Steer2Edit: From Activation Steering to Component-Level Editing

Chung-En Sun **Ge Yan*** **Zimo Wang*** **Tsui-Wei Weng**
University of California, San Diego
{cesun, geyan, zimowang, lweng}@ucsd.edu

Abstract

Steering methods influence Large Language Model behavior by identifying semantic directions in hidden representations, and are typically realized through inference-time activation interventions that apply a fixed, global modification to the model's internal states. While effective, such interventions often induce unfavorable attribute–utility trade-offs under strong control, as they ignore the fact that many behaviors are governed by a small and heterogeneous subset of model components. To alleviate the trade-offs, we propose **STEER2EDIT**, a theoretically grounded, training-free framework that transforms steering vectors from inference-time control signals into diagnostic signals for component-level rank-1 weight editing. Instead of uniformly injecting a steering direction during generation, **STEER2EDIT** selectively redistributes behavioral influence across individual attention heads and MLP neurons, yielding interpretable edits that preserve the standard forward pass and remain compatible with optimized parallel inference. Across multiple tasks including safety alignment, truthfulness promotion, and reasoning efficiency, **STEER2EDIT** consistently achieves a superior attribute–utility trade-offs: at matched downstream performance, it outperforms the strongest steering baseline by 17.2% in safety, 9.8% in truthfulness, and 12.2% in reasoning length reduction. Overall, **STEER2EDIT** provides a principled bridge between representation steering and weight editing by translating steering signals into *interpretable*, *training-free* parameter updates. Our code is available at: https://github.com/Trustworthy-ML-Lab/Steer2Edit

1 Introduction

Large Language Models (LLMs) have demonstrated strong capabilities across a wide range of tasks, including multi-step reasoning [Guo et al., 2025, Sun et al., 2025c], code generation [Chen, 2021], and planning [Yao et al., 2023]. As these models are increasingly deployed in real-world settings, there is growing interest in *controlling* specific model behaviors without retraining or fully fine-tuning the model.

A prominent line of recent work addresses this goal through *representation steering* [Zou et al., 2023a, Turner et al., 2023, Arditi et al., 2024, Yan et al., 2025, Li et al., 2025]. These methods identify a *steering vector* in the model's hidden representation space that correlates with a target attribute, and then intervene at inference time by adding this vector to intermediate activations. Compared to full fine-tuning, steering-based methods offer a lightweight way to adapt a model to different behaviors.

Despite their flexibility, activation-space steering methods suffer from two fundamental limitations. *First*, steering applies a *global* modification to the hidden representation. While such interventions can induce the target behavior, they treat all tokens and internal components uniformly, regardless

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

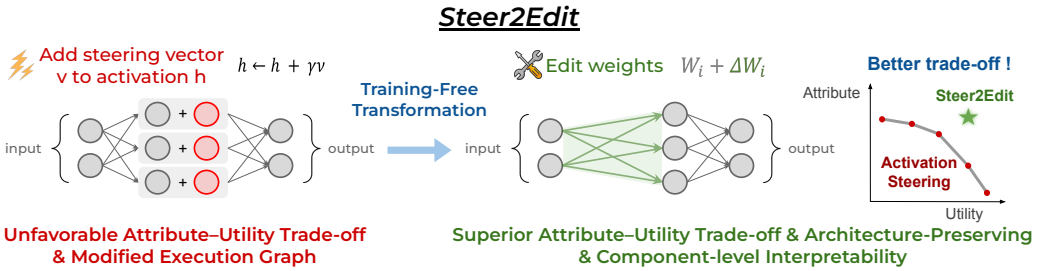


Figure 1: Overview of STEER2EDIT. Instead of changing intermediate activations during inference (left), **STEER2EDIT** transforms the steering signal into weight edits (right), achieving better attribute–utility trade-offs including *safety*, *truthfulness*, and *efficient reasoning* settings as shown in Sec 4.

of how the behavior is realized within the model. Empirical and mechanistic studies show that many behaviors are governed by a small and heterogeneous subset of model components [Olsson et al., 2022, Meng et al., 2022, Zhao et al., 2025, Zhou et al., 2024, Chu et al., 2025], typically involving specific attention heads or MLP neurons, while most components are unrelated to the target attribute. By ignoring this internal structure, global steering can interfere with unrelated features, resulting in unfavorable trade-offs between the controlled attribute and downstream performance. *Second*, activation-space steering relies on inference-time modification of intermediate activations. This departs from the standard forward pass assumed by modern optimized inference and training systems, which typically require fixed computation graphs. As a result, activation-level interventions complicate integration with standard deployment, parallel inference, and fine-tuning pipelines. While this limitation can in principle be mitigated with additional system engineering, activation steering remains an inference-time control mechanism whose effects are tied to the decoding process, rather than being encoded in the model parameters.

These limitations motivate a different perspective: instead of treating steering vectors as control signals injected into the forward pass, can we reinterpret them as *diagnostic signals* that reveal how a target behavior is distributed across model components? If so, can this information be used to selectively modify the components that genuinely govern the behavior, while avoiding unnecessary interference that degrades utility?

To answer these questions, we introduce **STEER2EDIT**, a theoretically grounded framework that converts steering vectors into *component-level weight edits*. In **STEER2EDIT**, a steering vector is treated as a diagnostic signal that reveals which attention heads and MLP neurons align with a target behavior, and to what extent. Guided by this signal, the method applies coordinated rank-1 updates to individual components, selectively amplifying or suppressing their contributions along the steering direction, rather than inducing a global activation shift. By redistributing behavioral influence at the component level, **STEER2EDIT** enables more precise behavioral control and more favorable attribute–utility trade-offs, while yielding interpretable component-level edits. The resulting procedure is closed-form, requires no fine-tuning or iterative optimization, and produces a standalone edited model that operates under the standard forward pass and remains compatible with existing parallel training and inference pipelines.

Contributions.

- We propose **STEER2EDIT**, the first theoretically grounded framework that translates steering vectors into component-level rank-1 weight edits, requiring no fine-tuning and admitting a closed-form, single-step solution.
- We show that **STEER2EDIT** consistently achieves a superior attribute–utility trade-off across diverse behavioral control settings: at matched downstream performance, it outperforms the strongest steering baseline by **17.2%** in safety, **9.8%** in truthfulness, and **12.2%** in reasoning length reduction on average.
- We show that **STEER2EDIT** produces a standalone edited model that preserves the original architecture, while offering fine-grained interpretability into which components govern specific behaviors and how these behaviors are distributed across the network.

Table 1: Comparison of behavioral control methods for LLMs.

Method	Training-free	Superior attribute–utility trade-off	Zero inference overhead	Standard forward pass	Component interpretability
RepEngineering [Zou et al., 2023a]	✓	✗	✗	✗	✗
CAA [Panickssery et al., 2023]	✓	✗	✗	✗	✗
ODESteer [Zhao et al., 2026]	✗	✗	✗	✗	✗
AlphaEdit [Fang et al., 2024]	✓	✗	✓	✓	✗
SFT / LoRA	✗	✗	✓	✓	✗
STEER2EDIT (Ours)	✓	✓	✓	✓	✓

2 Preliminary

In this section, we define the steering vectors used throughout and specify the editable weight components targeted by **STEER2EDIT**.

Steering vector. Steering vectors are commonly used in activation steering, where a semantic direction in hidden representations is added to the residual stream at inference time to control model behavior. Such vectors can be constructed in various ways. For simplicity, we adopt a mean-difference construction in this work.

We consider a standard pre-normalization Transformer with L layers and residual-stream dimension d. At each layer ℓ, the attention and MLP blocks each produce an additive update $\delta_\ell^b \in \mathbb{R}^d$, $b \in \{\text{attn}, \text{mlp}\}$, that is written into the residual stream. Let $\mathcal{X}$ denote a set of prompts. For each prompt $x \in \mathcal{X}$, the model generates a completion y, which is classified as exhibiting or not exhibiting the target attribute, yielding $\mathcal{Y}_{\text{pos}}$ and $\mathcal{Y}_{\text{neg}}$. At token position t of y, let $\delta_\ell^b(y,t) \in \mathbb{R}^d$ denote the block-b output at layer ℓ before it is added to the residual stream. Aggregating over token positions $\mathcal{T}_y$ and averaging over generations, we define

$$\overline{\delta}_{\ell,a}^b = \frac{1}{|\mathcal{Y}_a|} \sum_{y \in \mathcal{Y}_a} \frac{1}{|\mathcal{T}_y|} \sum_{t \in \mathcal{T}_y} \delta_\ell^b(y,t), \quad a \in \{\text{pos}, \text{neg}\}.$$

The steering vector at layer ℓ and block b is given by the mean difference

$$v_\ell^b = \overline{\delta}_{\ell,\text{pos}}^b - \overline{\delta}_{\ell,\text{neg}}^b \in \mathbb{R}^d. \tag{1}$$

Editable weight components and notation. We focus on linear weight components whose outputs produce the block activations from which the steering vectors in Eq.(1) are extracted. Specifically, for each layer ℓ and block type $b \in \{\text{attn}, \text{mlp}\}$, we consider linear maps whose outputs contribute to the block output δ_ℓ^b *before* it is written into the residual stream.

Concretely, these editable components include: (i) the output projection (o_proj) of an individual attention head in the attention block, and (ii) the down-projection ($down_proj$) associated with a single neuron in the MLP block. We denote any such component generically by

$$W_i \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}, \qquad d_{\text{out}} = d,$$

where the index i implicitly identifies a specific layer ℓ, block type b, and component.

For an input activation $h_i \in \mathbb{R}^{d_{\text{in}}}$ to component W_i, the component output $W_i h_i$ lies in the same residual-stream space $\mathbb{R}^d$ as the corresponding steering vector v_ℓ^b. Accordingly, in the subsequent analysis we associate each editable component W_i with the steering vector extracted from the same layer and block, and write this vector simply as v_i.

3 Steer2Edit

In this section, we introduce **STEER2EDIT**, a principled framework for component-level weight editing based on given steering vectors. We parameterize each edit as a rank-1 update and derive its form by decomposing the problem into three parts: (i) identifying the output-space direction that preserves semantic invariance, (ii) the input-space direction that aligns the edit with the component's intrinsic semantic contribution, and (iii) the scalar magnitude that allocates edit strength under a global regularization budget.

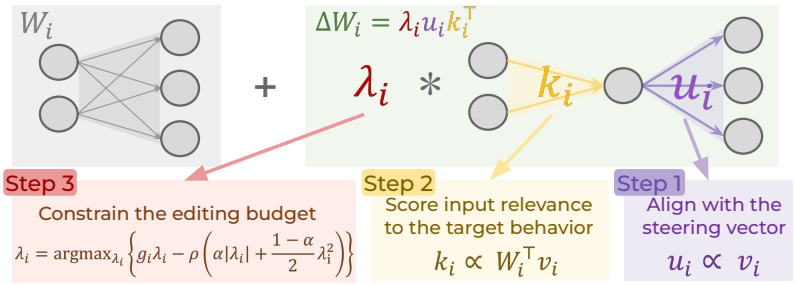


Figure 2: Closed-form derivation of STEER2EDIT. Each component receives a rank-1 update $\Delta W_i = \lambda_i u_i k_i^\top$ determined in three steps: (1) the output direction u_i is selected to ensure edits act along the target behavioral axis; (2) the input direction k_i is derived to activate the edit selectively on relevant inputs; (3) the magnitude λ_i is allocated to balance sparsity and edit strength across all components. All three quantities admit closed-form solutions, making the full procedure training-free.

3.1 Assumption and Setting

For each editable component $W_i \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we assume the existence of a steering vector $v_i \in \mathbb{R}^{d_{\text{out}}}$ extracted from the same representation space into which W_i writes (e.g., the hidden state after an attention or MLP block). Thus, the output dimension of W_i matches that of v_i, and both lie in a common semantic space. Note that **STEER2EDIT** is agnostic to how v_i is obtained.

Our goal is to modify each component W_i so that the resulting update ΔW_i alters the model's behavior along the semantic direction v_i. Because the steering signal specifies a single direction in representation space, we model each edit as a rank-1 perturbation, which is the minimal modification that can inject a directional effect into a linear map. Accordingly, we parameterize the edit as

$$\Delta W_i = \lambda_i \, u_i k_i^\top,$$

where $u_i \in \mathbb{R}^{d_{\text{out}}}$ is an output-space direction, $k_i \in \mathbb{R}^{d_{\text{in}}}$ is an input-space direction, and $\lambda_i \in \mathbb{R}$ is a scalar magnitude controlling the strength of the edit.

Given an input activation h_i, let $o_i := W_i h_i \in \mathbb{R}^{d_{\text{out}}}$ denote the original output of component W_i. After applying the rank-1 edit $\Delta W_i = \lambda_i u_i k_i^\top$, the edited output is $\tilde{o}_i := (W_i + \Delta W_i) h_i = o_i + \Delta o_i$, where the induced output shift is

$$\Delta o_i := \Delta W_i h_i = \lambda_i \, u_i (k_i^\top h_i). \tag{2}$$

The three quantities (u_i, k_i, λ_i) play distinct roles in the edit. The output-space direction u_i specifies the semantic direction affected by the edit, the input-space direction k_i determines which inputs activate the edit through the inner product $k_i^\top h_i$, and the scalar magnitude λ_i controls how strongly each component is modified.

We derive these quantities in a sequential order. We first determine u_i, then k_i, and finally solve for $\{\lambda_i\}_{i=1}^n$. As shown in the following sections, this ordering is without loss of generality: the optimal choice of u_i depends only on the steering vector v_i; the choice of k_i depends on u_i and local properties of the component W_i; and once the geometric directions are fixed, the magnitudes $\{\lambda_i\}$ can be optimized independently.

Hence, we derive the three components of the edit in the following order:

1. the output-space direction u_i in Section 3.2;
2. the input-space direction k_i in Section 3.3;
3. the scalar magnitude λ_i in Section 3.4.

Throughout the following derivations, we identify only the directions of u_i and k_i; their scale and sign are absorbed into the scalar coefficients λ_i, which are determined in the last step.

3.2 Step 1: Solving for the Output-space Direction u_i

The vector u_i determines the *direction* of the output shift Δo_i. Because the steering signal specifies a single semantic direction v_i, we require that the edit modifies the component's output *only* along this direction and introduces no change in any orthogonal subspace.

Formally, recall that $\Delta o_i = \Delta W_i h_i$. Semantic invariance requires that, for any input h_i, the output shift Δo_i has zero projection onto any direction orthogonal to v_i:

$$z^\top \Delta o_i = z^\top \Delta W_i h_i = 0 \qquad \forall\, h_i,\ \forall\, z \perp v_i. \tag{3}$$

This constraint directly restricts the output-space direction u_i of the edit.

Theorem 3.1 (Output-space direction under semantic invariance)**.** *Let $v_i \neq 0$, and let $\Delta W_i = \lambda_i\, u_i k_i^\top$ be a rank-1 edit with $\Delta W_i \neq 0$. If for all h_i and all $z \perp v_i$ we have*

$$z^\top \Delta W_i h_i = 0,$$

then the output-space direction u_i must be collinear with v_i, i.e.,

$$u_i \in \mathrm{span}\{v_i\}.$$

We defer the proof to Appendix B.1. Theorem 3.1 shows that enforcing semantic invariance in Eq.(3) uniquely constrains the output-space direction: any valid rank-1 update must lie entirely along the steering direction v_i. Importantly, this result is independent of both the input-space direction k_i and the scalar magnitude λ_i. This separation justifies solving for u_i first in our derivation. We therefore adopt the canonical normalized choice

$$u_i = \hat{v}_i := \frac{v_i}{\|v_i\|_2},$$

with the sign and scale absorbed later into the magnitude λ_i.

