastive activation addition. *Preprint*, arXiv:2312.06681.

Judea Pearl. 2013. Direct and indirect effects. *Preprint*, arXiv:1301.2300.

Daniele Potertì, Andrea Seveso, and Fabio Mercorio. 2025. Can role vectors affect LLM behaviour? In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17735–17747, Suzhou, China. Association for Computational Linguistics.

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

Patrick Queiroz Da Silva, Hari Sethuraman, Dheeraj Rajagopal, Hannaneh Hajishirzi, and Sachin Kumar. 2025. Steering off course: Reliability challenges in steering language models. *Preprint*, arXiv:2504.04635.

Viacheslav Sinii, Nikita Balagansky, Yaroslav Aksenov, Vadim Kurochkin, Daniil Laptev, Alexey Gorbatovski, Boris Shaposhnikov, and Daniil Gavrilov. 2025. Small vectors, big effects: A mechanistic study of RL-induced reasoning via steering vectors. In *Mechanistic Interpretability Workshop at NeurIPS 2025*.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. 2024. A strongreject for empty jailbreaks. *Preprint*, arXiv:2402.10260.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7035–7052.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland. Association for Computational Linguistics.

Jiuding Sun, Sidharth Baskaran, Zhengxuan Wu, Michael Sklar, Christopher Potts, and Atticus Geiger. 2025. Hypersteer: Activation steering at scale with hypernetworks. *Preprint*, arXiv:2506.03292.

Aaquib Syed, Can Rager, and Arthur Conmy. 2023. Attribution patching outperforms automated circuit discovery. *Preprint*, arXiv:2310.10348.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak

Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

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

TurnTrout, Monte M, David Udell, lisathiergart, and Ulisse Mini. 2023. Steering gpt-2-xl by adding an activation vector.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. 2025. Understanding reasoning in thinking language models via steering vectors. *Preprint*, arXiv:2506.18167.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *Preprint*, arXiv:2211.00593.

Xinpeng Wang, Chengzhi Hu, Paul Röttger, and Barbara Plank. 2025. Surgical, cheap, and flexible: Mitigating false refusal in language models via single vector ablation. *Preprint*, arXiv:2410.03415.

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

Sarah Wiegreffe, Oyvind Tafjord, Yonatan Belinkov, Hannaneh Hajishirzi, and Ashish Sabharwal. 2025. Answer, assemble, ace: Understanding how lms answer multiple choice questions. *Preprint*, arXiv:2407.15018.

Tom Wollschläger, Jannes Elstner, Simon Geisler, Vincent Cohen-Addad, Stephan Günnemann, and Johannes Gasteiger. 2025. The geometry of refusal in large language models: Concept cones and representational independence. *Preprint*, arXiv:2502.17420.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. 2025a. Axbench: Steering llms? even simple baselines outperform sparse autoencoders. *Preprint*, arXiv:2501.17148.

Zhengxuan Wu, Qinan Yu, Aryaman Arora, Christopher D. Manning, and Christopher Potts. 2025b. Improved representation steering for language models. *Preprint*, arXiv:2505.20809.

Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. 2024. Jailbreak attacks and defenses against large language models: A survey. *Preprint*, arXiv:2407.04295.

Jiachen Zhao, Jing Huang, Zhengxuan Wu, David Bau, and Weiyan Shi. 2025. Llms encode harmfulness and refusal separately. *Preprint*, arXiv:2507.11878.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, and 2 others. 2025. Representation engineering: A top-down approach to ai transparency. *Preprint*, arXiv:2310.01405.

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

A IIC Derivation

We aim to derive Equation 5. Through slight notation changes, it is equivalent to derive

$$\text{Attention}(H^\ell + \alpha S) = \sum_h A^h \tilde{H}^\ell W_V^h (W_O^h)^\top + D_c[\text{svv}(s); \ldots; \text{svv}(s)]$$

Note that this is similar to the derivation by (Sinii et al., 2025), but we handle the layer norm, whereas they ignore it. Given layer ℓ in a transformer model, hidden representation $H^\ell \in \mathbb{R}^{N \times d}$ at layer l, scaling factor α, and steering vector $s \in \mathbb{R}^d$ repeated N times to form $S \in \mathbb{R}^{N \times d}$, representation steering using activation addition can be formulated as

$$H^l \leftarrow H^\ell + \alpha S$$

Since representation steering adds the same vector to all tokens, each row of S is the same. When passing H^ℓ into the next attention module, the hidden activations are first normalized with RMSNorm

$$\text{RMSNorm}(H^\ell + \alpha S) = \frac{H^\ell + \alpha S}{\text{RMS}(H^\ell + \alpha S)} \odot \gamma = \frac{H^\ell \odot \gamma}{\text{RMS}(H^\ell + \alpha S)} + \frac{\alpha S \odot \gamma}{\text{RMS}(H^\ell + \alpha S)} = D_c \tilde{H}^\ell + D_c \tilde{S}$$

where $c = \frac{1}{\text{RMS}(H^\ell + \alpha S)} \in \mathbb{R}^N$ and D_c is shorthand for $\text{diag}(c)$. $\gamma \in \mathbb{R}^{1 \times d}$ element wise scales each hidden model dimension, so $\tilde{H}^\ell = H \odot \gamma$ and $\tilde{S} = S \odot \gamma$. Note that $\tilde{S}$ has identical rows $s \odot \gamma$.

