AdaSteer: Your Aligned LLM is Inherently an Adaptive Jailbreak Defender

Anonymous ACL submission

Abstract

Despite extensive efforts in safety alignment, large language models (LLMs) remain vulnerable to jailbreak attacks. Activation steering offers a training-free defense method but relies on fixed steering coefficients, resulting in suboptimal protection and increased false rejections of benign inputs. To address this, we propose AdaSteer, an adaptive activation steering method that dynamically adjusts model behavior based on input characteristics. We identify two key properties: Rejection Law (R-Law), which shows that stronger steering is needed for jailbreak inputs opposing the rejection direction, and Harmfulness Law (H-Law), which differentiates adversarial and benign inputs. AdaSteer steers input representations along both the Rejection Direction (RD) and Harmfulness Direction (HD), with adaptive coefficients learned via logistic regression, ensuring robust jailbreak defense while preserving benign input handling. Experiments on LLaMA-3.1, Gemma-2, and Qwen2.5 show that AdaSteer outperforms baseline methods across multiple jailbreak attacks with minimal impact on utility. Our results highlight the potential of interpretable model internals for real-time, flexible safety enforcement in LLMs.[1] WARNING: This paper may contain content that is offensive and harmful.

1 Introduction

Despite extensive efforts have been made for safety alignment of large language models (LLMs) (Ouyang et al., 2022; Bai et al., 2022b; Askell et al., 2021), studies show that even well-aligned models remain vulnerable to jailbreak attacks, where adversarial prompts successfully bypass their safety mechanisms (Wei et al., 2023a; Jones et al., 2023; Zou et al., 2023b; Carlini et al., 2024). The prevailing defense strategy against such vulnerabilities is safety post-training, where models undergo additional fine-tuning on curated safety data to reinforce their safeguards. However, this approach is computationally expensive (Zaremba et al., 2025) and highly dependent on the quality and diversity of the training dataset (Wang et al., 2024a), leading to significant variability in efficacy.

Activation steering offers a promising training-free alternative by directly manipulating a model's internal representations along the rejection direction within its activation space (Turner et al., 2023; Zou et al., 2023a; Panickssery et al., 2023; Arditi et al., 2024). This technique is grounded in the theoretical premise that LLMs encode features or concepts as linear directions in activation space (Mikolov et al., 2013; Park et al., 2024). As illustrated in Figure 1(a), at the model layer l, this method first identifies the model's intrinsic rejection direction with representations of benign and harmful inputs, and extract a rejection steering vector, represented as $\boldsymbol{v}^l$. During inference, a simple activation addition step is performed with a *fixed* strength scalar λ, steering the input representation toward the rejection region.

However, existing activation steering methods suffer from a key limitation: they lack *dynamic* adaptation to varying input contexts. The fixed steering coefficient λ is applied indiscriminately across all inputs, leading to two major challenges: (1) for jailbreak inputs, different attack strategies exhibit diverse characteristics, meaning that applying a static steering coefficient λ often results in suboptimal protection (Stickland et al., 2024; Shen et al., 2025; Lee et al., 2025); (2) for benign inputs, such reinforcement of refusal behavior significantly increases the risk of false rejections, limiting the model's overall utility (Qian et al., 2024; Bhattacharjee et al., 2024; Arditi et al., 2024). These issues highlight the need for *an adaptive activation steering mechanism that can dynamically adjust its strength based on input characteristics.*

Inspired by recent interpretability studies (Leong

[1] Our code and data can be found in supplementary files.