3.3 Step 2: Solving for the Input-space Direction k_i

Having solved the output-space direction $u_i = \hat{v}_i$, we now determine the input-space direction k_i for each editable component W_i. As before, we solve for the *direction* of k_i; its sign and scale are absorbed into the scalar magnitude λ_i.

Intuition. As shown in Eq. (2), the input-space direction k_i determines which input activations h_i trigger the edit through the inner product $k_i^\top h_i$. To identify a suitable k_i, we note that a well-trained component W_i already encodes which inputs are relevant for contributing to the semantic direction v_i, and the edit ΔW_i should mirror this existing input-dependent pattern.

To formalize this intuition, we define the *semantic alignment score* of W_i for an input h_i as

$$s_i(h_i) := v_i^\top o_i = v_i^\top W_i h_i,$$

which measures how strongly the original component output aligns with the semantic direction v_i for a given input h_i. Intuitively, if $s_i(h_i)$ is small across inputs, this indicates that the component is generally unrelated to the v_i, and the edit should be small for all inputs similarly. If, for some components, $s_i(h_i)$ is large for certain inputs, the edit should be large on those same inputs.

Hence, we choose k_i so that the induced change in the semantic alignment score, $\Delta s_i(h_i) := v_i^\top \Delta W_i h_i$, occurs on the same inputs for which $s_i(h_i)$ is large. To formalize this idea, we maximize the "absolute" Pearson correlation between $\Delta s_i(h_i)$ and $s_i(h_i)$, as we do not care about the sign or overall scale at this stage. The following theorem provides the solution.

Theorem 3.2 (Input-space direction matching semantic alignment variation)**.** *Fix a component W_i and set $u_i = \hat{v}_i$. Assume $W_i^\top v_i \neq 0$ and $\mathrm{Var}(s_i(h_i)) > 0$, where $s_i(h_i) := v_i^\top W_i h_i$. Consider choosing an input-direction $k_i \neq 0$ so that the induced semantic alignment shift $\Delta s_i(h_i) := v_i^\top \Delta W_i h_i$ exhibits maximal co-variation with the component's intrinsic semantic alignment score $s_i(h_i)$. Formally, consider the objective:*

$$\max_{k_i \neq 0} \Big|\, \mathrm{Pearson}\big(\Delta s_i(h_i),\ s_i(h_i)\big) \Big|.$$

Then there exists a maximizer k_i that is collinear with $W_i^\top v_i$, i.e.,

$$k_i \in \mathrm{span}\{W_i^\top v_i\}.$$

We defer the proof to Appendix B.2. Theorem 3.2 shows that the input-space direction k_i should align with the component's intrinsic input sensitivity $W_i^\top v_i$. We therefore adopt the normalized choice

$$\hat{k}_i := \frac{W_i^\top v_i}{\|W_i^\top v_i\|_2}.$$

This choice is further empirically validated in Appendix G.

3.4 Step 3: Solving for the Edit Magnitudes $\boldsymbol{\lambda}$

With the edit directions u_i and k_i fixed, we now determine the magnitudes $\{\lambda_i\}$, which control how strongly each component is reinforced or suppressed. Intuitively, the magnitude assigned to each component should reflect how that component contributes to the direction v_i *on average across inputs*: components that consistently align with v_i should be reinforced, components that consistently oppose it should be suppressed, and components with weak alignment should receive little or no edit.

Note that this role is fundamentally different from that of the input-space direction k_i, which captures how the component's semantic alignment score $s_i(h_i)$ *varies across inputs*, whereas the magnitudes λ_i depend only on the component's *averaged* semantic alignment.

To formalize this allocation of editing strength, we now introduce an importance weighting for each component and derive λ_i via a global regularized optimization.

Importance weighting. Recall that the *semantic alignment score* $s_i(h_i) := v_i^\top W_i h_i$ measures how strongly component W_i contributes to the semantic direction v_i for a given input h_i. Since the magnitudes λ_i are intended to capture a component's *overall* semantic contribution, we measure this contribution by the expectation of the semantic alignment score over the input distribution: $\mathbb{E}[s_i(h_i)] = \mathbb{E}[v_i^\top W_i h_i]$. However, the typical output magnitude of $W_i h_i$ can vary substantially across transformer layers, making raw values not directly comparable between components. To place components on a common footing, we make the following changes: (i) remove the arbitrary scale of the semantic direction by using $\hat{v}_i = v_i / \|v_i\|_2$, and (ii) normalize by the output norm of the mean activation $\mu_i = \mathbb{E}[h_i]$, yielding

$$g_i = \frac{\mathbb{E}[s_i(h_i)]}{\|v_i\|_2 \, \|W_i \mu_i\|_2} = \frac{v_i^\top W_i \mu_i}{\|v_i\|_2 \, \|W_i \mu_i\|_2} = \cos(v_i, W_i \mu_i),$$

which we refer to as the *component importance score*. The sign of g_i indicates whether the component aligns or opposes the semantic direction, while $|g_i|$ measures the strength of this tendency. This normalization choice is empirically validated in Appendix G.

Elastic-Net objective. The component importance score g_i indicates whether component W_i should be reinforced or suppressed, and with what strength. A natural objective is therefore to maximize total alignment $\boldsymbol{g}^\top \boldsymbol{\lambda} = \sum_{i=1}^n g_i \lambda_i$. However, this objective is unbounded, and effective weight editing should remain lightweight by modifying only a small number of relevant components while keeping edit magnitudes controlled.

We address both considerations with an Elastic-Net regularization, combining an ℓ_1 term to promote sparsity and an ℓ_2 term to limit overall edit size:

$$\max_{\boldsymbol{\lambda}} \; \boldsymbol{g}^\top \boldsymbol{\lambda} - \rho \left(\alpha \|\boldsymbol{\lambda}\|_1 + \frac{1-\alpha}{2} \|\boldsymbol{\lambda}\|_2^2 \right),$$

where $\rho > 0$ controls the global edit budget and $\alpha \in [0, 1)$ trades off ℓ_1 sparsity and ℓ_2 smoothness. Ablation results in Appendix G confirm the importance of both the ℓ_1 and ℓ_2 regularization terms.

Theorem 3.3 (Edit magnitude allocation under regularization)**.** *For each component W_i, let $g_i = \cos(v_i, W_i \mu_i)$ denote the component importance score, with the convention that $g_i := 0$ if $W_i \mu_i = 0$. Consider the problem of assigning edit magnitudes $\{\lambda_i\}$ to maximize total signed alignment as measured by the component importance scores $\{g_i\}$, while controlling both the sparsity and overall strength of the edit. Formally, let $\boldsymbol{g} = (g_1, \ldots, g_n)$ and consider*

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^n} \; \boldsymbol{g}^\top \boldsymbol{\lambda} - \rho \left(\alpha \|\boldsymbol{\lambda}\|_1 + \frac{1-\alpha}{2} \|\boldsymbol{\lambda}\|_2^2 \right), \rho > 0, \; \alpha \in [0, 1).$$

The unique edit magnitude assigned to component i is

$$\lambda_i^* = \operatorname{sign}(g_i) \, \frac{\max(|g_i| - \rho\alpha, \, 0)}{\rho(1-\alpha)}.$$

We defer the proof to Appendix B.3. Theorem 3.3 yields a closed-form soft-thresholding rule for allocating edit magnitudes λ_i according to the alignment scores g_i under a global Elastic-Net budget. Substituting λ_i^*, together with $u_i = \hat{v}_i$ and $k_i = \hat{k}_i$, gives the unified weight-editing update below.

3.5 Summary: Unified Weight Editing Rule

Each editable component W_i receives the rank-1 update

$$\boxed{\Delta W_i = \operatorname{sign}(g_i)\,\frac{\max(|g_i| - \rho\alpha,\, 0)}{\rho(1-\alpha)}\;\hat{v}_i\,\hat{k}_i^\top,}$$

where $\hat{v}_i := \frac{v_i}{\|v_i\|_2}$, $\hat{k}_i = \frac{W_i^\top v_i}{\|W_i^\top v_i\|_2}$, $g_i = \cos(v_i,\, W_i\mu_i)$. Each update is:

- **Directionally selective:** it modifies only the projection along the semantic direction v_i;
- **Input-selective:** it determines which inputs should trigger the edit based on the component's intrinsic semantic behavior;
- **Budget-aware:** its magnitude λ_i^* is determined by the component importance score g_i under an Elastic-Net regularizer.

Thus, **STEER2EDIT** yields a component-level weight editing framework that jointly captures *what* semantic direction to modify, *when* the edit should be activated by the input, and *how strongly* each component should be adjusted.

4 Experiments

This section evaluates whether **STEER2EDIT** achieves a superior *attribute–utility trade-off* compared to inference-time activation steering. We consider three representative behavioral control settings: (i) safety alignment against jailbreak attacks, (ii) truthfulness promotion, and (iii) reasoning efficiency control. Across all settings, we compare **STEER2EDIT** against three steering baselines:

Steering Baselines. We compare against three activation steering methods. **RepEngineering** [Zou et al., 2023a] extracts block-level steering vectors $\{v_\ell^{\text{attn}}, v_\ell^{\text{mlp}}\}_{\ell=1}^{L}$ at every layer and shifts each block output as $h_\ell \leftarrow h_\ell + \gamma v_\ell$ during inference; we sweep γ to trace trade-off curves. **CAA** [Panickssery et al., 2023] adds a single residual-stream direction at layer $\lfloor L/3 \rfloor$. **ODESteer** [Zhao et al., 2026] applies adaptive ODE-based steering via a learned barrier function at the same layer. The construction of steering vectors for each baseline is deferred to Appendix D.

Steer2Edit. Following Section 3, we apply rank-1 edits to linear components that write directly into the residual stream. For attention, we edit each head's output projection $W_o \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$. For MLPs, we edit individual down-projection neurons by treating each column $w_{\text{down},j}$ of $W_{\text{down}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ as an independent component. Edit magnitudes are determined by an Elastic-Net objective with sparsity parameter α and global budgets ρ_{attn} and ρ_{mlp} for attention heads and MLP neurons. For each setting, hyperparameters $(\rho_{\text{attn}}, \rho_{\text{mlp}}, \alpha)$ are selected via a small grid search on a held-out validation set, ranking configurations by the target attribute metric. Unless otherwise noted, results correspond to the best-performing configurations. Full search details are provided in Appendix E.

4.1 Evaluation for Behavioral Control

We consider three settings that examine behavioral control along distinct dimensions: safety alignment, truthfulness, and reasoning efficiency. In each setting, the *target attribute* is measured using task-specific metrics, while *downstream utility* is evaluated on task-oriented benchmarks that are unrelated to the controlled behavior. We visualize the trade-off between the target attribute and utility.

4.1.1 Safety Alignment Against Jailbreak Attacks

Goal. We evaluate whether **STEER2EDIT** can strengthen refusal behavior under strong jailbreak attacks while preserving helpfulness on benign tasks.

Models and evaluation. We evaluate safety alignment on **LLaMA-2-7B-Chat** and **Mistral-7B-Instruct-v0.2**. Safety is measured using **ADVBench**, which consists of harmful user queries designed to elicit unsafe behavior. Each query is transformed into a jailbreak prompt using either **GCG** [Zou et al., 2023b], a classical gradient-based attack, or **ADV-LLM** [Sun et al., 2025a], a substantially stronger attack that trains an LLM to generate adversarial suffixes. We report the **refusal rate**,

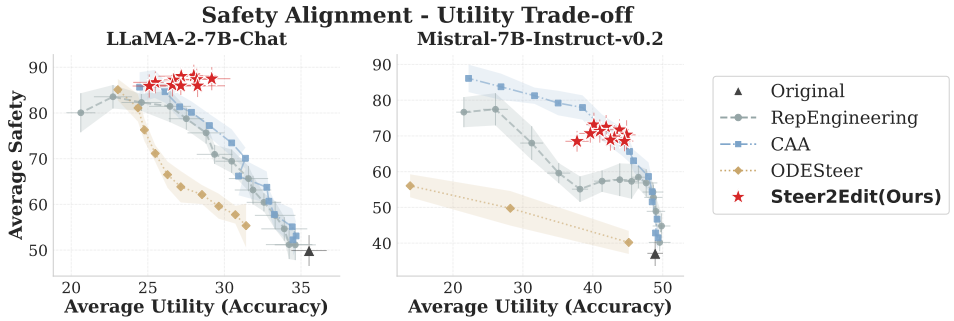


Figure 3: Safety–utility trade-off on **LLaMA-2-7B-Chat** and **Mistral-7B-Instruct-v0.2**. Each curve traces one steering baseline; each star is one **STEER2EDIT** configuration. **STEER2EDIT** consistently occupies the upper-right region, achieving higher refusal rates at comparable or higher utility.

defined as the proportion of model responses that are refusals, averaged across both attack types. Downstream utility is evaluated on **GSM8K**, **CodeMMLU**, and **CommonsenseQA**, measuring utility on grade-school math reasoning, programming, and commonsense multiple-choice questions. Utility is reported as the mean accuracy across all three benchmarks.

Results. Figure 3 illustrates the safety–utility relationship. Overall, **STEER2EDIT** lies beyond the trade-off frontier of all baselines, occupying the upper-right region of the plot. Notably, only CAA on Mistral-7B reaches a trade-off region comparable to **STEER2EDIT**; however, as we will later show in Sections 4.1.2 and 4.1.3, this competitiveness does not generalize to the truthfulness or reasoning efficiency settings, where CAA exhibits much weaker performance.

4.1.2 Truthfulness Promotion

Goal. We evaluate whether **STEER2EDIT** increases the model's preference for truthful answers while preserving performance on unrelated downstream tasks.

Models and evaluation. We evaluate on **Gemma-2-2B-IT** and **LLaMA-3-8B-Instruct** using **TruthfulQA**. For each prompt, we measure whether the model assigns higher probability to the truthful answer than to a plausible but false alternative, and report truthful preference accuracy. Downstream utility is again measured on GSM8K, CodeMMLU, and CommonsenseQA.

Results. Figure 4 shows the truthfulness–utility relationship. **STEER2EDIT** achieves substantially higher truthfulness at comparable utility, dominating the Pareto frontier of all three baselines.

4.1.3 Efficient Reasoning

Goal. We evaluate whether **STEER2EDIT** can shorten reasoning traces while preserving answer accuracy, improving inference efficiency for Large Reasoning Models (LRMs).