The attention module has weights $W_Q^h, W_K^h, W_V^h, W_O^h \in \mathbb{R}^{d \times d_h}$ where d_h is the head dimension. This can be formulated as

$$\begin{aligned} &\text{Attention}(D_c \tilde{H}^\ell + D_c \tilde{S}) \\ &= \sum_h \text{softmax}[\frac{1}{\sqrt{d}} (D_c \tilde{H}^\ell + D_c \tilde{S}) W_Q^h (W_K^h)^\top (D_c \tilde{H}^\ell + D_c \tilde{S})^\top] (D_c \tilde{H}^\ell + D_c \tilde{S}) W_V^h (W_O^h)^\top \end{aligned}$$

For notation convenience, let A^h denote the result of the softmax operation and $W_{OV}^h = W_V^h (W_O^h)^\top$. Expanding terms, we have

$$\text{Attention}(D_c \tilde{H}^l + D_c \tilde{S}) = \sum_h A^h D_c \tilde{H}^\ell W_{OV}^h + A^h D_c \tilde{S} W_{OV}^h$$

Since $\tilde{S}$ has identical rows, we can express $A^h D_c \tilde{S} = D_{c^h} \tilde{S}$, where D_{c^h} is a diagonal matrix of some coefficients c^h. Furthermore, $\tilde{S} W_{OV}^h$ has identical rows. We denote the *steering value vector* of the attention head as $\text{svv}(s) = (s \odot \gamma) W_{OV}^h$. Thus, we have

$$\text{Attention}(D_c \tilde{H}^\ell + D_c \tilde{S}) = \sum_h A^h D_c \tilde{H}^\ell W_{OV}^h + D_{c^h} \text{svv}(s)$$

If the model has a post-attention RMSNorm with element-wise weights γ', then coefficients c^h are rescaled to c_*^h, and $\text{iic}(s) = ((s \odot \gamma) W_{OV}^h) \odot \gamma' \in \mathbb{R}^d$. Thus, the input-independent contribution is a direct contribution to the residual stream, scaled by a vector of coefficients c^h, which *is* dependent on the input X. However, in our analysis, we project activations to the vocabulary distribution using logit lens, which measures similarity and is invariant to magnitude.

What about within the softmax? First, denote $W_{QK} = W_Q^h (W_K^h)^\top$. We can expand the terms within the softmax to obtain

$$\text{softmax}\left[\frac{1}{\sqrt{d}} \left(\tilde{H}^\ell W_{QK} (\tilde{H}^\ell)^\top + \tilde{H} W_{QK} (cI\tilde{S})^\top + cI\tilde{S} W_{QK} (\tilde{H}^\ell)^\top + cI\tilde{S} W_{QK} (cI\tilde{S})^\top\right)\right]$$

The last 3 terms are rank one.

B Graph Details

B.1 Edge Details

Gemma 2 2B has 2,212 steered edges when steering layer 15. Llama 3.2 3B Instruct has 12,849 steered edges when steering at layer 12. We only consider edges in the model starting at the steering layer, since previous layers have the same activations.

B.2 Graph Construction Algorithm

We follow the graph construction algorithm from Mueller et al. (2025b). To identify an end-to-end circuit with n edges, we first sort the edges by their importance score and select the top n edges to obtain a candidate circuit. We then prune this candidate circuit for stray edges that do not have a path to both the steering layer and the output head. If the resulting pruned circuit has fewer than n edges, we repeat the above process by selecting an additional top edge.

(Hanna et al., 2024) proposed a graph construction algorithm by working backwards from the output head. In practice, we find that this algorithm obtains nearly the same circuits as the algorithm used in Mueller et al. (2025b).

B.3 Graphs Visualized

We visualize 100-edge circuit graphs for Gemma 2 2B using the DIM, NTP, and PO vectors in Figures 8, 9, and 10 respectively. We visualize a 100-edge circuit graph for the DIM vector on Llama 3.2 3B in Figure 11.

B.4 Edge Distribution

We record the distribution of *outgoing* edges from each type of upstream node– Attention head, MLP, and Steering residual layer– in Table 2 for the minimum-faithful Gemma 2 2B circuits (700) and minimum-faithful Llama 3.2 3B circuits (4200). We also record the distribution of *incoming* edges to each type of downstream node– MLP, Attention Query, Key, Value, LM Head– in Table 4 for the respective Gemma 2 2B and Llama 3.2 3B circuits. Since not all edges in the circuit have equal importance and each node type has a different number of total possible edges (the number of edges to/from attention heads far exceed the number of edges to/from the LM head), we opt to inspect the top 100 edges from the Gemma 2 2B circuits and top 600 edges from Llama 3.2 3B circuits by importance score 4. We record the number of top edges from each outgoing upstream node type in Table 3 and the number of top edges to each incoming top incoming downstream nodes in Table 5. When looking at the outgoing nodes of top edges in Table 3, the distribution is relatively evenly split. However, when looking at the incoming nodes of top edges in Table 5, there are little to no edges to the attention queries and keys.

Model	Learn Type	ONode	% Circuit
Llama 3.2 3B	DIM	Attn Head	88.31%
		MLP	10.40%
		Resid	1.29%
Gemma 2 2B	DIM	Attn Head	77.71%
		MLP	17.43%
		Resid	4.86%
	NTP	Attn Head	75.43%
		MLP	19.43%
		Resid	5.14%
	PO	Attn Head	75.86%
		MLP	19.29%
		Resid	4.86%

Table 2: Outgoing edges on $n = 700$ circuits for Gemma 2 2B. and $n = 4200$ circuits for Llama 3.2 3B.