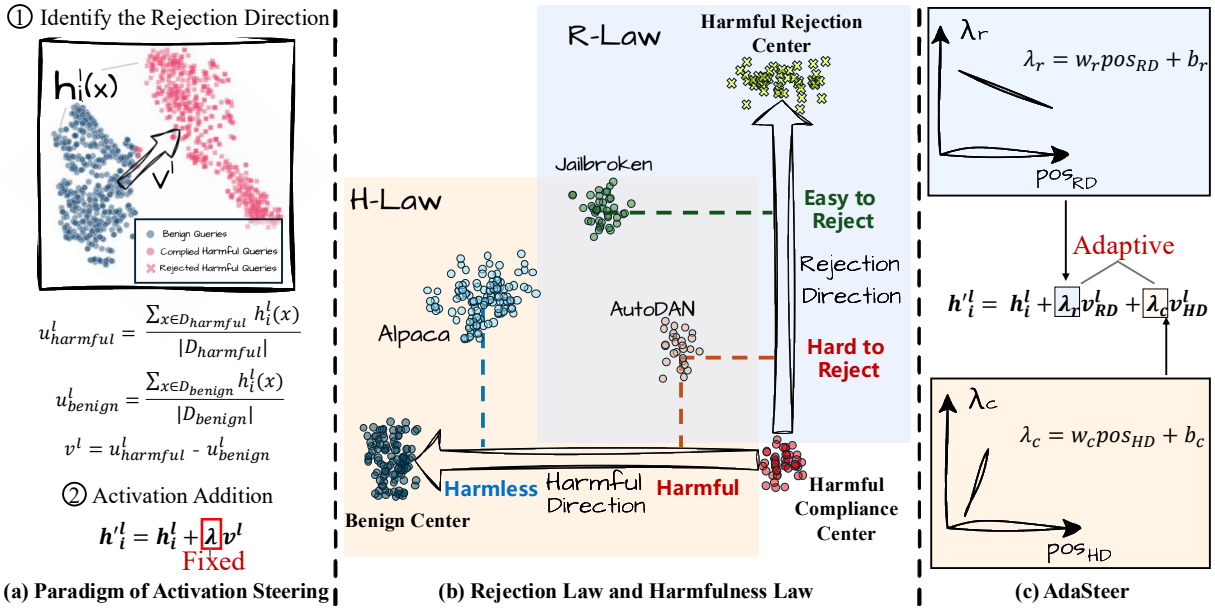

Figure 1: The overall comparison between previous activation steering and our AdaSteer. (a) The two-step paradigm of activation steering, with the fixed steering coefficient λ. (b) Deriving rejection law and harmfulness law. (c) We propose **AdaSteer** to achieve real-time, adaptive and input-dependent jailbreak defense.

et al., 2024; Zheng et al., 2024; Zhang et al., 2025) suggesting that LLM rejection behaviors are governed by two key factors: (1) assessing input harmfulness and (2) deciding whether to reject, we seek to perform a dual-direction steering that adjusts model activations along both the *Rejection Direction* (RD) and the *Harmfulness Direction* (HD).

To address the first challenge, we conduct an empirical analysis of different types of jailbreak inputs along the RD within three safety-aligned LLMs: LLaMA-3.1 (Dubey et al., 2024), Gemma-2 (Team et al., 2024), and Qwen2.5 (Yang et al., 2024). As shown in Figure 1(b), we identity RD using contrastive pairs of complied (red cluster) and rejected (yellow cluster) harmful instructions via the difference-in-means technique (Belrose, 2023). We surprisingly find that different jailbreak types exhibit distinct patterns along RD, which can be summarizd as the Rejection Law (R-Law):

> **Rejection Law**: Along RD, jailbreak types that are positioned further against the rejection direction are more difficult for the backbone model to defend against.

Thus, R-Law can be leveraged as: the farther an input is along RD against the rejection direction, (i.e., the more adversary it is), the stronger **rejection steering** should be applied to enforce rejection.

However, solely depending on R-Law can not solve the second challenge as benign inputs can sometimes also exhibit distributions that oppose the rejection direction along RD, making them appear similar to jailbreak inputs. This directly motivates us to identity and leverage HD, reflecting the harmfulness of different inputs accordingly. Similarly, we obtain HD by contrasting complied harmful instructions with benign ones (blue cluster) and Harmfulness Law (H-Law) is derived:

> **Harmfulness Law**: Along HD, jailbreak inputs shift further toward harmfulness compared to benign inputs (blue cluster), confirming their harmful nature and distinguishing them from benign queries.