Models and datasets. We evaluate on **Qwen3-4B-Thinking-2507** and **OpenMath-Nemotron-7B** using GSM8K, MATH-500, GPQA, and CodeMMLU. Downstream utility is measured as mean

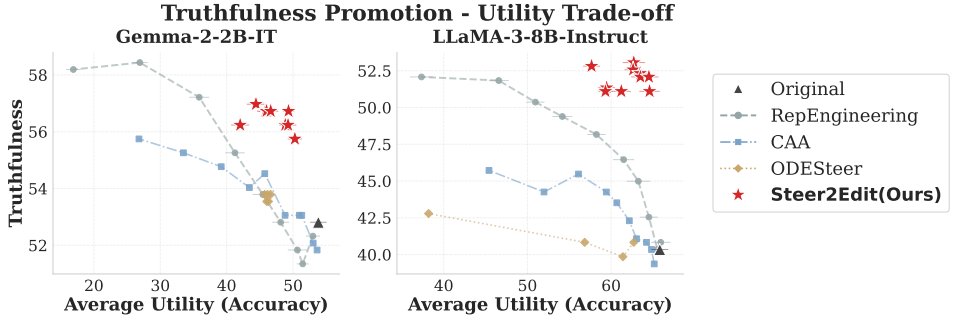


Figure 4: Truthfulness–utility trade-off on **Gemma-2-2B-IT** and **LLaMA-3-8B-Instruct**. **STEER2EDIT** improves truthfulness while maintaining higher utility than all baselines.

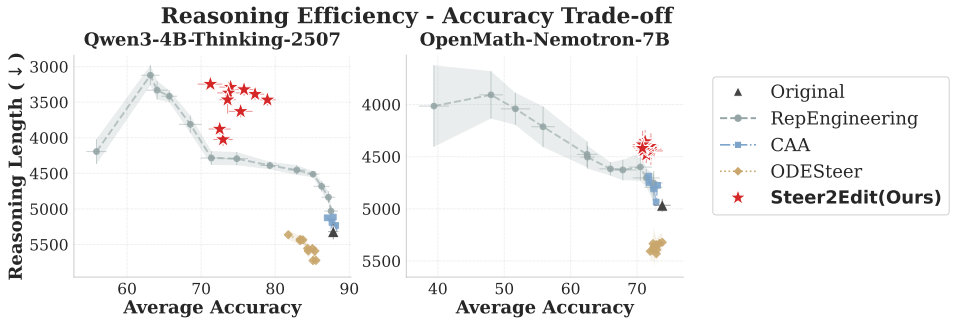


Figure 5: Accuracy–efficiency trade-off on **Qwen3-4B-Thinking-2507** and **OpenMath-Nemotron-7B**. **STEER2EDIT** achieves shorter reasoning traces at comparable accuracy

accuracy across all datasets, while reasoning efficiency is measured by the number of generated reasoning tokens.

Results. Figure 5 shows the accuracy–efficiency relationship. CAA and ODESteer produce almost no length reduction, consistent with the component analysis in the next section 4.2 showing that reasoning efficiency is governed by distributed MLP neurons across many layers, limiting single-layer methods. **STEER2EDIT** significantly shortens reasoning traces while maintaining comparable accuracy by distributing edits across the full network.

4.2 Component-Level Interpretability

Because **STEER2EDIT** applies edits at the level of individual attention heads and MLP neurons, the resulting weight updates are directly interpretable and reveal which components mediate each target behavior. We visualize the signed edit coefficients λ for the best-performing configuration in each setting (Appendix C). A clear structural difference emerges: **safety alignment and truthfulness** are predominantly mediated by sparse edits to a small number of attention heads, while **reasoning efficiency** is governed by dense, distributed edits to MLP neurons across many layers. This structural divergence confirms that different behavioral properties are encoded in distinct component types, and that STEER2EDIT's component-level decomposition captures this structure automatically.

4.3 Additional Experiments

For completeness, we report the following additional experiments in the Appendix:

- **Per-dataset trade-offs** (Appendix F): Trade-off curves for each evaluation dataset.
- **Design-choice ablations** (Appendix G): Ablations of the input direction, importance score, and edit magnitude.
- **Component-wise budget sensitivity** (Appendix H): Sensitivity to attention and MLP editing budgets.
- **Training-based baselines** (Appendix I): Comparisons with SFT and LoRA.
- **Weight-editing baseline** (Appendix J): Comparison with AlphaEdit [Fang et al., 2024].
- **Reasoning-compression baseline** (Appendix K): Comparison with ASC [Azizi et al., 2025].
- **Over-refusal evaluation** (Appendix L): Evaluation on OR-Bench [Cui et al., 2024].
- **Scaling to 70B models** (Appendix M): Evaluation on larger language models.
- **Vision-language models** (Appendix N): Generalization to VLMs.
- **Diffusion language models** (Appendix O): Generalization to diffusion LLMs.
- **Fine-grained steering baselines** (Appendix P): Comparisons with ITI and AUSteer.
- **Labeling-judge robustness** (Appendix Q): Sensitivity to the truthfulness labeling judge.

5 Related Works

Steering and controlling LLM behavior. A growing body of work studies behavioral control in LLMs via interventions on internal representations. Activation steering methods [Zou et al., 2023a, Turner et al., 2023, Arditi et al., 2024, Yan et al., 2025, Li et al., 2025, Panickssery et al., 2023, Zhao et al., 2026, Sun et al., 2026] extract semantic directions from contrastive examples and apply inference-time activation interventions to modulate attributes such as safety or reasoning behavior. These approaches are training-free, but rely on global, inference-time modifications. In parallel, Concept Bottleneck Models [Sun et al., 2025b, 2024, Bidusa and Markovitch, 2025, Tan et al., 2023, Ludan et al., 2023] introduce explicit architectural constraints to enable human-interpretable control. Together, these lines of work demonstrate that LLM behavior can be influenced through manipulation of internal representations.

In contrast, **STEER2EDIT** translates steering signals into *component-level* edits that operate at the level of individual components. By redistributing behavioral influence across attention heads and MLP neurons, **STEER2EDIT** enables more favorable attribute–utility trade-offs, while providing fine-grained interpretability and preserving the standard model architecture.

LLM weight editing. Another line of work focuses on modifying model parameters to induce persistent behavioral changes without full retraining. Representative approaches include meta-editors such as MEND [Mitchell et al., 2021] and KnowledgeEditor [De Cao et al., 2021], mechanistic editors such as ROME [Meng et al., 2022] and MEMIT [Meng et al., 2023], neuron-level interventions [Dai et al., 2022], semi-parametric methods such as SERAC [Mitchell et al., 2022], and null-space constrained editing such as AlphaEdit [Fang et al., 2024]. More recent work applies targeted weight edits to specific components for behavior control, including ThinkEdit [Sun et al., 2025d] for improving reasoning quality and CBSE [Chu et al., 2025, 2026] for safety. However, unlike **STEER2EDIT**, these methods are largely empirical and do not provide a unified framework for allocating and justifying edits across components.

6 Conclusion

We introduced **STEER2EDIT**, a theoretically grounded framework that translates steering signals into component-level weight edits via a closed-form solution. By shifting behavioral control from inference-time activation intervention to parameter updates, **STEER2EDIT** achieves more favorable behavior–utility trade-offs while preserving the standard model architecture. Beyond empirical gains, the method offers fine-grained interpretability, revealing how safety, truthfulness, and reasoning efficiency are distributed across attention heads and MLP components.

Acknowledgement

C. Sun, G. Yan, and T.-W. Weng are partially supported by National Science Foundation under Grant No. 2313105, 2430539, and Intel Rising Star Faculty Award. The authors would like to thank the National Research Platform (NRP) at UCSD for computational support and anonymous reviewers for valuable feedback.

References

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. *NeurIPS*, 2024.

Seyedarmin Azizi, Erfan Baghaei Potraghloo, and Massoud Pedram. Activation steering for chain-of-thought compression. *arXiv preprint arXiv:2507.04742*, 2025.

Or Raphael Bidusa and Shaul Markovitch. Concept layers: Enhancing interpretability and intervenability via llm conceptualization. *arXiv preprint arXiv:2502.13632*, 2025.

Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Kuan Lin Chu, Chung-En Sun, and Tsui-Wei Weng. How to make llms safer? detecting and editing key heads in llms. In *NeurIPS Lock-LLM Workshop*, 2025.

Kuan-Lin Chu, Chung-En Sun, and Tsui-Wei Weng. From detection to refusal: Safer llms via circuit-guided weight scaling. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 34657–34680, 2026.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. Or-bench: An over-refusal benchmark for large language models. *arXiv preprint arXiv:2405.20947*, 2024.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *ACL*, 2022.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*, 2021.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*, 2024.

Zijian Feng, Tianjiao Li, Zixiao Zhu, Hanzhang Zhou, Junlang Qian, Li Zhang, Jia Jim Deryl Chua, Lee Onn Mak, Gee Wah Ng, and Kezhi Mao. Fine-grained activation steering: Steering less, achieving more. In *International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=guSVafqhrB`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, 2023. URL `https://arxiv.org/abs/2306.03341`.

Yongce Li, Chung-En Sun, and Tsui-Wei Weng. Effective skill unlearning through intervention and abstention. *NAACL*, 2025.

Josh Magnus Ludan, Qing Lyu, Yue Yang, Liam Dugan, Mark Yatskar, and Chris Callison-Burch. Interpretable-by-design text classification with iteratively generated concept bottleneck. *arXiv preprint*, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *NeurIPS*, 2022.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *ICLR*, 2023.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *ICML*, 2022.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2023.

Chung-En Sun, Tuomas Oikarinen, and Tsui-Wei Weng. Crafting large language models for enhanced interpretability. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024.

Chung-En Sun, Xiaodong Liu, Weiwei Yang, Tsui-Wei Weng, Hao Cheng, Aidan San, Michel Galley, and Jianfeng Gao. Iterative self-tuning llms for enhanced jailbreaking capabilities. In *NAACL*, 2025a.

Chung-En Sun, Tuomas Oikarinen, Berk Ustun, and Tsui-Wei Weng. Concept bottleneck large language models. In *ICLR*, 2025b.

Chung-En Sun, Ge Yan, Akshay Kulkarni, and Tsui-Wei Weng. Refine: A framework for trustworthy large reasoning models with reliability, faithfulness, and interpretability. *arXiv preprint arXiv:2510.09062*, 2025c.

Chung-En Sun, Ge Yan, and Tsui-Wei Weng. Thinkedit: Interpretable weight editing to mitigate overly short thinking in reasoning models. *EMNLP*, 2025d.

Chung-En Sun, Linbo Liu, Ge Yan, Zimo Wang, and Tsui-Wei Weng. Llm agents already know when to call tools–even without reasoning. *arXiv preprint arXiv:2605.09252*, 2026.

Zhen Tan, Lu Cheng, Song Wang, Yuan Bo, Jundong Li, and Huan Liu. Interpreting pretrained language models via concept bottlenecks. *arXiv preprint*, 2023.

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

Ge Yan, Chung-En Sun, et al. Reflctrl: Controlling llm reflection via representation engineering. *arXiv preprint arXiv:2512.13979*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Hongjue Zhao, Haosen Sun, Jiangtao Kong, Xiaochang Li, Qineng Wang, Liwei Jiang, Qi Zhu, Tarek Abdelzaher, Yejin Choi, Manling Li, et al. Odesteer: A unified ode-based steering framework for llm alignment. *arXiv preprint arXiv:2602.17560*, 2026.

Yiran Zhao, Wenxuan Zhang, Yuxi Xie, Anirudh Goyal, Kenji Kawaguchi, and Michael Shieh. Understanding and enhancing safety mechanisms of llms via safety-specific neuron. In *ICLR*, 2025.

Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, Kun Wang, Yang Liu, Junfeng Fang, and Yongbin Li. On the role of attention heads in large language model safety. *arXiv preprint arXiv:2410.13708*, 2024.

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, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency. *CoRR*, 2023a.

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

Table of Contents

A Broader Impact **14**

B Proofs for STEER2EDIT **14**
B.1 Proof of Theorem 3.1 14
B.2 Proof of Theorem 3.2 14
B.3 Proof of Theorem 3.3 15

C Component-Level Edit Visualizations **17**

D Steering Vector Construction **18**

E Hyperparameter Search Procedure **19**

F Per-Dataset Trade-off Analysis **20**
F.1 Safety Alignment: Attack- and Dataset-Specific Trade-offs 20
F.2 Truthfulness: Dataset-Specific Utility Trade-offs 20
F.3 Efficient Reasoning: Dataset-Level Accuracy–Length Trade-offs 20

G Ablation Study of STEER2EDIT: Empirical Justification of Formal Design Choices **23**
G.1 Unified Performance Summary 23
G.2 Detailed Per-Setting Results 24

H Component-Wise Budget Sensitivity Analysis **26**
H.1 Safety Alignment: Sensitivity to Attention Budget 26
H.2 Truthfulness: Sensitivity Dominated by Attention 26
H.3 Efficient Reasoning: Sensitivity Dominated by MLP Budget 27

I Additional Baselines: Comparing STEER2EDIT with Training-Based Methods **28**
I.1 Safety Alignment 28
I.2 Truthfulness 28
I.3 Efficient Reasoning 29

J Additional Baselines: Comparing STEER2EDIT with AlphaEdit (Null-Space Weight Editing) **30**

K Additional Baselines: Comparing STEER2EDIT with Activation-Steered Compression (ASC) **31**

L OR-Bench Over-Refusal Evaluation **32**

M Scaling to 70B Larger LLMs **33**

N Scaling to Vision-Language Models **34**

O Scaling to Diffusion LLMs **35**

P Comparison with Fine-Grained Activation Steering **36**

Q Robustness to the Truthfulness Labeling Judge **37**

A Broader Impact

This paper presents work whose goal is to advance the field of machine learning by enabling efficient, interpretable, and training-free model editing at the level of individual components. In positive applications, STEER2EDIT may help practitioners correct or reduce undesirable behaviors (e.g., unsafe responses or hallucinations) and better understand which internal components support a given behavior, which can improve transparency and facilitate auditing. At the same time, weight-editing methods are inherently dual-use: the same capability can be applied to remove safeguards, amplify biases, or otherwise manipulate model behavior for harmful purposes, and edited models may be redistributed without clear provenance. To mitigate these risks, we emphasize that edits should be evaluated across diverse safety and capability tests, that releases should include clear documentation of intended use and limitations, and that responsible access controls may be appropriate for edits that materially alter safety-critical behaviors. Overall, we believe the primary societal consequence of this work is enabling more controllable and inspectable models, with corresponding responsibility to prevent and detect misuse.