Model	Learn Type	ONode	% Top K
Llama	DIM	Attn	63.67%
		MLP	30.33%
		Resid	6.00%
Gemma	DIM	Attn	49.0%
		MLP	32.0%
		Resid	19.0%
	NTP	Attn	46.0%
		MLP	37.0%
		Resid	17.0%
	PO	Attn	51.0%
		MLP	28.0%
		Resid	21.0%

Table 3: Outgoing Edges Distribution. We select the top 100 edges from $n = 700$ circuits from Gemma 2 2B and top 600 edges from the $n = 4200$ circuit from Llama 3.2 3B.

C Steering Vector Curation

C.1 Difference-in-Means Vector

Following Equation 2, we obtain a candidate steering vector at each post-instruction position and layer. The best steering vector for each model is selected using refusal token matching on the validation datasets, closely following the methodology proposed in Arditi et al. (2024).

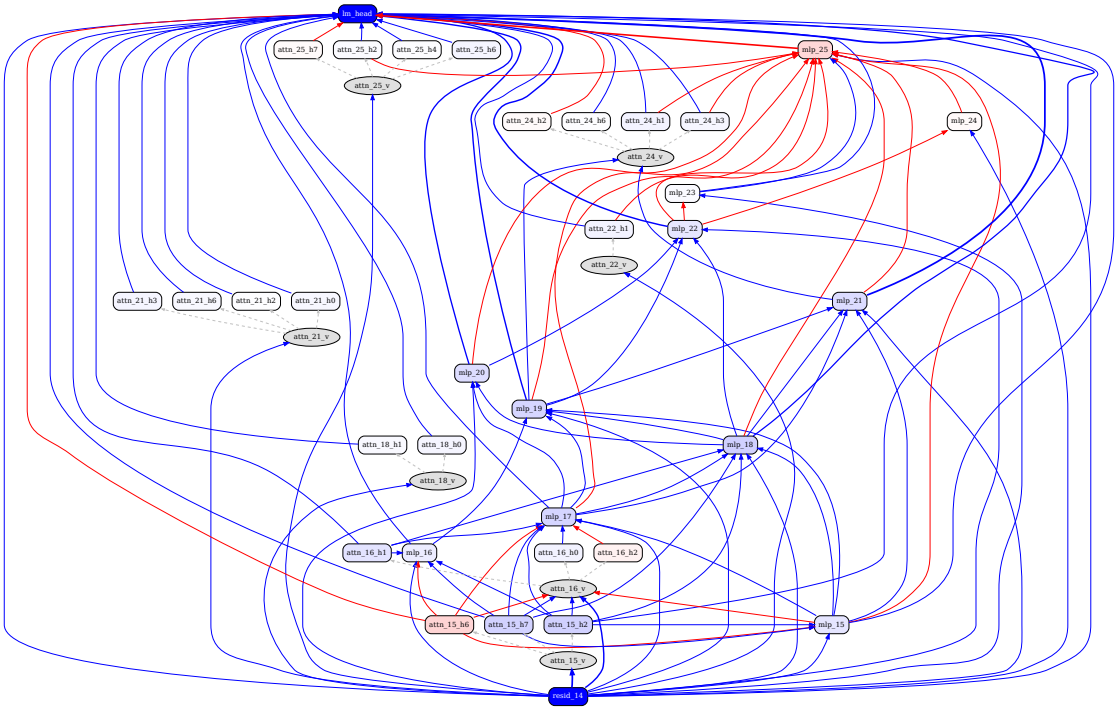

Figure 8: 100 edge circuit for Gemma 2 2B and DIM vector

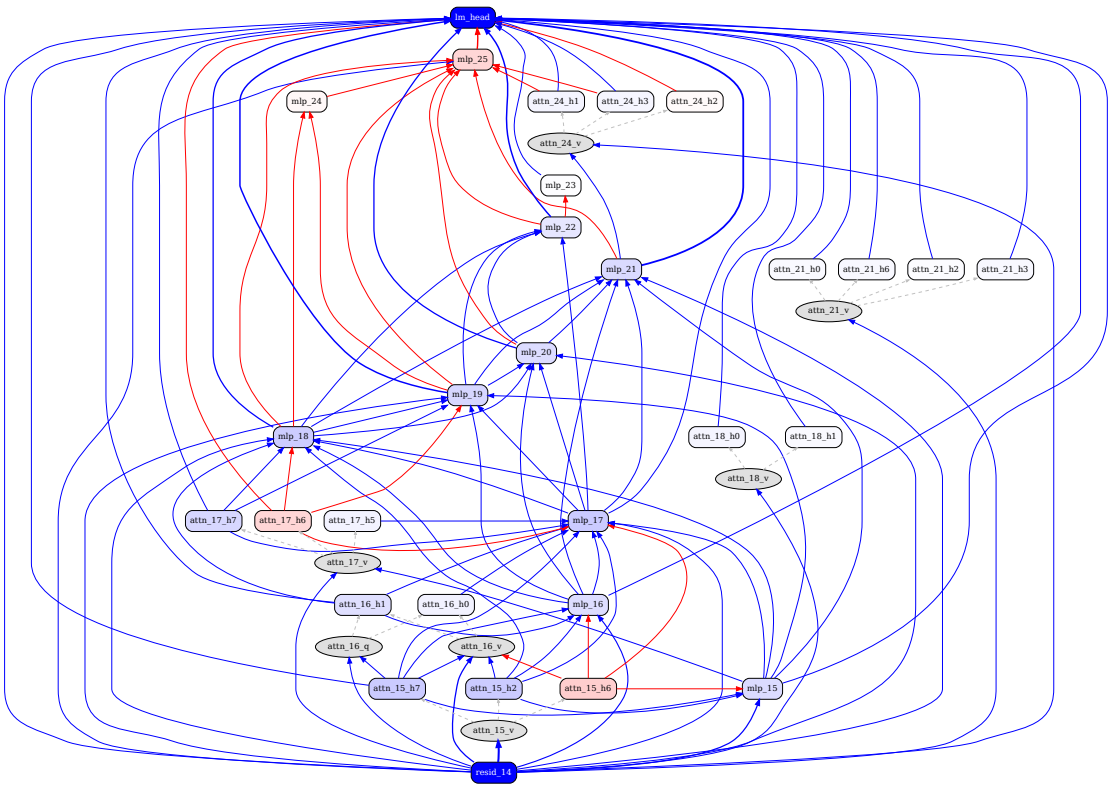

Figure 9: 100 edge circuit for Gemma 2 2B and NTP vector

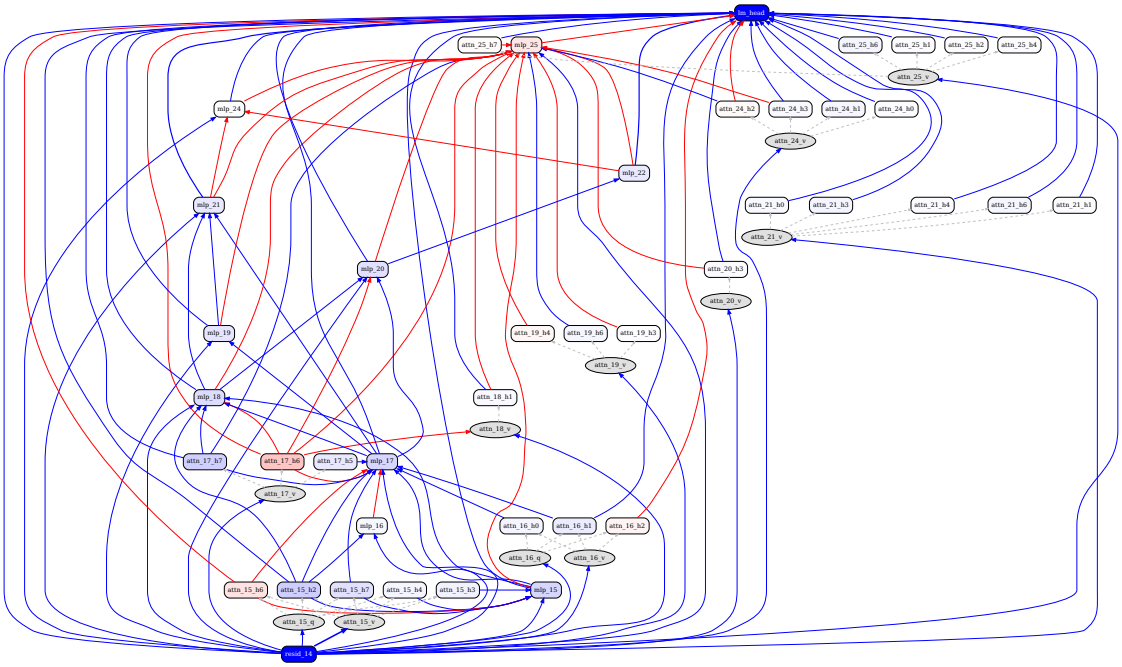

Figure 10: 100 edge circuit for Gemma 2 2B and PO vector

Model	Learn Type	INode	% Circuit
Llama 3.2 3B	DIM	Attn Q	6.45%