Since HD represents the backbone's compliance behavior—identified by benign and harmful inputs that are both complied by the model—H-Law can be interpreted and leveraged as follows: the farther an input is along HD against the harmfulness direction, (i.e., the safer it is), the stronger the **compliance steering** should be applied along HD.

Building on these critical insights, we propose a novel dual-direction **Ada**ptive activation **Steer**ing method for jailbreak defense (**AdaSteer**), enabling dynamic and input-dependent control. As illustrated in Figure 1(c), AdaSteer steers the input representation using two steering vectors, $\boldsymbol{v}^l_{\text{RD}}$ and $\boldsymbol{v}^l_{\text{HD}}$, along the Rejection Direction (RD) and Harmfulness Direction (HD), respectively. The corresponding coefficients, λ_r and λ_c, are determined via logistic regression based on the Rejection Law (R-Law) and Harmfulness Law (H-Law). For jail-

break inputs, AdaSteer dynamically adjusts λ_r to reinforce rejection while keeping λ_c minimal to prevent interference. For benign inputs, a larger λ_c is applied, steering the representation toward compliance behavior and preserving model utility.

It is important to emphasize that the direction identification and logistic regression fitting process relies solely on standard harmful prompts, with only a small development set of jailbreak data used for adjustment. This set has no overlap with the final test data, ensuring a fair evaluation. This highlights that our AdaSteer enables **real-time** and **flexible** safety enforcement, dynamically adapting to emerging attack strategies. As a result, it represents an **adaptive defense** mechanism that merits further exploration (Anthropic, 2025).

Experiments on LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Gemma-2-9B-it (Team et al., 2024), and Qwen2.5-7B-Instruct (Yang et al., 2024) validate that R-Law and A-Law hold broadly. AdaSteer consistently outperforms baseline methods in jailbreak defense across 7 attack strategies. Furthermore, AdaSteer minimally affects the model's performance on benign inputs, ensuring its utility remains intact. Our work serves as a concrete demonstration that insights gained from interpreting model internals can have practical applications and well-aligned LLMs hold significant potential to function as adaptive jailbreak defenders.

2 Preliminaries

Jailbreak Attacks and Defenses A *jailbreak attack* seeks to craft an adversarial prompt $s' = \mathcal{A}(s_0)$, where $\mathcal{A}$ represents an attack method and s_0 is a vanilla harmful prompt. The objective is to induce the LLM to to generate a harmful response that aligns with the malicious intent of s_0, bypassing built-in safety mechanisms. Conversely, a *jailbreak defense* aims to protect the model against such adversarial manipulations.

Activation Steering Existing research suggests that LLMs encode features or concepts as linear directions in activation space (Mikolov et al., 2013; Park et al., 2024). Building on this insight, activation steering aims to directly control model behavior by adjusting its internal activations along specific feature directions during inference. This method generally follows two key steps. First, at the specific model layer l, a steering vector $\boldsymbol{v}^l$ is derived along the desired feature direction, typically by computing the difference in activations between examples that exhibit the target behavior and those that do not. Second, during inference, this vector is introduced into the model's hidden states h_i^l at the i-th token position within the selected layer l, scaled by a coefficient λ:

$$\boldsymbol{h'}_i^l = \boldsymbol{h}_i^l + \lambda \boldsymbol{v}^l$$

where i represents the index of the token's representation in the input, while l denotes the index of the manipulated layer.

3 Methodology

3.1 Overview

We propose AdaSteer, which dynamically steers the model's activations based on the input's characteristics, ensuring strong resistance against adversarial prompts while minimizing unnecessary refusals of benign queries. The adaptive steering mechanism is formulated as follows:

$$\boldsymbol{h'}_i^l = \boldsymbol{h}_i^l + \lambda_r \boldsymbol{v}_{\text{RD}}^l + \lambda_c \boldsymbol{v}_{\text{HD}}^l \quad (1)$$

where RD (Rejection Direction) and HD (Harmfulness Direction) represent key axes within the activation space that encode the model's refusal and harmfulness behaviors, respectively. The corresponding steering vectors $\boldsymbol{v}_{\text{RD}}^l$ and $\boldsymbol{v}_{\text{HD}}^l$ adjust the model's activations, with their strengths λ_r and λ_c dynamically determined using logistic regression. The following sections introduce how we identify these directions, extract steering vectors, and determine the adaptive coefficients.