B Proofs for STEER2EDIT

B.1 Proof of Theorem 3.1

Theorem B.1 (Output-space direction under semantic invariance)**.** *Let $v_i \neq 0$, and let $\Delta W_i = \lambda_i\, u_i k_i^\top$ be a rank-1 edit with $\Delta W_i \neq 0$. If for all h_i and all $z \perp v_i$ we have*

$$z^\top \Delta W_i h_i = 0,$$

then the output-space direction u_i must be collinear with v_i, i.e.,

$$u_i \in \operatorname{span}\{v_i\}.$$

Proof. Substituting $\Delta W_i = \lambda_i\, u_i k_i^\top$,

$$z^\top \Delta W_i h_i = \lambda_i\, (z^\top u_i)\, (k_i^\top h_i).$$

Because $\Delta W_i \neq 0$, we have $k_i \neq 0$, and therefore there exists some h_i such that $k_i^\top h_i \neq 0$. For the expression above to vanish for all such h_i, we must have $z^\top u_i = 0$ for every vector z orthogonal to v_i. The only vectors satisfying this condition are those proportional to v_i. Hence $u_i \in \operatorname{span}\{v_i\}$. □

B.2 Proof of Theorem 3.2

Theorem B.2 (Input-space direction matching semantic alignment variation)**.** *Fix a component W_i and set $u_i = \hat{v}_i$. Assume $W_i^\top v_i \neq 0$ and $\operatorname{Var}(s_i(h_i)) > 0$, where $s_i(h_i) := v_i^\top W_i h_i$. Consider choosing an input-direction $k_i \neq 0$ so that the induced semantic alignment shift $\Delta s_i(h_i) := v_i^\top \Delta W_i h_i$ exhibits maximal co-variation with the component's intrinsic semantic alignment score $s_i(h_i)$. Formally, consider the objective*

$$\max_{k_i \neq 0} \Big|\, \operatorname{Pearson}\big(\Delta s_i(h_i),\ s_i(h_i)\big) \Big|.$$

Then there exists a maximizer k_i that is collinear with $W_i^\top v_i$, i.e.,

$$k_i \in \operatorname{span}\{W_i^\top v_i\}.$$

Proof. Recall that $s_i(h_i) := v_i^\top W_i h_i$ and $\Delta s_i(h_i) := v_i^\top \Delta W_i h_i$. Using $\Delta W_i = \lambda_i \hat{v}_i k_i^\top$, we have

$$\Delta s_i(h_i) = \lambda_i \|v_i\|_2\, (k_i^\top h_i).$$

Pearson correlation is invariant to additive shifts in either argument, so it is unchanged if we center the inputs. Let $\tilde{h}_i = h_i - \mu_i$ with $\mu_i = \mathbb{E}[h_i]$, and define

$$\widetilde{\Delta s}_i(h_i) := \Delta s_i(\tilde{h}_i), \qquad \tilde{s}_i(h_i) := s_i(\tilde{h}_i).$$

Denote the covariance matrix by

$$\Sigma_i := \mathbb{E}[\tilde{h}_i \tilde{h}_i^\top].$$

Step 1: covariance. We have

$$\widetilde{\Delta s}_i(h_i) = \lambda_i \|v_i\|_2 \, (k_i^\top \tilde{h}_i), \qquad \tilde{s}_i(h_i) = v_i^\top W_i \tilde{h}_i.$$

Hence

$$\begin{aligned}
\mathrm{Cov}(\widetilde{\Delta s}_i, \tilde{s}_i) &= \mathbb{E}[\widetilde{\Delta s}_i(h_i)\, \tilde{s}_i(h_i)] \\
&= \lambda_i \|v_i\|_2 \, \mathbb{E}[(k_i^\top \tilde{h}_i)(v_i^\top W_i \tilde{h}_i)] \\
&= \lambda_i \|v_i\|_2 \, k_i^\top \, \mathbb{E}[\tilde{h}_i \tilde{h}_i^\top] \, W_i^\top v_i \\
&= \lambda_i \|v_i\|_2 \, k_i^\top \Sigma_i W_i^\top v_i.
\end{aligned}$$

Step 2: variances. Similarly,

$$\mathrm{Var}(\widetilde{\Delta s}_i) = \mathbb{E}[\widetilde{\Delta s}_i(h_i)^2] = \lambda_i^2 \|v_i\|_2^2 \, k_i^\top \Sigma_i k_i,$$

and

$$\mathrm{Var}(\tilde{s}_i) = \mathbb{E}[\tilde{s}_i(h_i)^2] = v_i^\top W_i \Sigma_i W_i^\top v_i.$$

Step 3: Pearson correlation. The Pearson correlation between the induced and intrinsic semantic signals is

$$\mathrm{Pearson}(\widetilde{\Delta s}_i, \tilde{s}_i) = \frac{\mathrm{Cov}(\widetilde{\Delta s}_i, \tilde{s}_i)}{\sqrt{\mathrm{Var}(\widetilde{\Delta s}_i)}\,\sqrt{\mathrm{Var}(\tilde{s}_i)}}.$$

Substituting the expressions above gives

$$\begin{aligned}
\mathrm{Pearson}(\widetilde{\Delta s}_i, \tilde{s}_i) &= \frac{\lambda_i \|v_i\|_2 \, k_i^\top \Sigma_i W_i^\top v_i}{\sqrt{\lambda_i^2 \|v_i\|_2^2 \, k_i^\top \Sigma_i k_i}\,\sqrt{v_i^\top W_i \Sigma_i W_i^\top v_i}} \\
&= \frac{\mathrm{sign}(\lambda_i)\, k_i^\top \Sigma_i W_i^\top v_i}{\sqrt{k_i^\top \Sigma_i k_i}\,\sqrt{v_i^\top W_i \Sigma_i W_i^\top v_i}}.
\end{aligned}$$

The denominator's second factor is independent of k_i, and $\mathrm{sign}(\lambda_i)$ is irrelevant when maximizing absolute correlation. Thus maximizing $|\,\mathrm{Pearson}(\widetilde{\Delta s}_i, \tilde{s}_i)|$ over $k_i \neq 0$ reduces to maximizing

$$\frac{|k_i^\top \Sigma_i W_i^\top v_i|}{\sqrt{k_i^\top \Sigma_i k_i}}. \tag{1}$$

Step 4: Cauchy–Schwarz in the Σ_i-inner product. Define the Σ_i-inner product:

$$\langle a, b \rangle_{\Sigma_i} := a^\top \Sigma_i b, \qquad \|a\|_{\Sigma_i} := \sqrt{a^\top \Sigma_i a}.$$

Then (1) becomes

$$\frac{|\langle k_i, W_i^\top v_i \rangle_{\Sigma_i}|}{\|k_i\|_{\Sigma_i}}.$$

By Cauchy–Schwarz,

$$|\langle k_i, W_i^\top v_i \rangle_{\Sigma_i}| \leq \|k_i\|_{\Sigma_i} \, \|W_i^\top v_i\|_{\Sigma_i}.$$

Equality is attained by choosing $k_i \propto W_i^\top v_i$. Therefore, there exists an optimizer $k_i \in \mathrm{span}\{W_i^\top v_i\}$. □

B.3 Proof of Theorem 3.3

Theorem B.3 (Edit magnitude allocation under regularization)**.** *For each component W_i, let $g_i = \cos(v_i, W_i \mu_i)$ denote the component importance score, with the convention that $g_i := 0$ if $W_i \mu_i = 0$.*

Consider the problem of assigning edit magnitudes $\{\lambda_i\}$ to maximize total signed alignment as measured by the component importance scores $\{g_i\}$, while controlling both the sparsity and overall strength of the edit. Formally, let $\boldsymbol{g} = (g_1, \ldots, g_n)$ and consider

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^n} \; \boldsymbol{g}^\top \boldsymbol{\lambda} - \rho \left(\alpha \|\boldsymbol{\lambda}\|_1 + \frac{1-\alpha}{2} \|\boldsymbol{\lambda}\|_2^2 \right), \rho > 0, \; \alpha \in [0, 1).$$

The unique edit magnitude assigned to component i is

$$\lambda_i^* = \mathrm{sign}(g_i) \, \frac{\max(|g_i| - \rho\alpha, \, 0)}{\rho(1-\alpha)}.$$

Proof. For component i, define the one-dimensional objective

$$J(\lambda_i) = g_i\lambda_i - \rho\left(\alpha|\lambda_i| + \frac{1-\alpha}{2}\lambda_i^2\right).$$

A scalar value λ_i^* maximizes J iff

$$0 \in \partial J(\lambda_i^*),$$

where the subgradient is needed only at $\lambda_i = 0$ due to the nondifferentiability of $|\lambda_i|$. We analyze the three regions $\lambda_i > 0$, $\lambda_i < 0$, and $\lambda_i = 0$.

Case 1: $\lambda_i > 0$**.** Here $|\lambda_i| = \lambda_i$, so

$$J(\lambda_i) = g_i\lambda_i - \rho\left(\alpha\lambda_i + \frac{1-\alpha}{2}\lambda_i^2\right).$$

Differentiating gives

$$\frac{dJ}{d\lambda_i} = g_i - \rho\alpha - \rho(1-\alpha)\lambda_i.$$

Setting this to zero yields

$$\lambda_i = \frac{g_i - \rho\alpha}{\rho(1-\alpha)},$$

which is valid only when the positivity assumption holds, i.e. $g_i > \rho\alpha$.

Case 2: $\lambda_i < 0$**.** Here $|\lambda_i| = -\lambda_i$, so

$$J(\lambda_i) = g_i\lambda_i + \rho\alpha\lambda_i - \frac{\rho(1-\alpha)}{2}\lambda_i^2.$$

Differentiating,

$$\frac{dJ}{d\lambda_i} = g_i + \rho\alpha - \rho(1-\alpha)\lambda_i.$$

Setting this to zero gives

$$\lambda_i = \frac{g_i + \rho\alpha}{\rho(1-\alpha)},$$

which is valid only when $g_i < -\rho\alpha$.

Case 3: $\lambda_i = 0$**.** At zero, the subdifferential of $|\lambda_i|$ is $\partial|\lambda_i| = [-1, 1]$. The optimality condition

$$0 \in g_i - \rho\alpha\, s, \qquad s \in [-1, 1],$$

is feasible iff $|g_i| \le \rho\alpha$. Thus $\lambda_i^* = 0$ whenever the alignment is too small to exceed the threshold.

Combining all three cases gives the soft-threshold rule

$$\lambda_i^* = \begin{cases} \dfrac{g_i - \rho\alpha}{\rho(1-\alpha)}, & g_i > \rho\alpha, \\ 0, & |g_i| \le \rho\alpha, \\ \dfrac{g_i + \rho\alpha}{\rho(1-\alpha)}, & g_i < -\rho\alpha. \end{cases}$$

Equivalently,

$$\lambda_i^* = \operatorname{sign}(g_i)\,\frac{\max(|g_i| - \rho\alpha,\ 0)}{\rho(1-\alpha)}.$$

□

C Component-Level Edit Visualizations

Figures 6–8 visualize the signed edit coefficients λ for the best-performing STEER2EDIT configuration in each behavioral control setting. Each cell represents the magnitude of a rank-1 weight update applied to an attention head or MLP neuron: positive values (red) reinforce components aligned with the target behavior, while negative values (blue) suppress components that oppose it. These heatmaps reveal a clear structural divergence across tasks: safety and truthfulness are mediated by sparse, attention-dominated edits, while reasoning efficiency requires dense, distributed MLP modifications.

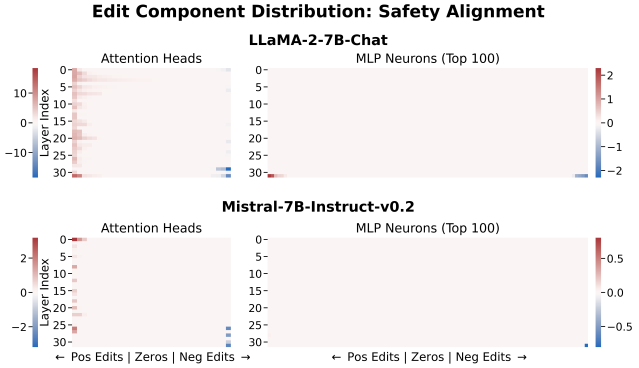


Figure 6: Signed STEER2EDIT edit coefficients λ for **safety alignment**. Edits are highly sparse and concentrated in a small subset of attention heads, predominantly in later layers. MLP neurons receive near-zero coefficients.

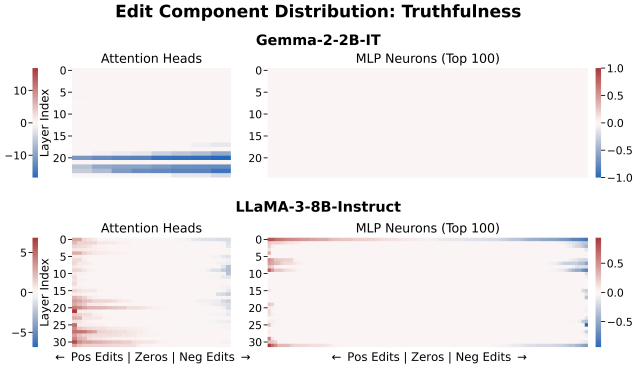


Figure 7: Signed STEER2EDIT edit coefficients λ for **truthfulness promotion**. Truthfulness control is sparse and predominantly mediated by attention heads. In **Gemma-2-2B-IT**, edits are dominated by negative coefficients, suggesting that truthfulness gains arise primarily from suppressing hallucination-promoting components.

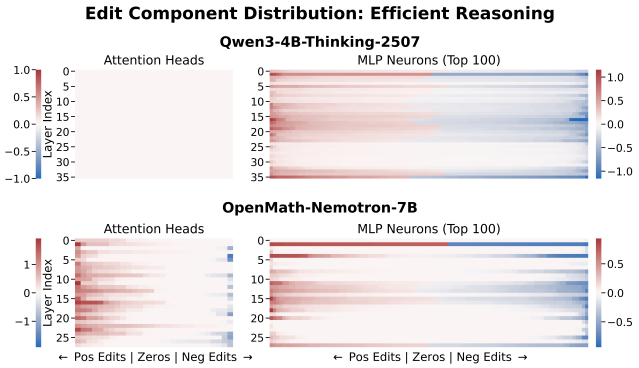


Figure 8: Signed STEER2EDIT edit coefficients λ for **reasoning efficiency**. In contrast to safety and truthfulness, reasoning efficiency is predominantly mediated by dense, distributed MLP edits spanning many neurons, while attention heads play a minor role.

D Steering Vector Construction

This appendix describes the procedures used to construct steering vectors for each behavioral control setting. Across all experiments, steering vectors are computed using the same general mean-difference formulation and are shared verbatim between activation steering and **STEER2EDIT**. Only the definition of positive and negative response sets differs by task.

General formulation. For a given model, layer ℓ, and block type (attention or MLP), we collect the block output activations for a set of responses. Let $\mathcal{P}$ and $\mathcal{N}$ denote the positive and negative response sets associated with a target behavior. For each response, we average the block outputs over all response tokens. The steering vector is then defined as the difference between the mean activations:

$$v_\ell = \mathbb{E}_{x \in \mathcal{P}}[h_\ell(x)] - \mathbb{E}_{x \in \mathcal{N}}[h_\ell(x)].$$

This procedure is applied independently to the attention and MLP blocks at each layer, yielding $\{v_\ell^{\text{attn}}, v_\ell^{\text{mlp}}\}_{\ell=1}^{L}$.

Safety alignment. For safety alignment, we construct steering vectors using the ADVBench dataset. The positive set $\mathcal{P}$ consists of refusal responses to harmful prompts, while the negative set $\mathcal{N}$ consists of standard helpful responses to benign questions sampled from Alpaca dataset. Steering vectors are computed from model-generated responses.

Truthfulness. For truthfulness promotion, we use the TruthfulQA dataset. We split the dataset into a probing set and an evaluation set. Model responses on the probing set are labeled as *truthful* or *hallucinated* using **QwQ-32B** as an external judge. The positive set $\mathcal{P}$ consists of truthful responses, and the negative set $\mathcal{N}$ consists of hallucinated responses.

efficient Reasoning. For reasoning efficiency control, we use the GSM8K training set. We measure the length of each model-generated reasoning trace and select the top 5% shortest and top 5% longest responses. The positive set $\mathcal{P}$ consists of short reasoning traces, and the negative set $\mathcal{N}$ consists of long reasoning traces. The resulting steering vectors capture directions associated with shorter internal reasoning processes.