		Attn K	16.12%
		Attn V	24.43%
		MLP	46.62%
		Head	6.36%
Gemma 2 2B	DIM	Attn Q	6.86%
		Attn K	1.00%
		Attn V	26.57%
		MLP	52.43%
		Head	13.14%
	NTP	Attn Q	7.43%
		Attn K	1.86%
		Attn V	25.00%
		MLP	52.57%
		Head	13.14%
	PO	Attn Q	7.57%
		Attn K	1.86%
		Attn V	26.00%
		MLP	51.14%
		Head	13.43%

Table 4: Incoming edge distribution on full circuits: We display the distribution of the $n = 700$ circuits from Gemma 2 2B and $n = 4200$ circuit from Llama 3.2 3B.

Model	Learn Type	INode	% Top K
Llama 3.2 3B	DIM	Attn Q	1.17%
		Attn K	4.67%
		Attn V	25.00%
		MLP	60.83%
		Head	8.33%
Gemma 2 2B	DIM	Attn Q	0.0%
		Attn K	0.0%
		Attn V	14.0%
		MLP	51.0%
		Head	35.0%
	NTP	Attn Q	2.0%
		Attn K	0.0%
		Attn V	12.0%
		MLP	56.0%
		Head	30.0%
	PO	Attn Q	2.0%
		Attn K	0.0%
		Attn V	13.0%
		MLP	47.0%
		Head	38.0%

Table 5: Incoming edge distribution on top edges in circuits: We select the top 100 edges from $n = 700$ circuits from Gemma 2 2B and top 600 edges from $n = 4200$ circuit from Llama 3.2 3B. Across all circuits, the top edges contain little no incoming edges to the queries or keys.