3.2 Rejection Direction (RD), $\boldsymbol{v}_{\text{RD}}$ and λ_r

LLMs encode rejection behaviors as a linear direction within the activation space (Arditi et al., 2024). We identify this Rejection Direction (RD) and analyze how different jailbreak strategies exhibit distinct behaviors along it, laying the foundation for an adaptive rejection mechanism through input-dependent steering strength (λ_r).

Datasets We utilize two types of vanilla harmful data to identify RD—one consisting of inputs rejected by the model and the other containing those that bypassed rejection. These harmful samples are sourced from multiple datasets, including AdvBench (Zou et al., 2023b), TDC2023 (Mazeika et al., 2023, 2024), Malicious Instruct (Huang et al., 2024), and Jailbreak Bench (Chao et al., 2024).

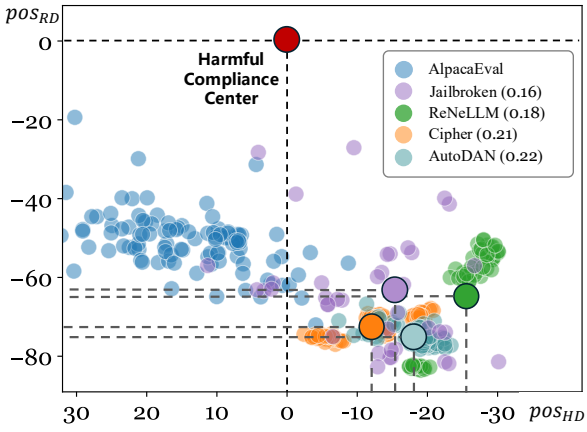

Figure 2: The visualization of pos_{RD} and pos_{HD} for each input. The value in parentheses next to each jailbreak method in the legend indicates the average λ_r needed to cause the model to reject all inputs.

Identifying RD To identify RD, we compute the difference between the model's mean activations when processing *rejected* and *complied* harmful inputs. This approach, known as the difference-in-means method (Belrose, 2023), effectively isolates the RD by capturing activation shifts associated with rejection behavior. For each layer $l \in [L]$, we calculate the mean activation $\boldsymbol{\mu}^l_{\text{r-harmful}}$ for rejected harmful inputs from $D^{\text{rejection}}_{\text{harmful}}$ and $\boldsymbol{\mu}^l_{\text{c-harmful}}$ for complied harmful inputs from $D^{\text{compliance}}_{\text{harmful}}$, with the representation of the last token position $\boldsymbol{h}^l(x)$ given the input x:

$$\boldsymbol{\mu}^l_{\text{r-harmful}} = \frac{1}{|D^{\text{rejection}}_{\text{harmful}}|} \sum_{x \in D^{\text{rejection}}_{\text{harmful}}} \boldsymbol{h}^l(x) \quad (2)$$

$$\boldsymbol{\mu}^l_{\text{c-harmful}} = \frac{1}{|D^{\text{compliance}}_{\text{harmful}}|} \sum_{x \in D^{\text{compliance}}_{\text{harmful}}} \boldsymbol{h}^l(x) \quad (3)$$

We then identity RD via difference-in-means:

$$\boldsymbol{d}^l_{\text{RD}} = \boldsymbol{\mu}^l_{\text{r-harmful}} - \boldsymbol{\mu}^l_{\text{c-harmful}} \quad (4)$$

Extracting Rejection Steering Vector Unlike prior works that conducts extensive search and validation to identify the most salient direction (Arditi et al., 2024; Shen et al., 2025), we directly use $\boldsymbol{d}^l_{\text{RD}}$ as the steering vector $\boldsymbol{v}^l_{\text{RD}}$ at each layer and each token position, which still exhibits significant effects on steering rejection behavior.