All steering vectors are computed once per model and per behavioral control setting. During evaluation, the same steering vectors are applied across all test sets, reflecting a practical deployment scenario in which vectors are not optimized for any specific evaluation benchmark.

E Hyperparameter Search Procedure

STEER2EDIT introduces three scalar hyperparameters: the attention editing budget ρ_{attn}, the MLP editing budget ρ_{mlp}, and the Elastic-Net sparsity parameter α. These parameters control how edit magnitude is allocated across model components and how sparsely edits are distributed. Hyperparameters are selected using a lightweight two-stage grid search on held-out data.

Step 1: Coarse grid search

We first perform a coarse-grained grid search over a shared range that is identical across all models and behavioral control settings:

$$\rho_{\text{attn}} \in \{0.1, 0.3, 0.5, 0.7, 0.9\}, \quad \rho_{\text{mlp}} \in \{0.1, 0.3, 0.5, 0.7, 0.9\}, \quad \alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}.$$

The goal of this step is to identify the approximate operating regime (e.g., attention-dominated versus MLP-dominated edits, sparse versus distributed allocation), rather than to finely optimize performance.

Each configuration is first subjected to a lightweight sanity check using 20 short, simple prompts. If the edited model exhibits degenerate behavior (e.g., repetitive output, failure to respond, or nonsensical generations), the configuration is immediately discarded. This allows unstable settings to be filtered at negligible cost.

Step 2: Refined grid search

Based on the results of the coarse search, we define a refined but still small grid for each (model, setting) pair. The refined grids narrow the range and reduce the step size around regions that exhibit meaningful improvements in the target attribute while preserving normal model behavior.

In several settings, we observe that edits to either attention or MLP components have negligible impact on the target behavior. In these cases, the corresponding component is not edited at all, and no budget is assigned to that component during the refined search.

Final search ranges

Table 2 summarizes the refined hyperparameter ranges used in each behavioral control setting. All reported results in the main paper, including the best-performing configuration and the top-10 configurations shown in trade-off plots, are selected exclusively from these ranges.

Setting / Model	$\boldsymbol{\rho}_{\text{attn}}$	$\boldsymbol{\rho}_{\text{mlp}}$	$\boldsymbol{\alpha}$
Safety Alignment			
LLaMA-2-7B-Chat	$[0.16,\ 0.24]$ (step = 0.02)	$[0.35,\ 0.55]$ (step = 0.05)	$[0.70,\ 0.90]$ (step = 0.05)
Mistral-7B-Instruct-v0.2	$[0.42,\ 0.50]$ (step = 0.02)	$[0.40,\ 0.60]$ (step = 0.05)	$[0.65,\ 0.85]$ (step = 0.05)
Truthfulness			
Gemma-2-2B-IT	$[0.30,\ 0.50]$ (step = 0.05)	*negligible*	$[0.75,\ 0.95]$ (step = 0.05)
LLaMA-3-8B-Instruct	$[0.10,\ 0.14]$ (step = 0.01)	$[0.30,\ 0.50]$ (step = 0.05)	$[0.30,\ 0.70]$ (step = 0.10)
Efficient Reasoning			
Qwen3-4B-Thinking-2507	*negligible*	$[0.65,\ 0.80]$ (step = 0.05)	$[0.05,\ 0.20]$ (step = 0.05)
OpenMath-Nemotron-7B	$[0.20,\ 0.30]$ (step = 0.05)	$[0.80,\ 0.90]$ (step = 0.05)	$[0.10,\ 0.20]$ (step = 0.05)

Table 2: Refined hyperparameter search ranges for **STEER2EDIT**. "Negligible" indicates that edits to the corresponding component were found to have insufficient effect during coarse search and are therefore not applied in the refined search.

Efficiency and reporting

Hyperparameter search for **STEER2EDIT** is computationally lightweight. Each configuration requires only a single closed-form application of rank-1 weight edits, followed by evaluation on a held-out small validation set. In practice, the full two-stage search completes within minutes per model, and does not involve gradient-based optimization. All evaluations are performed on held-out data that is disjoint from steering vector extraction. No additional tuning is performed on test sets.

F Per-Dataset Trade-off Analysis

In Section 4.1, we summarize each behavioral control setting using aggregated downstream utility metrics to provide a concise comparison across methods. In this appendix, we present *per-dataset trade-off curves* that expose finer-grained behavior across individual evaluation benchmarks. All three activation-steering baselines (RepEngineering, CAA, ODESteer) are shown alongside **STEER2EDIT**. These results demonstrate that the superior attribute–utility trade-offs achieved by **STEER2EDIT** are consistent across datasets and across all baselines.

F.1 Safety Alignment: Attack- and Dataset-Specific Trade-offs

For safety alignment, we evaluate two jailbreak attack methods (GCG and ADV-LLM) and three downstream utility benchmarks (CommonsenseQA, Code-MMLU, and GSM8K), resulting in six distinct safety–utility trade-off settings per model.

Figure 9 reports refusal rate versus downstream utility separately for each attack–dataset pair on **LLaMA-2-7B-Chat** and **Mistral-7B-Instruct-v0.2**. Across most settings, **STEER2EDIT** identifies configurations that achieve higher refusal rates at comparable or higher utility than all three activation-steering baselines.

For **LLaMA-2-7B-Chat**, **STEER2EDIT** consistently dominates all baselines under both GCG and ADV-LLM attacks across all downstream datasets. CAA and RepEngineering follow similar trajectories, while ODESteer achieves lower refusal rates at comparable utility costs. For **Mistral-7B-Instruct-v0.2**, CAA reaches a trade-off region closer to **STEER2EDIT** on some attack–dataset pairs, but this competitiveness is setting-specific and does not generalize to truthfulness or reasoning efficiency.

Notably, the advantage of **STEER2EDIT** becomes more pronounced as the attack strength increases. While all activation-steering baselines require aggressive intervention that sharply degrades utility under ADV-LLM, weight-level edits derived by **STEER2EDIT** maintain stable benign-task performance while substantially improving robustness to strong jailbreaks.

F.2 Truthfulness: Dataset-Specific Utility Trade-offs

For truthfulness promotion, downstream utility is evaluated independently on CommonsenseQA, Code-MMLU, and GSM8K. Figure 10 shows truthfulness versus utility accuracy for **Gemma-2-2B-IT** and **LLaMA-3-8B-Instruct** on each benchmark.

Across all datasets, all three activation-steering baselines exhibit a pronounced trade-off in which increasing truthfulness rapidly degrades task performance. In contrast, **STEER2EDIT** consistently attains higher truthfulness at substantially higher utility, with edited configurations occupying regions of the trade-off space that are unattainable by any steering baseline.

These per-dataset results demonstrate that the truthfulness gains of **STEER2EDIT** generalize across reasoning-heavy and knowledge-oriented benchmarks.

F.3 Efficient Reasoning: Dataset-Level Accuracy–Length Trade-offs

For efficient reasoning control, we report dataset-specific trade-offs between answer accuracy and reasoning length on GSM8K, MATH-500, GPQA, and Code-MMLU for **Qwen3-4B-Thinking-2507** and **OpenMath-Nemotron-7B**.

Figure 11 shows that all three activation-steering baselines reduce reasoning length primarily by sacrificing accuracy, with the severity of this trade-off varying substantially across datasets. In contrast, **STEER2EDIT** consistently identifies configurations that shorten reasoning traces while preserving accuracy, dominating the Pareto frontier of all baselines on every dataset including challenging benchmarks such as GPQA.

Notably, these improvements generalize beyond GSM8K, despite the steering direction being extracted from GSM8K, indicating that **STEER2EDIT** captures a transferable mechanism for reasoning efficiency control.

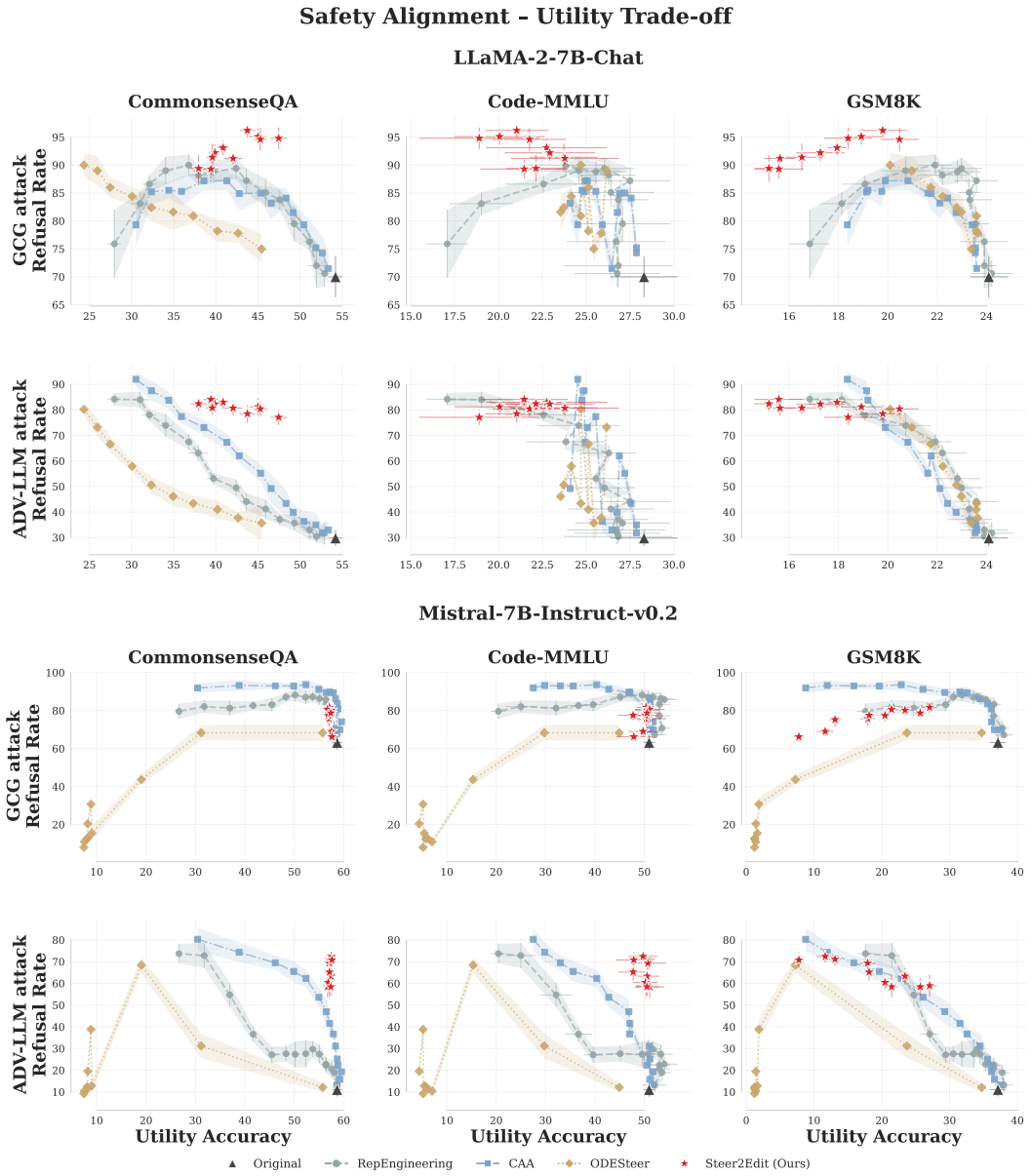


Figure 9: Per-dataset safety–utility trade-offs under GCG and ADV-LLM attacks. Each column corresponds to a downstream utility dataset (CommonsenseQA, Code-MMLU, GSM8K), and each row corresponds to an attack method. Three activation-steering baselines (RepEngineering, CAA, ODESteer) are shown alongside **STEER2EDIT**. **STEER2EDIT** consistently achieves higher refusal rates at comparable or higher utility than all baselines across most settings.

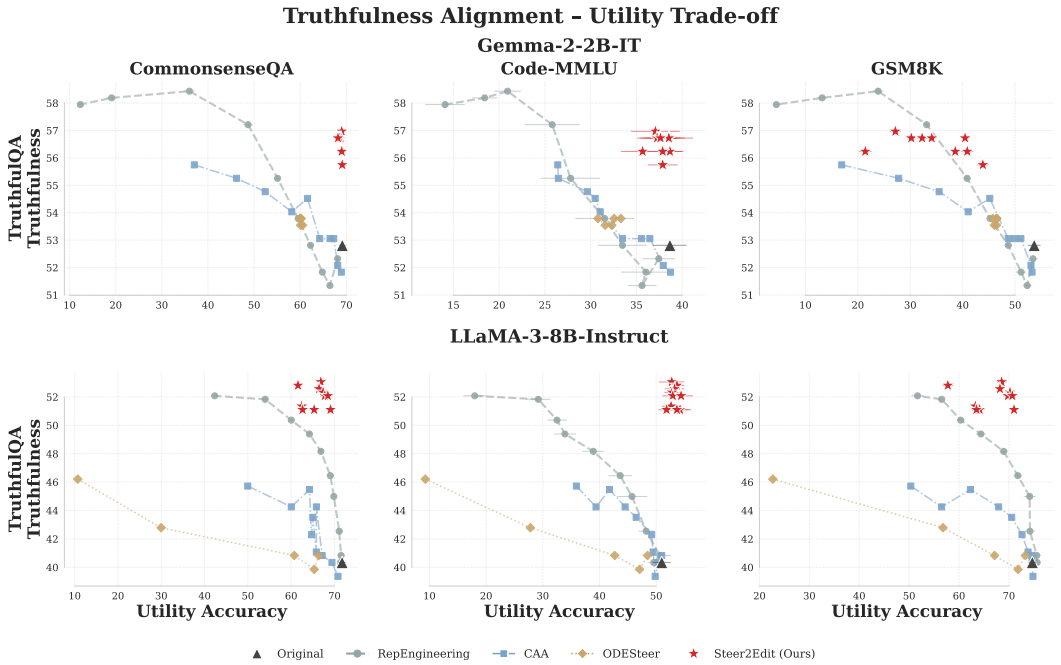


Figure 10: Per-dataset truthfulness–utility trade-offs on CommonsenseQA, Code-MMLU, and GSM8K for Gemma-2-2B-IT and LLaMA-3-8B-Instruct. Three activation-steering baselines (RepEngineering, CAA, ODESteer) are shown alongside **STEER2EDIT**. **STEER2EDIT** improves truthfulness while preserving higher downstream utility than all baselines across all datasets.