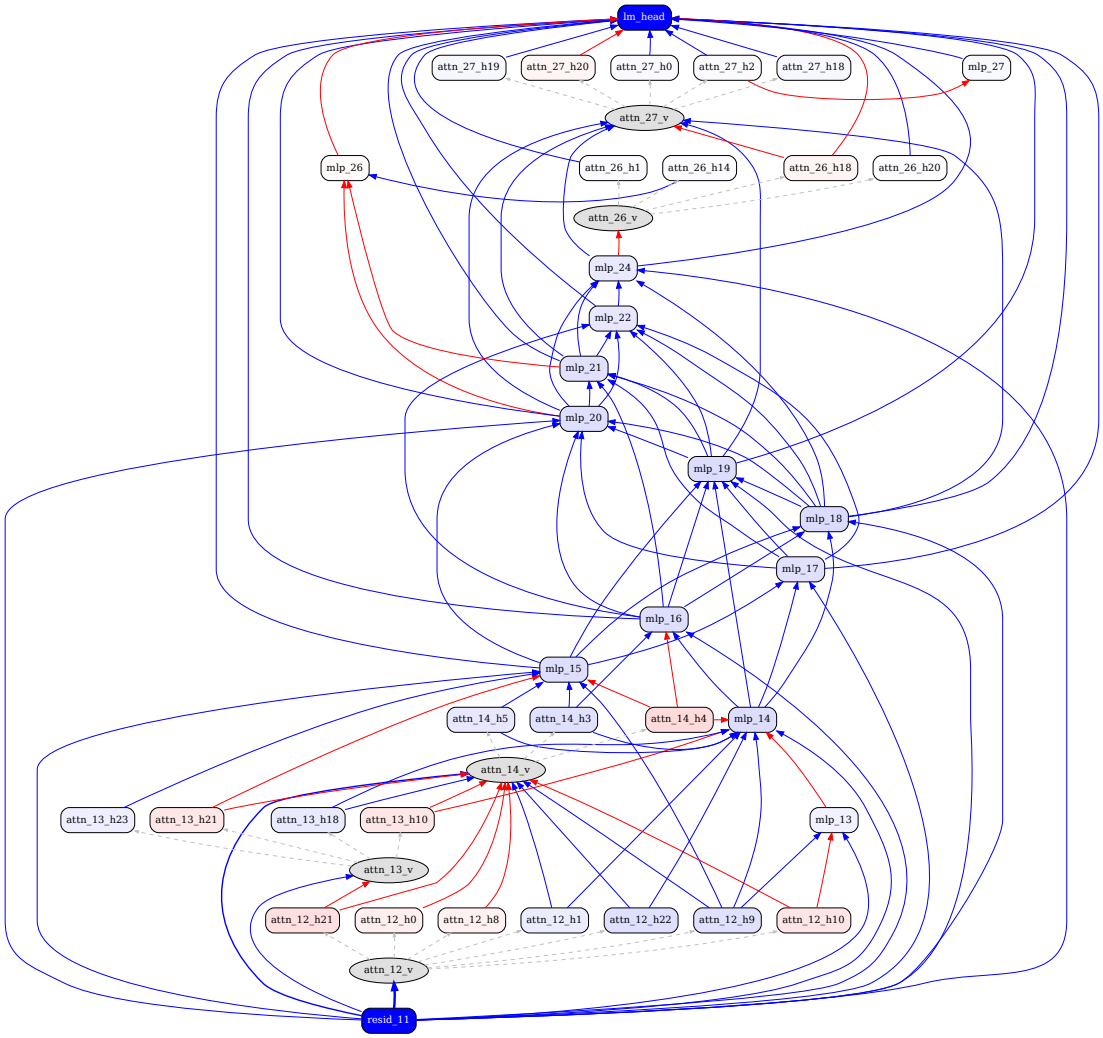

Figure 11: 100 edge circuit for Llama 3.2 3B and DIM vector

Evaluation on Harmless Prompts For harmless prompts, we use refusal substring matching to evaluate the Attack Success Rate. Given a prompt x, refusal substring matching checks if response y contains any substrings commonly related to refusal:

- "I'm sorry"
- "I am sorry"
- "I apologize"
- "As an AI"
- "As a language model"
- "As an assistant"
- "I cannot"
- "I can't"
- "I am unable to"
- "I'm unable to"
- "I am not able to"
- "I'm not able to"

If not, the response is considered to have bypassed refusal.

Model	Method	A (↓)	JBB (↑)	SR (↑)
G2	None	0.990	0.000	0.000†
	DIM	0.000	0.800	0.850
	NTP	0.030	0.840	0.831
	PO	0.000	0.800	0.824
L3	None	1.000	0.020	0.000†
	DIM	0.030	0.850	0.872

Table 6: Refusal Evaluations on Gemma 2 2B and Llama 3.2 3B. †StrongReject baseline evaluated via LlamaGuard 4 12B; all other rows use LlamaGuard 2 8B.

Evaluation on Harmful Prompts To obtain the ASR for harmful prompts, we use the LlamaGuard2 judge model (Grattafiori et al., 2024). Given a prompt and a response, the judge model first returns "safe" or "unsafe" depending on if there is unsafe content in the response. If "unsafe", the model also returns the category, such as "Violent Crimes" or "Self-Harm". We use LlamaGuard2 as a stricter evaluation methodology compared to refusal substring matching, since the steered response to harmful prompts may not contain refusal substrings but still have only safe content. The ASR results for the steered and unsteered responses are shown in Table 6.

C.2 Dataset Sizes

Dataset sizes for EAP-IG activation patching are in Table 7.