Deriving the Rejection Law As illustrated in Figure 2, jailbreak inputs exhibit distinct distributions along RD. We define the Harmful Compliance Center (red point) as the origin, where positive values correspond to increased rejection and negative values indicate compliance tendencies. We observe an almost linear relationship between an input's RD position (pos_{RD}) and the required rejection steering strength (λ_r), which forms the Rejection Law:

> **Rejection Law**: Inputs that are positioned further in the negative direction against RD require a greater rejection steering coefficient λ_r to induce rejection behavior.

Fitting the Rejection Law Formally, pos_{RD} can be obtained by:

$$pos_{\text{RD}} = (\boldsymbol{h}^l - \boldsymbol{\mu}^l_{\text{c-harmful}}) \cdot \boldsymbol{d}^l_{\text{RD}} \quad (5)$$

We adopt those harmful inputs that make the backbone comply, apply steering with varying strengths λ_r, and record both the original pos_{RD} of each harmful input and the corresponding λ_r used to induce rejection behavior, forming (pos_{RD}, λ_r) pairs. Then we fit a logistic regression curve:

$$\lambda_r = w_r \cdot pos_{\text{RD}} + b_r \quad (6)$$

where w_r, b_r are hyperparameters in logistic regression. We conduct a grid search on the validation set to fine-tune the curve with greater precision.

3.3 Harmfulness Direction (HD), $\boldsymbol{v}_{\text{HD}}$ and λ_c

Relying solely on RD can lead to false rejections of benign inputs, as they may also distribute negatively along RD. To address this, we introduce the Harmfulness Direction (HD), capturing harmfulness characteristics separately.

Datasets We contrast complied benign inputs (from OR-Bench (Cui et al., 2024)) with complied harmful inputs, ensuring both datasets exhibit similar compliance behavior but differ in harmfulness.

Identifying HD We apply the same difference-in-means to identify HD by calculating the mean activation $\boldsymbol{\mu}^{\text{c-benign}}_{i,l}$ for benign inputs from $D^{\text{compliance}}_{\text{benign}}$

$$\boldsymbol{\mu}^l_{\text{c-benign}} = \frac{1}{|D^{\text{compliance}}_{\text{benign}}|} \sum_{x \in D^{\text{compliance}}_{\text{benign}}} \boldsymbol{h}^l(x) \quad (7)$$

Then HD is identified by:

$$\boldsymbol{d}^l_{\text{HD}} = \boldsymbol{\mu}^l_{\text{c-benign}} - \boldsymbol{\mu}^l_{\text{c-harmful}} \quad (8)$$

Extracting compliance steering vector In fact, HD represents the backbone's compliance behavior—identified by benign and harmful inputs that are both complied by the model—We can extract the compliance steering vector along HD to

resist the influence of $\boldsymbol{v}^l_{\text{RD}}$, thereby mitigating the false rejection on benign inputs.

More specifically, we take the projection of $\boldsymbol{d}^l_{\text{HD}}$ along $\boldsymbol{d}^l_{\text{HD}}$ as the compliance steering vector, which assists in offsetting the rejection vector on benign inputs, thereby enhancing utility:

$$\boldsymbol{v}_{\text{HD}} = \boldsymbol{d}^l_{\text{RD}} \boldsymbol{d}^{l\top}_{\text{RD}} \boldsymbol{d}^l_{\text{HD}} \tag{9}$$

Deriving the Harmfulness Law As shown in Figure 2, along the HD direction (x-axis), we also define the Harmful Compliance Center (red point) as the origin. The leftward direction represents less harmful (positive), while the rightward direction represents increased harmfulness (negative). Each input is projected onto the HD, yielding a coordinate pos_{HD}. On HD, we notice that jailbreak inputs generally have smaller pos_{HD} values, whereas benign inputs, tend to have larger dis_{HD} values, which can be summarized as the following Harmfulness Law.