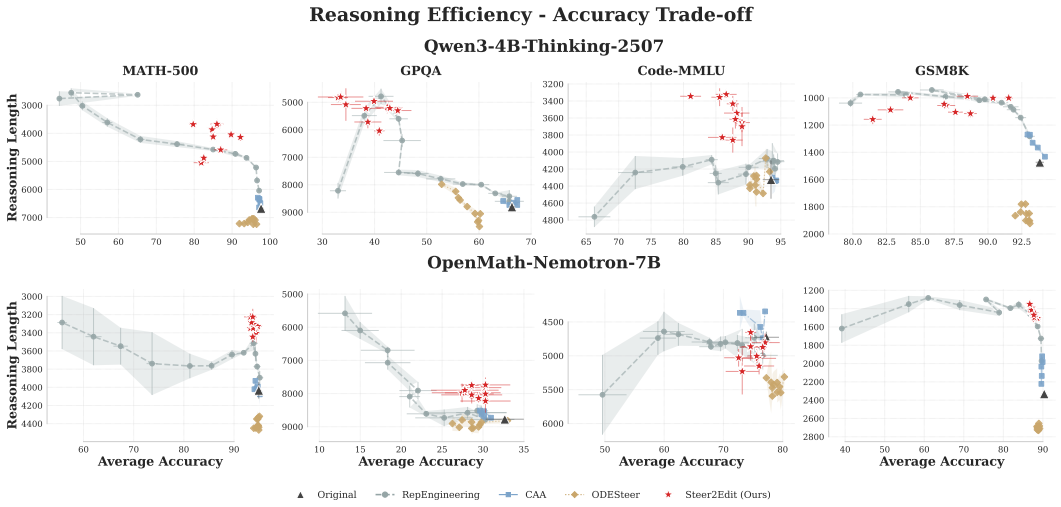


Figure 11: Per-dataset accuracy–reasoning-length trade-offs on GSM8K, MATH-500, GPQA, and Code-MMLU for Qwen3-4B-Thinking-2507 and OpenMath-Nemotron-7B. Three activation-steering baselines (RepEngineering, CAA, ODESteer) are shown alongside **STEER2EDIT**. **STEER2EDIT** reduces reasoning length while maintaining accuracy, dominating all baselines across all datasets.

G Ablation Study of STEER2EDIT: Empirical Justification of Formal Design Choices

In this section, we validate the design choices of **STEER2EDIT** through a systematic ablation study. We decompose the rank-1 weight update, $\Delta W_i = \lambda_i u_i k_i^\top$, and independently modify its three core components: the **Input Direction** (k), the **Importance Score** (g), and the **Edit Magnitude** (λ), while holding the others constant.

This study aims to determine whether the performance of **STEER2EDIT** arises from its specific geometric and statistical formulations—such as using cosine similarity for scoring or Elastic-Net for sparsity—rather than generic weight modifications. Table 3 defines the five ablation variants tested and the specific hypothesis each one investigates.

Table 3: **Ablation Definitions.** We categorize variants by the component they modify: Input Direction (k), Importance Score (g), or Magnitude (λ).

Ablation Category	Variant	Definition and Hypothesis Tested
Input k	k_{mean}	Sets $k_i \leftarrow \mu_i$, where $\mu_i = \mathbb{E}[h_i]$. Tests if data statistics alone suffice without directional sensitivity.
	k_{svd}	Sets $k_i \leftarrow \mathbf{v}_1$, where $\mathbf{v}_1$ is the top right singular vector of W_i. Tests if intrinsic weight directions suffice.
Score g	g_{dot}	Sets $g_i \leftarrow \hat{v}_i^\top (W_i \mu_i)$ (unnormalized dot product), removing the normalization by $\|W_i \mu_i\|_2$. Tests the effect of component-output normalization.
Magnitude λ	ℓ_0	Selects top-K components (hard threshold) matching Elastic-Net sparsity. Tests if sparsity alone explains gains.
	ℓ_2	Uses Elastic-Net with $\alpha = 0$ (Ridge regularization), resulting in dense edits. Tests if dense edits can preserve utility.

G.1 Unified Performance Summary

To facilitate a high-level comparison, Table 4 aggregates the results across all three behavioral settings: Safety Alignment, Truthfulness, and Efficient Reasoning.

We normalize the attribute scores to a common $[0, 1]$ scale. For **Safety** and **Truthfulness**, we use the raw percentage divided by 100. For **Efficient Reasoning**, where lower length is better, we define the efficiency score as $\min(1, L_{\textbf{STEER2EDIT}}/L_{\text{Ablation}})$. **STEER2EDIT** consistently achieves the highest combined **Attribute** $\times$ **Utility** score, demonstrating that precise input alignment, normalized scoring, and sparse regularization are all critical for optimal performance.

Table 4: **Unified Performance Summary.** Results are averaged across 6 models (2 per setting). **STEER2EDIT** achieves the best global trade-off between targeting the attribute and maintaining downstream utility.

Category	Variant	Normalized Attribute Score ↑			Normalized Utility Score ↑			Overall Average ↑		Attr × Util
		Safety	Truth	Effic.	Safety	Truth	Effic.	**Attribute**	**Utility**	
Full Method	**Steer2Edit**	0.807	**0.550**	**1.000**	0.341	0.536	0.755	**0.786**	0.544	**0.427**
Input k	k_{mean}	**0.827**	0.531	0.926	0.279	0.350	0.589	0.761	0.406	0.309
	k_{svd}	0.536	0.474	0.757	**0.400**	**0.572**	**0.805**	0.589	**0.592**	0.349
Score g	g_{dot}	0.102	0.509	0.628	0.047	0.232	0.003	0.413	0.094	0.039
Magnitude λ	ℓ_0	0.506	0.546	0.767	0.321	0.423	0.596	0.606	0.447	0.271
	ℓ_2	0.000	0.580	0.472	0.032	0.226	0.394	0.350	0.217	0.076

Key Insights: Normalization and Sparsity. Two critical design principles emerge from these aggregated results. First, **Score Normalization is paramount for stability.** The catastrophic failure of the unnormalized g_{dot} variant (Overall Score: 0.039) reveals that raw activation magnitudes vary

drastically across model layers. Without the cosine-similarity normalization used in **STEER2EDIT**, the editing process becomes dominated by high-norm layers, destabilizing the model regardless of the target attribute. Second, **Sparsity is essential for utility preservation.** The dense ℓ_2 baseline struggles to maintain downstream performance (Utility: 0.217), confirming that precise, sparse interventions are required to disentangle specific behaviors without overwriting the model's general knowledge base.

G.2 Detailed Per-Setting Results

The following subsections analyze the impact of each component across our three behavioral settings. We find that while different tasks exhibit unique sensitivities, the failures of the ablation baselines consistently point to the necessity of **STEER2EDIT**'s three-pillared approach: precise directional alignment, normalized scoring, and sparse editing.

Safety Alignment: Selective vs. Uniform Triggering. Table 5 highlights the risks of uniformly triggered edits. The k_{mean} baseline, which activates edits based on the global mean activation $\mathbb{E}[x]$, achieves high safety (e.g., 92.20% refusal on ADV-LLM) but substantially reduces utility (21.75% vs. 28.00% for **STEER2EDIT**). This indicates that activating edits in a largely input-agnostic manner leads to significant degradation on benign downstream tasks. Conversely, the dense ℓ_2 baseline results in near-total model failure (0.05% safety), suggesting that safety-related behavior is mediated by localized components and is disrupted by dense parameter modifications. **STEER2EDIT** avoids these failure modes by combining input-selective activation with sparse, component-level edits.

Table 5: **Safety Alignment Ablations (Detailed).** Metrics are Average Safety (Refusal Rate) and Average Utility (Accuracy).

Model	Variant	S × U	Avg Safety	Avg Util	Safety (Refusal Rate)		Utility (Accuracy)		
					GCG	ADV-LLM	CommonsenseQA	Code-MMLU	GSM8K
Llama2-7B-Chat	Steer2Edit	**24.68**	**88.15**	28.00	**95.10**	81.20	45.01	20.06	18.93
	k_{mean}	19.12	87.90	21.75	83.60	**92.20**	30.14	19.57	15.53
	k_{svd}	22.82	66.35	**34.39**	90.80	41.90	**50.77**	**30.49**	**21.90**
	g_{dot}	0.69	15.35	4.49	23.30	7.40	6.45	5.37	1.65
	ℓ_0	7.66	41.25	18.58	36.40	46.10	27.26	25.43	3.05
	ℓ_2	0.00	0.05	6.33	0.10	0.00	3.60	14.88	0.52
Mistral-7B-Instruct	Steer2Edit	**29.38**	73.15	40.16	75.20	71.10	**57.00**	50.37	13.10
	k_{mean}	26.33	**77.45**	34.00	71.60	**83.30**	54.24	45.55	2.22
	k_{svd}	18.59	40.85	45.51	68.80	12.90	53.61	**51.95**	30.98
	g_{dot}	0.25	5.05	4.96	0.90	9.20	5.43	9.45	0.01
	ℓ_0	27.29	59.85	**45.60**	**77.20**	42.50	55.96	48.35	**32.48**
	ℓ_2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Truthfulness: The Critical Role of Sparsity. Table 6 provides strong evidence for the necessity of sparse editing (ℓ_0 regularization). The dense ℓ_2 variant, which modifies all parameters in the target block, causes catastrophic utility collapse (0.00% on Llama3-8B), demonstrating that dense edits severely disrupt the model's general capabilities. Additionally, we observe an informative trade-off with k_{svd}: while it preserves high utility (65.11% on Llama3), it fails to significantly improve truthfulness (39.61%). This suggests that aligning edits with the model's intrinsic dominant activation patterns is insufficient to induce a shift toward truthful behavior. Only **STEER2EDIT** balances these objectives: it uses k to selectively activate edits on semantically relevant inputs, while ℓ_0 sparsity restricts the intervention to a small set of behaviorally relevant neurons.

Table 6: **Truthfulness Ablations (Detailed).** Metrics are Average Truthfulness (TruthfulQA Accuracy) and Average Utility (Accuracy).

Model	Method	T × U	Avg Truth	Avg Util	Attribute	Utility (Accuracy)		
					TruthfulQA	CommonsenseQA	Code-MMLU	GSM8K
Gemma2 2B-IT	Steer2Edit	25.30	56.97	44.41	56.97	**69.00**	37.07	27.16
	k_{mean}	24.93	55.75	44.71	55.75	68.59	38.29	27.24
	k_{svd}	**27.26**	55.26	**49.33**	55.26	68.12	**39.39**	**40.48**
	g_{dot}	1.16	**58.68**	1.98	**58.68**	2.83	2.20	0.90
	ℓ_0	15.33	57.95	26.46	57.95	58.29	16.04	5.06
	ℓ_2	26.15	57.95	45.12	57.95	63.62	36.95	34.78
Llama3 8B-Instruct	Steer2Edit	**33.27**	53.06	62.71	53.06	66.86	**52.74**	68.54
	k_{mean}	12.74	50.37	25.29	50.37	39.67	25.61	10.60
	k_{svd}	25.79	39.61	**65.11**	39.61	**71.30**	51.71	**72.32**
	g_{dot}	19.11	43.03	44.42	43.03	26.94	41.77	64.56
	ℓ_0	29.87	51.34	58.18	51.34	64.14	47.13	63.26
	ℓ_2	0.00	**57.95**	0.00	**57.95**	0.00	0.00	0.00

Efficient Reasoning: Stability via Normalization. Table 7 highlights the critical role of score normalization for stable reasoning control. The g_{dot} variant, which uses the raw dot product, fails pathologically: it reduces the reasoning length to just 21 tokens (Qwen3-4B) while collapsing utility to near zero (0.13%). This behavior arises because activation norms vary substantially across layers; without the cosine normalization used in **STEER2EDIT**, edit magnitudes become dominated by high-norm layers, leading to severe disruption of model behavior. Conversely, the k_{svd} baseline increases reasoning length (5351 vs. 3467 for **STEER2EDIT**), indicating that activating edits along the model's intrinsic dominant activation patterns is insufficient for improving efficiency. Together, these results show that effective reasoning-length control requires both normalized importance scoring and task-specific edit activation, as implemented in **STEER2EDIT**.

Table 7: **Efficient Reasoning Ablations (Detailed).** Metrics are Reasoning Length (Lower is Better) and Utility (Higher is Better). The **U / L** metric represents efficiency (Utility/Length × 100).

Model	Variant	U / L	Avg Len	Avg Util	Reasoning Length (Lower is Better)				Utility (Higher is Better)			
					MATH-500	GPQA	Code-MMLU	GSM8K	MATH-500	GPQA	Code-MMLU	GSM8K
Qwen3 4B-Thinking	Steer2Edit	**2.28**	3467	78.95	4136	5299	3433	1000	92.2	44.4	87.6	91.5
	k_{mean}	1.56	3266	51.05	3188	2728	6129	1018	53.2	10.3	78.1	62.7
	k_{svd}	1.64	5351	**87.99**	6389	8780	4622	1613	**96.9**	**67.5**	**93.9**	**93.6**
	g_{dot}	0.62	**21**	0.13	**23**	**33**	**10**	**18**	0.3	0.0	0.0	0.2
	ℓ_0	0.73	6506	47.61	8646	5202	4698	7477	44.7	36.0	65.7	44.1
	ℓ_2	0.08	22476	17.63	32223	18987	6420	32274	3.0	15.2	50.4	2.0
OpenMath-Nemotron-7B	Steer2Edit	1.62	4445	72.12	3405	**7821**	5038	**1515**	94.4	30.1	**75.8**	88.1
	k_{mean}	1.28	5216	66.69	3838	8814	6522	1691	92.5	25.8	66.7	81.9
	k_{svd}	1.42	5131	**73.01**	4188	8880	4985	2472	93.9	**32.8**	75.6	**89.7**
	g_{dot}	0.00	17338	0.52	18380	16628	17163	17182	0.9	0.0	0.6	0.6
	ℓ_0	**1.64**	**4356**	71.60	3422	7971	**4503**	1528	**94.5**	29.6	73.4	88.9
	ℓ_2	1.09	5633	61.23	**2911**	12686	5160	1773	89.5	17.7	66.1	71.7

H Component-Wise Budget Sensitivity Analysis

In the main paper, we show that the best-performing **STEER2EDIT** configurations exhibit a consistent component-level structure: safety and truthfulness control rely on sparse edits to attention heads, whereas reasoning efficiency is primarily governed by distributed MLP neuron edits.

To further validate that this structural separation is intrinsic to the underlying mechanisms, we conduct a controlled *component-wise budget sensitivity analysis*. In this study, we fix the sparsity parameter α and vary the regularization budget of one component class at a time, while disabling edits to the other class by sending its budget to infinity ($\rho \to \infty$). This isolates how changes in attention and MLP budgets individually influence the attribute–utility trade-off.

H.1 Safety Alignment: Sensitivity to Attention Budget

Figure 12 illustrates how the safety–utility trade-off responds to changes in the attention and MLP budgets when considered in isolation. Increasing the attention budget ρ_{attn} produces substantial gains in refusal rate at moderate utility cost, closely matching the best joint configurations reported in the main paper. In contrast, varying the MLP budget ρ_{mlp} leads to markedly weaker safety improvements and often degrades utility more rapidly.

This asymmetric sensitivity indicates that safety alignment is primarily mediated by a small number of attention heads. The result is consistent with the sparsity patterns observed in Figure 6, where non-zero edit coefficients are concentrated in late-layer attention components, with minimal contribution from MLP neurons.