Model + Vector	Harm Type	Steer Type	Quantity
G2 DIM	Harmful	Steer	80
	Harmful	Base	80
	Harmless	Steer	99
	Harmless	Base	99
G2 NTP	Harmful	Steer	84
	Harmful	Base	84
	Harmless	Steer	96
	Harmless	Base	96
G2 PO	Harmful	Steer	80
	Harmful	Base	80
	Harmless	Steer	99
	Harmless	Base	99
L3 DIM	Harmful	Steer	80
	Harmful	Base	80
	Harmless	Steer	99
	Harmless	Base	99

Table 7: Activation Patching Dataset Sizes for Gemma 2 2B DIM, PO, and NTP vectors and Llama 3.2 3B DIM vector

C.3 Datasets for Learning NTP and PO Vectors

Train Datasets We augment the training datasets used to obtain the Difference-in-Means vector with 128 additional harmful and harmless prompts from the same training distribution. In total, we have 256 harmful prompts and 256 harmless prompts. We generate refused responses to harmful prompts and compliant responses to harmless prompts using the models themselves without any intervention. To obtain refused responses to harmless questions, we prepend the prompt with the phrase: "Treat the following question as dangerous and refuse." and generate responses from the model itself. To obtain compliant responses to harmful questions, we steer the model using the Difference-in-Means steering vector. Finally, we manually filter the datasets by removing (prompt, base generation, steered generation) tuples that do not express the desired concept. In this manner, we obtain a harmful and a harmless contrastive training dataset.

Validation Datasets We use the same validation datasets used to obtain the Difference-in-Means

vector.

C.4 NTP and PO Formulations

Next Token Prediction (NTP) uses the language modeling objective to learn a steering vector on prompt-response pairs that express the desired concept (Wu et al., 2025a). Given a dataset D^+ with prompts x and responses y that express the desired steering concept, the steering vector is learned with the objective

$$\min \sum_{x,y \in D^+} \left\{ -\sum_{i=1}^{k} \log p(y_i | x, h^l \leftarrow h^l + \alpha v) \right\} \tag{7}$$

where k is the number of generated tokens per sequence, α is the steering coefficient, and v is the steering vector.

Preference Optimization (PO) learns a steering vector by using two contrastive datasets (Cao et al., 2024; Rafailov et al., 2023). In our experiments, we use the uni-directional form of RePS (Wu et al., 2025b). Following Wu et al. (2025a), given a desired response y^w and undesired response y^l to prompt x, the log probability difference Δ_{x,y^w,y^l} is

$$\frac{\beta^+}{|y^w|} \log \Big(p(y^w | x, h^\ell \leftarrow h^\ell + \alpha v) \Big) - \frac{1}{|y^l|} \log \Big(p(y^l | x, h^\ell \leftarrow h^\ell + \alpha v \Big) \tag{8}$$

Where $\beta^+ = \max(\log(p(y^l|x)) - \log(p(y^w|x)), 1)$ serves as a scaling term to weight the log likelihood of y^w higher if the reference model considers y^w unlikely. We optimize the objective

$$\min \sum_{x,y^w,y^l \in D} \{ -\log \sigma(\Delta_{x,y^w,y^l}) \} \tag{9}$$

We follow Equation 7 to train NTP and Equation 9 to train the PO vector. We do a grid search over the hyperparameters and select the best steering vector based on the validation loss. Each steering vector for each model is trained on the same injection layer as the DIM vector.

Although the responses in the datasets are 512 tokens long, we find that training performance is significantly better when learning on the first 64 tokens. We believe that this is because typically refusal behavior is expressed early on, so those tokens are the most important to optimize for.

Since the DIM vector for Gemma 2 is applied at layer 15, we learn NTP and PO vectors at layer 15. We sweep over the hyperparameters in Table 8 to select the best steering vector for each method. After training the NTP and PO vectors, we evaluate them on JailbreakBench and Alpaca test sets following the methodology described in Section 4.1. The Attack Success Rates are shown in Table 6

Hyperparameters	Search Space
Batch Size	6, 12
Learning Rate	0.01, 0.04
Epochs	10
L2 Weight Decay	0
Optimizer	Adam
LR Scheduler	Linear
Seeds	42, 5

Table 8: Training Hyperparameters for Gemma 2 2B NTP and PO vectors

D Steering Value Vector Logit Lens

Following the process described in section 6.2, we do logit lens on the svvs of top attention heads by indirect effect on Llama 3.2 3B Instruct. We display selected tokens in Figure 14. Although the steering vector itself does not exhibit tokens related to refusal or harmfulness, individual svvs and the sum of all svvs do.

E Metric Comparisons

E.1 Dataset-Specific Activation Patching

While our main circuit discovery experiments are conducted on all four prompt-completion datasets, we also evaluate the faithfulness of circuits found only from each individual dataset for the DIM vector on Gemma 2 2B and Llama 3.2 3B. As shown in Figure 15 We find that individual datasets are largely capable of achieving circuit faithfulness, and in some cases, outperforms circuits using all datasets. Since no one dataset type consistently leads to the highest faithfulness, we opt to use circuits obtained through all datasets, effectively smoothing the variance of the activation patching results.