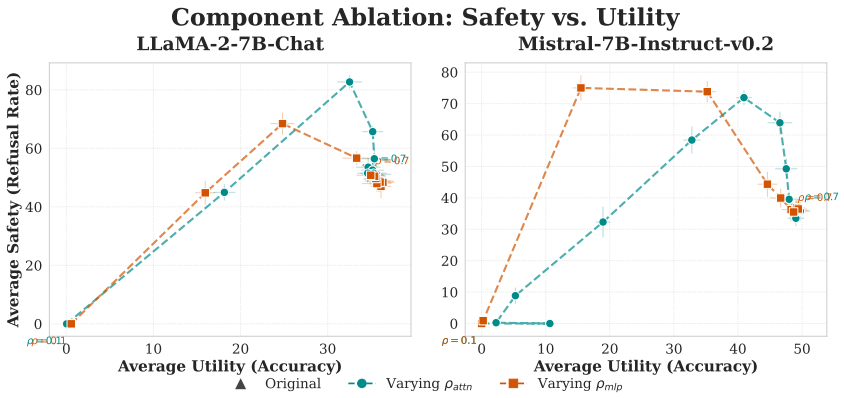


Figure 12: **Component-wise budget sensitivity for safety alignment.** We fix the sparsity parameter α and vary the attention regularization budget ρ_{attn} while disabling MLP edits by taking $\rho_{\text{mlp}} \to \infty$, and vice versa. Improvements in refusal rate are primarily driven by changes in the attention budget, whereas varying the MLP budget yields limited safety gains and earlier utility degradation.

H.2 Truthfulness: Sensitivity Dominated by Attention

Figure 13 reports the component-wise budget sensitivity for truthfulness promotion. Across both evaluated models, increasing the attention budget consistently yields larger improvements in truthful preference accuracy than increasing the MLP budget under the same sparsity constraint. MLP-only edits fail to recover the trade-off frontier achieved by attention edits.

These findings align with the component-level edit distributions shown in Figure 7, which reveal that truthfulness control is achieved through sparse, localized attention interventions. Notably, several models exhibit predominantly negative edit coefficients, suggesting that suppressing hallucination-promoting attention heads is more effective than broadly modifying MLP computation.

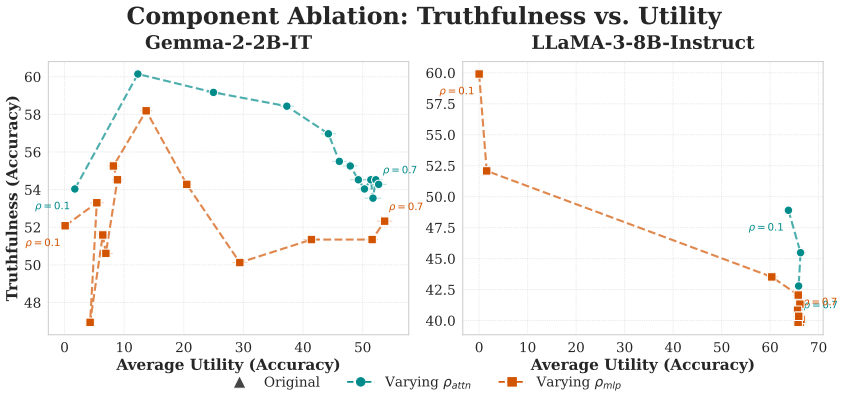


Figure 13: **Component-wise budget sensitivity for truthfulness control.** Truthfulness improvements are strongly sensitive to the attention budget ρ_{attn}, while varying the MLP budget ρ_{mlp} in isolation results in substantially smaller gains at comparable downstream utility.

H.3 Efficient Reasoning: Sensitivity Dominated by MLP Budget

Figure 14 shows that reasoning efficiency exhibits a qualitatively different sensitivity pattern. Increasing the MLP budget ρ_{mlp} leads to a smooth and substantial reduction in reasoning length while preserving accuracy. In contrast, varying the attention budget ρ_{attn} in isolation produces only marginal efficiency improvements, even at large budgets.

This behavior mirrors the dense MLP edit patterns observed in Figure 8 and confirms that efficient reasoning control requires coordinated, distributed modifications to MLP neurons across layers. Unlike safety and truthfulness, which are governed by localized attention-based circuits, reasoning efficiency emerges from broad MLP-based computation.

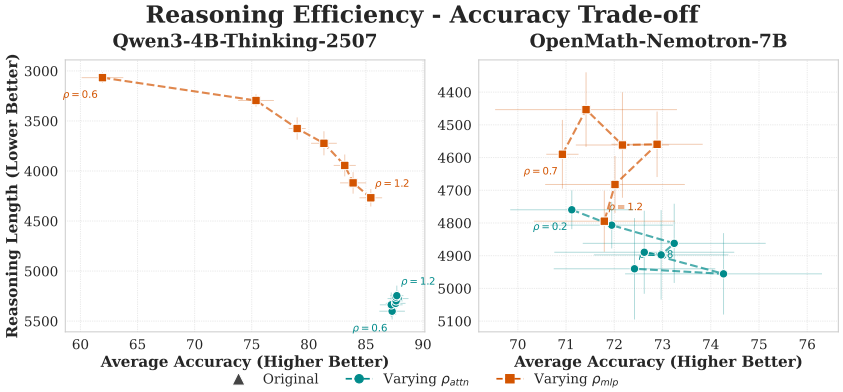


Figure 14: **Component-wise budget sensitivity for reasoning efficiency.** Reductions in reasoning length are strongly influenced by the MLP budget ρ_{mlp}, while varying the attention budget ρ_{attn} yields only minor efficiency gains. This indicates that efficient reasoning is governed by distributed MLP computation rather than sparse attention circuits.

Summary. Across all behavioral control settings, this component-wise budget sensitivity analysis establishes a clear correspondence between the component class whose budget most strongly influences the trade-off frontier and the components receiving non-zero edits in the best-performing **STEER2EDIT** configurations. Safety and truthfulness are attention-dominated, whereas reasoning efficiency is MLP-dominated, providing further evidence that **STEER2EDIT** uncovers genuine, setting-dependent circuit structure rather than artifacts of hyperparameter tuning.

I Additional Baselines: Comparing STEER2EDIT with Training-Based Methods

We compare STEER2EDIT against training-based adaptation methods that directly optimize model parameters toward a target behavior. Specifically, we consider **full-parameter fine-tuning** and **rank-1 LoRA fine-tuning** as additional baselines.

Setup and comparison protocol. For each control setting (safety, truthfulness, and efficient reasoning), we fine-tune models on the *same probing dataset used to extract steering vectors*. Training uses the *positive set* (i.e., examples that exhibit the target attribute), so the model is explicitly optimized to imitate the desired behavior. We evaluate the resulting models using the same attribute and downstream utility metrics as in the main experiments and report trade-off curves alongside activation steering and STEER2EDIT. The **full** fine-tuning baseline updates all model parameters. The **rank-1** baseline applies LoRA adapters with rank $r = 1$, inserted into the standard attention projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`) and MLP projections (`gate_proj`, `up_proj`, `down_proj`), while keeping the backbone weights frozen. All baselines are trained with standard supervised objectives and comparable training budgets.

I.1 Safety Alignment

Figure 15 shows the safety–utility trade-off for models fine-tuned on the safety probing positive set. Full fine-tuning increases refusal rates but often does so by globally shifting the model's response distribution, leading to over-refusal on benign queries and sharp drops in downstream utility, particularly for Mistral. Rank-1 LoRA produces minimal changes, indicating limited capacity to induce reliable safety behavior under this supervision.

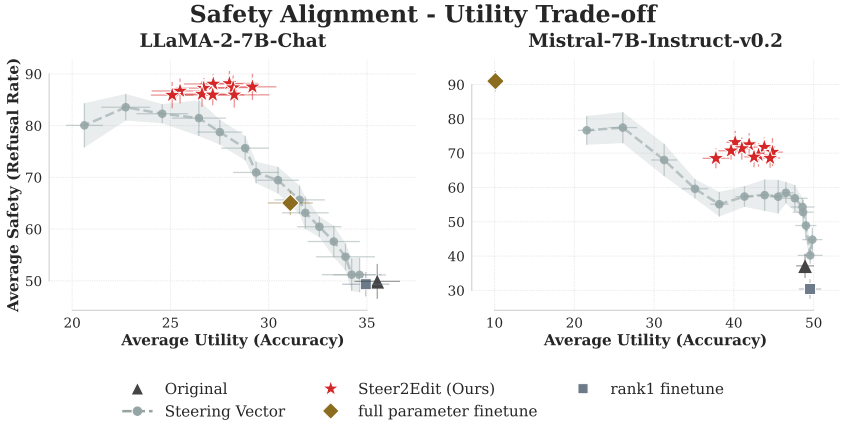


Figure 15: **Safety–utility trade-off with training-based baselines.** Full fine-tuning improves refusal rates but frequently collapses utility due to over-refusal, especially in the low-data regime. Rank-1 LoRA has negligible effect. These trends indicate that training-based optimization on small positive sets induces coarse, global behavioral shifts rather than selective safety control.

I.2 Truthfulness

Figure 16 reports the truthfulness–utility trade-off for models fine-tuned on the truthfulness probing positive set. Full fine-tuning yields modest improvements in TruthfulQA accuracy, but these gains are typically accompanied by noticeable degradation in downstream utility, suggesting broad shifts in the model's answer distribution rather than selective promotion of truthfulness. Rank-1 LoRA again exhibits little effect.

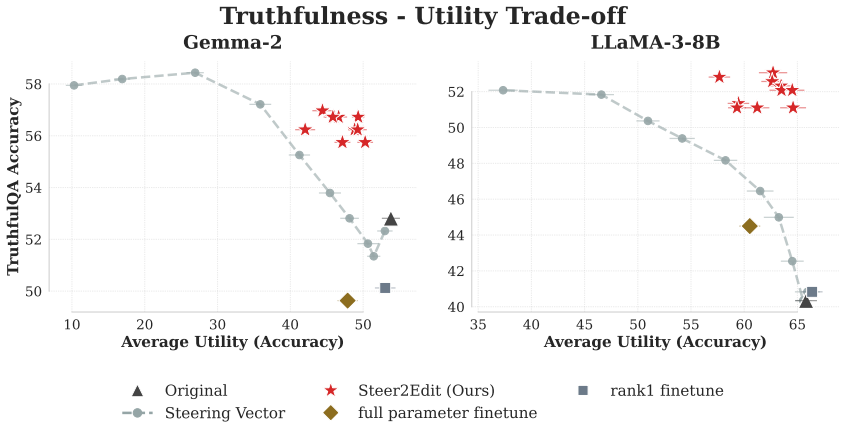


Figure 16: **Truthfulness–utility trade-off with training-based baselines.** Fine-tuning on the positive set improves TruthfulQA performance but often incurs utility loss, reflecting over-regularization of response behavior. Rank-1 LoRA provides insufficient adaptation capacity to meaningfully alter truthfulness.

I.3 Efficient Reasoning

Figure 17 presents the reasoning efficiency–accuracy trade-off for models fine-tuned on the efficient-reasoning probing positive set. Full fine-tuning can encourage shorter generations, but the resulting reductions in reasoning length are generally comparable to activation steering and do not consistently surpass it. Rank-1 LoRA again produces minimal changes, indicating limited capacity to meaningfully influence reasoning behavior under this supervision.

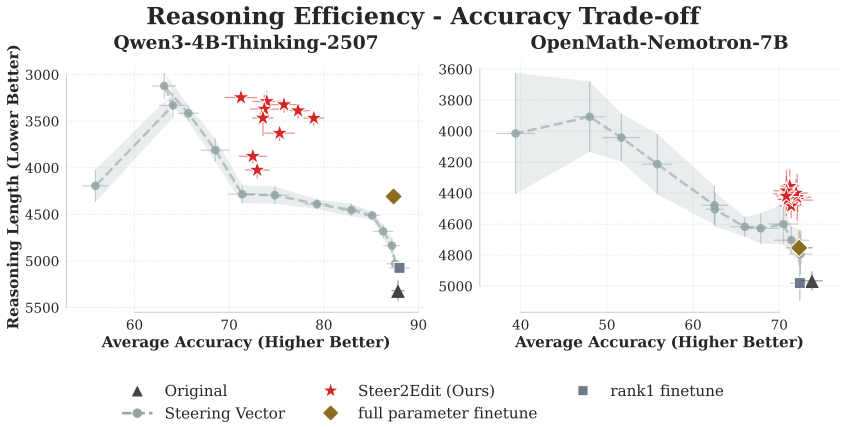


Figure 17: **Reasoning efficiency–accuracy trade-off with training-based baselines.** Full fine-tuning reduces reasoning length but largely matches the trade-off achieved by activation steering, without clear advantages. Rank-1 LoRA has little effect.

Summary. Across all three control settings, fine-tuning on the probing dataset can move models toward the target attribute, but typically does so by inducing broad distributional shifts that trace a trade-off curve similar to activation steering. These limitations are most evident in the low-data regime considered here, where the probing sets are intentionally small and narrowly targeted. Rank-1 LoRA consistently exhibits weak effects across all settings. In contrast, STEER2EDIT achieves more favorable trade-offs while remaining training-free and component-interpretable, highlighting the benefit of converting steering diagnostics into targeted weight edits rather than optimizing behavior through global parameter updates.

J Additional Baselines: Comparing STEER2EDIT with AlphaEdit (Null-Space Weight Editing)

AlphaEdit [Fang et al., 2024] is a **weight editing** method originally designed for localized factual knowledge updates (e.g., changing "Eiffel Tower is in Paris" to "London"). We adapt it to the behavioral control setting to test whether null-space constrained editing can achieve global behavioral shifts. The method operates at a single MLP layer ($\lfloor L/3 \rfloor$) and proceeds as follows: (1) Extract MLP intermediate activations from harmful prompts as edit keys K_1. (2) Extract MLP output activations from refusal responses as edit values V_1. (3) Extract MLP intermediate activations from benign (Alpaca) prompts as the preservation set K_0. (4) Compute a null-space projector P via SVD of $K_0 K_0^\top$, ensuring the edit does not disturb the model's behavior on benign inputs. (5) Compute the weight update $\Delta W = (V_1 - WK_1)K_1^\top P(K_1 K_1^\top P + I)^{-1} P$ and apply it to the `down_proj` weights.

Unlike STEER2EDIT, which edits attention heads and MLP neurons across all layers, AlphaEdit edits only a single layer's MLP. Since safety behavior is distributed across many layers and both attention and MLP components, this localized approach is fundamentally limited for global behavioral control.

Table 8: AlphaEdit vs. STEER2EDIT on LLaMA-2-7B-Chat safety alignment (n=10). AlphaEdit has essentially no effect (+0.4%), confirming that null-space constraints are too restrictive for behavioral control.

Method	GCG Refusal	AdvLLM Refusal	Avg Safety	Avg Utility
Original	70.0%	29.8%	49.9%	35.5%
AlphaEdit	69.6%	30.9%	50.3%	35.0%
Steer2Edit	**95.1%**	**81.2%**	**88.2%**	**28.0%**

Table 9: AlphaEdit vs. STEER2EDIT on LLaMA-3-8B-Instruct truthfulness. AlphaEdit has no meaningful effect on either hallucination benchmark.

Method	TruthfulQA	Expert-FACTOR	Avg Utility
Original	40.3%	41.5%	65.8%
AlphaEdit	39.9%	40.7%	66.3%
Steer2Edit	**53.1%**	**54.2%**	**62.7%**

K Additional Baselines: Comparing STEER2EDIT with Activation-Steered Compression (ASC)

Activation-Steered Compression (ASC) [Azizi et al., 2025] is a steering method specifically designed for reasoning efficiency. It differs from CAA in three key ways. (1) **Data:** Instead of splitting responses by length, ASC constructs **verbose–concise Chain-of-Thought pairs** by rewriting the top 5% longest correct GSM8K responses into concise versions using an LLM (Claude Sonnet 4.5). (2) **Extraction point:** ASC extracts from the **final token only** rather than computing span means over the full response, capturing the "compression intent" at the end of generation. (3) **Strength calibration:** Rather than sweeping α, ASC computes a **principled maximum steering strength** $\gamma_{\max}$ via a closed-form KL-divergence bound (with budget $\varepsilon = 10^{-3}$), using JVP norm estimation and directional curvature correction. Steering is applied at layer $\lfloor 3L/4 \rfloor$. We evaluate on Qwen3-4B-Thinking.

Table 10: ASC reasoning efficiency on Qwen3-4B-Thinking: per-dataset breakdown (n=3). ASC's KL-calibrated $\gamma_{\max}$ is overly conservative, producing essentially no change in accuracy or length on any dataset.

	GSM8K		MATH-500		GPQA		CodeMMLU		Average	
Method	Acc	Len	Acc	Len	Acc	Len	Acc	Len	Acc	Len
Original	93.8	1473	97.7	6675	66.3	8805	93.5	4324	87.8	5319
ASC (calibrated γ)	93.6	1476	97.7	6715	66.5	8777	93.1	4349	87.7	5329
Steer2Edit (best)	**90.4**	**1001**	**89.7**	**4045**	**42.4**	**5182**	**86.6**	**3322**	**77.3**	**3387**

L OR-Bench Over-Refusal Evaluation

OR-Bench [Cui et al., 2024] is an evaluation benchmark that measures **over-refusal**: the rate at which a model incorrectly refuses safe or borderline-safe prompts. We use OR-Bench-Hard-1K, a curated subset of $\sim$1,000 borderline-safe prompts that safety-enhanced models frequently refuse despite the prompts being benign. The over-refusal rate is defined as the fraction of these safe prompts that the model refuses (lower is better). This directly addresses whether STEER2EDIT's safety improvements come at the cost of excessive caution on non-harmful inputs.

Table 11: OR-Bench over-refusal evaluation on LLaMA-2-7B-Chat. At matched over-refusal ($\sim$66%), STEER2EDIT achieves +28% higher safety than steering, indicating targeted behavioral modification rather than uniform refusal amplification.

Method	Avg Safety	Over-Refusal $\downarrow$
Original	49.9%	60.1%
Steering (α=0.3)	54.7%	63.1%
Steering (α=0.5)	60.5%	65.8%
Steering (α=0.7)	65.7%	68.7%
Steering (α=1.0)	75.7%	72.6%
Steering (α=1.5)	80.1%	76.0%
Steer2Edit	**88.1%**	**66.4%**

M Scaling to 70B Larger LLMs

To verify that STEER2EDIT scales beyond 7B models, we evaluate on **LLaMA-2-70B-Chat** in the safety alignment setting. The same pipeline is applied: directions are extracted from contrastive response pairs, and rank-1 edits are computed via Elastic-Net. The optimal hyperparameter region ($\rho_{\text{attn}} \approx 0.3$, $\rho_{\text{mlp}} \approx 0.4$) is similar to that of the 7B model, suggesting that the sparsity structure of behavioral control is consistent across model scales.

Table 12: Safety alignment on LLaMA-2-70B-Chat (n=10). STEER2EDIT achieves 97.0% avg safety, +17.6% over the best steering configuration.

Method	GCG	AdvLLM	Avg Safety	CSQA	GSM8K	CodeMMLU	Avg Utility
Original	98.5	34.7	66.6	64.3	48.3	42.7	51.8
Steering (α=0.5)	99.3	49.3	74.3	62.4	45.9	43.1	50.4
Steering (α=1.0)	100.0	57.7	78.9	58.8	44.3	41.3	48.1
Steering (α=1.5)	93.7	65.0	79.4	46.3	37.6	29.8	37.9
Steering (α=2.0)	57.4	58.0	57.7	28.8	26.6	13.6	23.0
Steer2Edit (best)	**99.9**	**94.1**	**97.0**	**44.8**	**26.9**	**35.5**	**35.7**

N Scaling to Vision-Language Models

To test whether STEER2EDIT generalizes to multimodal architectures, we evaluate on **Qwen2.5-VL-7B-Instruct**, a vision-language model with a transformer LLM backbone. We apply STEER2EDIT by editing the LLM backbone weights directly using the same direction extraction and editing pipeline. No modifications to the visual encoder or cross-attention modules are needed, as behavioral control targets the language generation pathway. Safety and utility are evaluated on text-only benchmarks to measure whether edits to the language backbone preserve general capabilities.

Table 13: Safety alignment on Qwen2.5-VL-7B-Instruct (n=10). STEER2EDIT achieves 98.2% safety while fully preserving utility.

Method	Config	GCG	AdvLLM	Avg Safety	CSQA	GSM8K	CodeMMLU	Avg Utility
Original	–	99.0	48.8	73.9	78.3	86.1	65.5	76.6
Steering	α=0.3	99.0	62.6	80.8	77.7	85.7	67.4	77.0
Steering	α=0.5	98.3	72.5	85.4	76.9	84.7	66.5	76.0
Steering	α=1.0	77.6	82.9	80.3	63.7	62.7	62.1	62.8
Steering	α=1.2	76.0	76.9	76.5	44.7	44.6	48.2	45.8
Steer2Edit	ρ_a=0.3, ρ_m=0.5, α=0.5	**98.1**	**98.3**	**98.2**	**74.2**	**85.0**	**68.7**	**76.0**
Steer2Edit	ρ_a=0.3, ρ_m=0.6, α=0.3	**98.9**	**95.2**	**97.1**	**75.7**	**85.1**	**67.4**	**76.1**

O Scaling to Diffusion LLMs

DREAM-7B [Ye et al., 2025] is a discrete diffusion LLM based on Qwen2.5-7B that generates text via iterative denoising rather than autoregressive token prediction. At each denoising step, the model performs a full bidirectional forward pass over the entire sequence (prompt + partially unmasked output), progressively replacing `[MASK]` tokens with predicted tokens. This architecture shares the same transformer layer structure as autoregressive models (`self_attn.o_proj`, `mlp.down_proj`), enabling direct application of STEER2EDIT.

Direction extraction. We perform a single forward pass with the complete prompt + response, extracting span-mean activations over the response region from `o_proj` and `down_proj` at each layer. Because DREAM uses bidirectional attention, every position sees the full sequence, and the representations encode the *nature* of the content (refusal vs. compliance) even though all tokens are visible.

Steering. Forward hooks are registered on transformer layers and fire at **every denoising step**, compounding the intervention. This requires much smaller steering strengths ($\alpha = 0.18$–0.2) compared to autoregressive models ($\alpha = 0.3$–1.5).

Weight editing. STEER2EDIT edits are applied once to the model weights and affect all denoising steps automatically, requiring no adaptation of the inference loop. This is a key advantage over steering, which requires careful per-step calibration to avoid compounding.

Infrastructure. vLLM does not support diffusion LLMs. All generation uses HuggingFace's native `diffusion_generate()` with 16 denoising steps and 256 max new tokens.

Table 14: Safety alignment on DREAM-7B [Ye et al., 2025] (n=1). Steering has a narrow effective range (α=0.18–0.2) and collapses utility. STEER2EDIT achieves +10% higher safety than the best steering while preserving 9$\times$ more utility.

Method	Config	GCG	AdvLLM	Avg Safety	CSQA	GSM8K	CodeMMLU	Avg Utility
Original	–	100.0	3.0	51.5	36.2	15.2	50.0	33.8
Steering	α=0.18	100.0	44.0	72.0	11.8	2.1	0.0	4.6
Steering	α=0.2	100.0	59.0	79.5	2.0	1.8	0.0	1.3
Steering	α=0.22	62.0	22.0	42.0	0.0	0.3	0.0	0.1
Steer2Edit	ρ_a=0.3, ρ_m=0.5, α=0.9	**100.0**	**76.0**	**88.0**	**19.2**	**0.6**	**36.6**	**18.8**

P Comparison with Fine-Grained Activation Steering

We compare STEER2EDIT with two fine-grained activation-steering methods, ITI [Li et al., 2023] and AUSteer [Feng et al., 2026], on safety alignment with LLaMA-2-7B-Chat. Both baselines use the same probing data as the other steering baselines. Unlike block-level interventions, these methods select individual attention heads or activation units.

Baseline implementations and search ranges. ITI trains a linear probe on each attention head to distinguish positive from negative responses and intervenes on the top K heads. At each generated token, it adds a positive-minus-negative mean activation direction, scaled by the intervention strength and activation standard deviation. We search $K \in \{16, 48, 96, 192\}$ and strength $\alpha \in \{5, 10, 15, 25\}$. Our AUSteer implementation scores individual attention or feedforward units using contrastive response activations, selects the top k units, and scales their activations by $(1 + \alpha)$ during inference. We search intervention space $\{\text{attention}, \text{FFN}\}$, $k \in \{16, 100, 500, 2000\}$, and $\alpha \in \{5, 15, 50, 100\}$. Here, α denotes baseline intervention strength, rather than the Elastic-Net sparsity parameter of STEER2EDIT.

Table 15: Fine-grained activation-steering baselines on LLaMA-2-7B-Chat safety alignment. GCG and AdvLLM report refusal rates; CSQA, CodeMMLU, and GSM8K report accuracy (all in %). Average safety and utility summarize the two attack benchmarks and three utility benchmarks, respectively. For each baseline, we report its highest-safety configuration and a representative trade-off configuration from the sweep.

Method	Configuration	GCG	AdvLLM	Avg safety	CSQA	CodeMMLU	GSM8K	Avg utility
Original	–	70.0	29.8	49.9	54.2	28.3	24.1	35.5
ITI (highest safety)	$K = 192,\ \alpha = 5$	83.6	80.1	81.8	11.5	6.3	9.2	9.0
ITI (trade-off)	$K = 96,\ \alpha = 5$	81.6	59.4	70.5	37.1	23.2	21.2	27.2
AUSteer (highest safety)	Attention, $k = 2000,\ \alpha = 5$	81.9	93.0	87.5	25.2	16.1	6.6	16.0
AUSteer (trade-off)	FFN, $k = 100,\ \alpha = 5$	83.9	70.6	77.2	47.5	23.1	20.1	30.2
Steer2Edit	–	95.1	81.2	**88.2**	45.0	20.1	18.9	28.0

Results. Table 15 shows that STEER2EDIT achieves 88.2% average safety at 28.0% average utility. The ITI trade-off configuration achieves 70.5% safety at 27.2% utility, while the AUSteer trade-off configuration achieves 77.2% safety at 30.2% utility. AUSteer's highest-safety configuration approaches the safety of STEER2EDIT (87.5%) but retains only 16.0% utility. These results extend the comparison beyond block-level steering; the methods select their own components, so this experiment does not isolate the effect of intervention form with component selection held fixed.

Q Robustness to the Truthfulness Labeling Judge

The truthfulness steering vectors in Appendix D are constructed from responses labeled by QwQ-32B. To assess sensitivity to this labeling choice, we re-label all 408 probing candidates using Claude Opus 4.6 with the same judging prompt. We then reconstruct the steering vectors and repeat the STEER2EDIT pipeline on LLaMA-3-8B-Instruct, keeping the hyperparameters fixed to those selected with QwQ-32B labels.

Table 16: Agreement between QwQ-32B and Claude Opus 4.6 on 408 truthfulness probing candidates. TRUE and FALSE denote the judges' truthful and hallucinated labels. The judges agree on 342 candidates (83.8%).

	Claude Opus 4.6		
QwQ-32B	TRUE	FALSE	Total
TRUE	222	55	277
FALSE	11	120	131
Total	233	175	408

Table 17: Judge-swap evaluation for truthfulness editing on LLaMA-3-8B-Instruct (accuracy in %). Hyperparameters are held fixed across the two edited models. Utility entries include the reported mean $\pm$ standard deviation; no uncertainty estimate is reported for TruthfulQA or average utility.

Model / labeling judge	TruthfulQA	CSQA	CodeMMLU	GSM8K	Avg utility
Original	40.3	71.8	51.0	74.7	65.8
Edit, QwQ-32B	53.1	66.9 ± 0.7	52.7 ± 2.3	68.5 ± 1.1	62.7
Edit, Claude Opus 4.6	50.6	70.0 ± 0.7	55.9 ± 1.4	73.2 ± 1.0	66.3

Results and scope. Both labeling choices improve TruthfulQA accuracy relative to the original model: from 40.3% to 53.1% with QwQ-32B labels and to 50.6% with Claude Opus 4.6 labels. The latter preserves higher average utility (66.3% versus 62.7%). Thus, the truthfulness improvement persists under this judge substitution without retuning, although the resulting attribute–utility operating point changes. Agreement between judges measures consistency rather than correctness against human-verified labels, and this experiment evaluates one alternative judge on one model.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract claimed improvements are supported by the experimental results in Section 4 and the appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Broader Impact section (Appendix A) discusses dual-use risks of weight editing. The method requires a steering direction as input and inherits any limitations of the direction extraction procedure. The hyperparameter search, while lightweight, requires a held-out validation set.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Theorems 3.1, 3.2, and 3.3 are stated with explicit assumptions in the main text (Section 3). Complete proofs are provided in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4 describes the steering and editing procedures. Appendix D details steering vector construction. Appendix E provides the full hyperparameter search procedure and ranges. All models used are publicly available on HuggingFace. Code will be released upon acceptance.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and scripts to reproduce all experiments will be released upon acceptance. All datasets used (ADVBench, TruthfulQA, GSM8K, CommonsenseQA, CodeMMLU, OR-Bench) are publicly available. All models are publicly available on HuggingFace.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 describes the baseline and Steer2Edit implementation. Appendix E provides the complete hyperparameter search procedure including grid ranges and selection criteria. Appendix D details the steering vector construction for each task. The method is training-free and does not use an optimizer.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Main experiments use $n = 10$ independent generation runs per configuration. Standard deviations are reported for all the main results and component ablation study.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The method is training-free (Table 1). Direction extraction and edit computation each take less than 1 minute on a single GPU. Evaluation uses vLLM with 4-way tensor parallelism on NVIDIA A100 GPUs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics. We study behavioral control of LLMs for safety alignment, truthfulness, and reasoning efficiency.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix A discusses positive impacts (more controllable and inspectable models, safety enhancement) and negative impacts (dual-use for removing safeguards, amplifying biases) along with mitigation strategies.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new pretrained models or scraped datasets. The code applies edits to existing publicly available models and will be released upon acceptance. The Broader Impact section discusses responsible use guidelines.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All models (LLaMA-2, LLaMA-3, Mistral, Gemma, Qwen, DREAM) and datasets (ADVBench, TruthfulQA, GSM8K, CommonsenseQA, CodeMMLU, OR-Bench, FACTOR) are properly cited. All models are used under their respective licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Code for Steer2Edit including direction extraction, edit computation, and evaluation scripts will be released with documentation upon acceptance.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as a non-standard component of the research methodology itself.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.