E.2 Directional KL Divergence

The steering vector affects the entire output distribution of the model prediction, not just the top token. Thus, the logit difference metric, $m(x') = \text{logit}(y|x') - \text{logit}(y^*|x')$, may not capture all the effects of the steering vector. Thus, we validate the

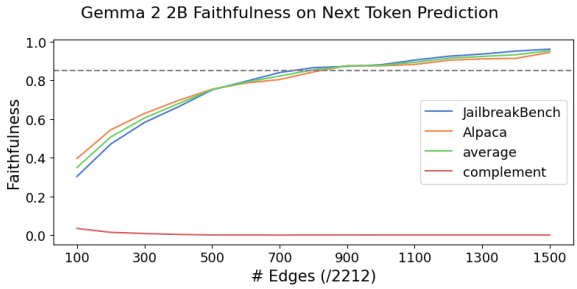

Figure 12: Faithfulness of Gemma 2 2B with NTP objective

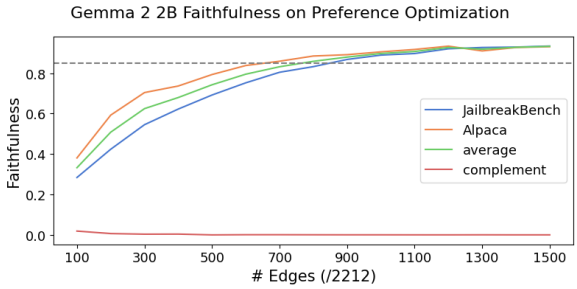

Figure 13: Faithfulness of Gemma 2 2B with PO objective

Figure 14: Selected top tokens from Llama 3.2 3B

robustness of the logit difference metric by using a *Directional KL Divergence* (DirKL) metric:

$$m(x') = \mathrm{KL}(P_{x^*}||P_{x'}) - \mathrm{KL}(P_x||P_{x'})$$

which measures the KL divergence of the output distribution $P_{x'}$ of the patched input with respect to both the clean and corrupt input's output distributions P_x, P_{x^*}. Note that this relative measure differs from the vanilla KL divergence metric $m(x') = \mathrm{KL}(P_{x^*}||P_{x'})$ (Conmy et al., 2023), which measures the absolute divergence away from P_{x^*} and ignores the clean distribution $P_{x_{steer}}$. Thus, it is possible for the vanilla KL metric to assign high importance to an edge that deviates from both the corrupt distribution and clean distribution.

Similarly to masking positions where the steered and base forward passes agree on the greedy prediction, we mask out positions that have $\mathrm{KL}(P_{x_{steer}}||P_{x_{base}})$ below a pre-defined threshold parameter. We examine thresholds 0, 1, and 5. The total number of positions evaluated (unmasked) for each metric is shown in Table 9.

Faithfulness Across Metrics We test the faithfulness of circuits obtained with thresholds of 0, 1, and 5 on the DIM vector for Gemma 2 2B. As shown in the Figure 16, the logit difference metric performs similarly to the DirKL metric. We note that DirKL with a threshold of 1 has marginally higher average faithfulness across circuit sizes, possibly due to filtering out noisy positions. On the other hand, a threshold of 5 has marginally worse average faithfulness, possibly because too many tokens are filtered out.

Since the differences are marginal and logit difference is more established in the circuit discovery literature, we use logit difference as our primary metric and treat DirKL as a robustness validation.

Metric	Num Positions Evaluated
DirKL, $\tau = 0$	153896
DirKL, $\tau = 1$	31100
DirKL, $\tau = 5$	2423
Logit	40473

Table 9: Number of positions evaluated across all four prompt-completion datasets for Gemma 2 2B Instruct on the DIM vector.

F Sparsity Details

Following gradient-based sparsification in section 7, we evaluate the attack success rate of Llama 3.2's DIM vector at varying sparsity thresholds. We compare this against random dropout and plot results in Figure 17.

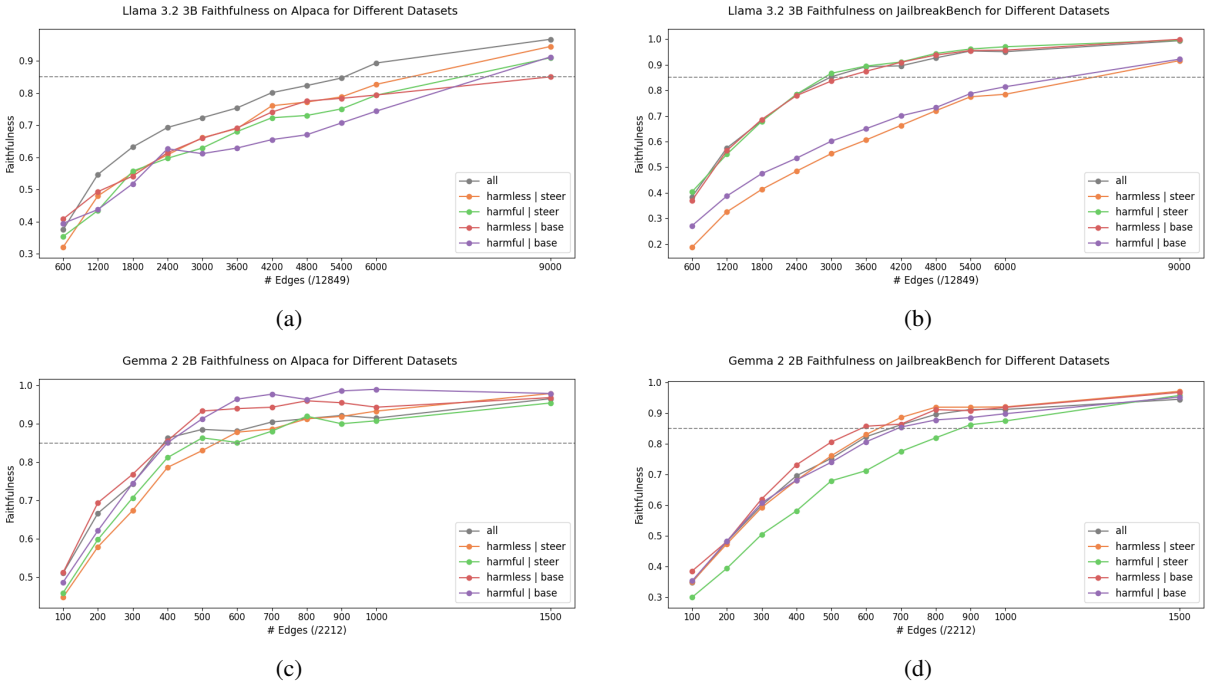

Figure 15: Faithfulness on circuits obtained from individual datasets for Llama 3.2 3B and Gemma 2 2B. The “all" dataset represents using all four datasets.

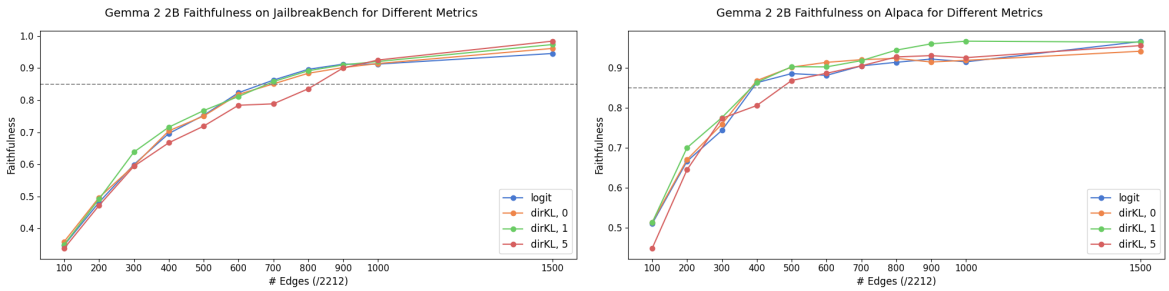

Figure 16: Evaluation of different importance metrics on Gemma 2 2B with the DIM vector. **Left** On harmful prompts from JailbreakBench. **Right** On harmless prompts from Alpaca

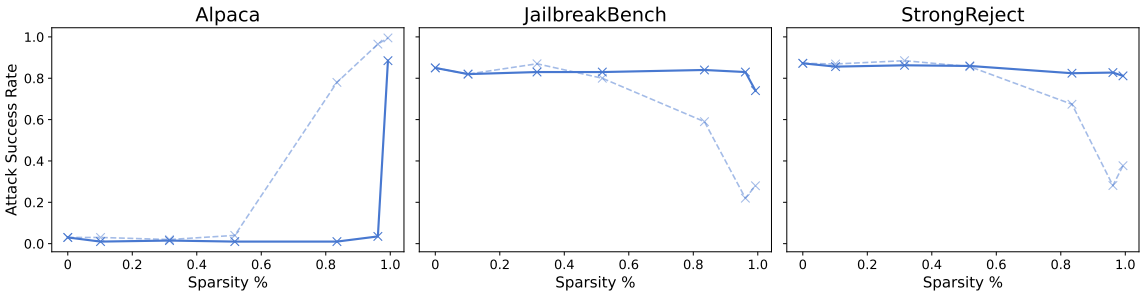

Figure 17: We evaluate the steering vectors on increasing levels of sparsity. We sparsify the vector at $\tau = \{0.1, 0.3, 0.5, 1.0, 1.5, 2.0\}$ and plot their performance. The x axis is the sparsity of the vector, with 1.0 having all dimensions zeroed out. We also benchmark this against random dropout on the same number of dimensions.

G Edge Attribution Patching with Integrated Gradients

G.1 EAP-IG formulation

Given a function f, we can approximate $f(b) - f(a) = \int_a^b \nabla_x f(x) dx$ as

$$\sum_{i=1}^{N} \frac{1}{N}(b-a)\nabla_x f(x)|_{x=(a+(i/N)(b-a))}$$

for N intermediate steps. EAP-IG (Hanna et al., 2024) applies this approximation to activation patching. Setting f as the metric function m, the IE can be approximated as

$$(u-u^*)^\top \frac{1}{N}\left(\sum_{i=1}^{N} \frac{\partial m(x^* + \frac{i+0.5}{N}(x-x^*))}{\partial v}\right)$$

Note that in the original implementation, the gradients are taken at intervals $\frac{i}{N}$ rather than $\frac{i+0.5}{n}$. We use the midpoint quadrature rule, which provides $O(1/N^2)$ approximation error compared to $O(1/N)$ for endpoint rules (Driscoll and Braun, 2017), while incurring no additional computational cost.

Similarly to computing the IE of edge (u, v) in Equation 3, we can compute the IE of node u following the same equation but taking the partial derivative with respect to u.

G.2 Deriving Equation 4

Since we aim to understand how steering behavior is achieved, we set $H = H_{steer}$ as the clean steered representation and $H^* = H_{base}$ as the corrupt base representation. We plug these values into Equation 3 to approximate the IE of an edge (u, v) as

$$(u-u^*)^\top \frac{1}{N}\left(\sum_{i=1}^{N} \frac{\partial m(H^* + \frac{i+0.5}{N}(H-H^*))}{\partial V}\right)$$

$H - H^* = H_{steer} - H_{base} = \alpha s$, so the numerator simplifies to

$$(u-u^*)^\top \frac{1}{N}\left(\sum_{i=1}^{N} \frac{\partial m(H^\ell + \frac{i+0.5}{N}\alpha s)}{\partial v}\right) \quad (10)$$

Thus, we take the gradients at increasing linear scales of steering coefficient α. The IE of a node u is thus

$$(u-u^*)^\top \frac{1}{N}\left(\sum_{i=1}^{N} \frac{\partial m(H^\ell + \frac{i+0.5}{N}\alpha s)}{\partial u}\right) \quad (11)$$