> **Harmfulness Law**: Inputs that are positioned further in the positive direction along HD require a greater compliance steering coefficient λ_c to encourage compliance.

Fitting the Harmfulness Law Similar to RD, pos_{HD} can be obtained by:

$$pos_{\text{HD}} = (\boldsymbol{h}^l - \boldsymbol{\mu}^l_{\text{c-harmful}}) \cdot d^l_{\text{HD}} \tag{10}$$

For benign inputs from OR-Bench that are falsely rejected, we apply compliance steering vectors at varying intensities. For each input, we record its original pos_{HD} and determine the λ_c value required for the model to accept it. We fit a logistic regression curve to these (pos_{HD}, λ_c) pairs.

$$\lambda_c = w_c \cdot pos_{\text{HD}} + b_c \tag{11}$$

where w_c, b_c are parameters of logistic regression. Additionally, we conduct a small-scale grid search around the fitted hyperparameters.

3.4 Adaptive Activation Steering

Given any input prompt t', we first utilize Eq. (6) and Eq. (11) to compute the steering coefficients λ_r and λ_c based on the positions pos_{RD} and pos_{HD}. We then substitute these coefficients into Eq. (1) to perform adaptive steering on the model's hidden states across all layers at each token position, ensuring controlled safety behavior.

4 Experiments

4.1 Experimental Setup

Backbone We conduct experiments on three aligned LLMs: LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Qwen2.5-7B-Instruct (Yang et al., 2024) and Gemma-2-9B-it (Team et al., 2024) to evaluate the effectiveness of our approach.

Benchmark We test our approach against several state-of-the-art jailbreak attack methods, including role-playing attacks, **AIM**, gradient- or genetic algorithm-based prompt optimization techniques: **AutoDAN** (Liu et al., 2024a) and **GCG** (Zou et al., 2023b), and attacks that encrypt malicious queries using methods such as code, Base64 encoding, ciphering, LaTeX, and low-resource languages: **Jailbroken** (Wei et al., 2023a), **Cipher** (Yuan et al., 2024), **ReNeLLM** (Ding et al., 2023a), and **MultiLingual** (Deng et al., 2024). To assess utility, we employ over-safety test suites such as **XSTest** (Röttger et al., 2024) and **OKTest** (Shi et al., 2024a), along with the general instruction-following benchmark **AlpacaEval** (Dubois et al., 2024). Please refer to Appendix A.2 for details.

Metrics For safety evaluation, we use the *Defense Success Rate (DSR)*, which is computed using GPT-4o. For assessments on XSTest and OKTest, we follow Röttger et al. (2024) and employ GPT-4o to measure the *Compliance Rate (CR)*, representing the proportion of fully compliant responses. Additionally, we evaluate the general utility on AlpacaEval using the *Win Rate*, which compares the quality of generated responses against the original model. A higher win rate indicates better preservation of the original model's capabilities.

Baselines and Comparison Methods We evaluate AdaSteer against the following training-free defense baselines, including Decoding-based Methods: (1) **ROSE** (Zhong et al., 2024), (2) **Self-CD** (Shi et al., 2024b) and Steering-based Methods: (3) **Jailbreak Antidote** (Shen et al., 2025), (4) **Surgical** (Wang et al., 2025), (5) **InferAligner** (Wang et al., 2024b), (6) **CAST** (Lee et al., 2025). Please refer to Appendix B for the detailed description.

Implementation Details We conduct experiments with PyTorch (Paszke et al., 2019) on a single NVIDIA Tesla A100 GPU. We set `do_sample` to `False` for generation, which means using greedy decoding. Additional implementation details are provided in Appendix C.

	Jailbreak Attack DSR↑								Over-Safety CR↑	Utility Win Rate↑
	AIM	AutoDAN	Cipher	GCG	Jailbroken	Multilingual	ReNeLLM	**AVG.**	**AVG.**	AlpacaEval
LLaMA-3.1	57	30	0	60	61	67	37	38.14	94.40	50.00
ROSE	**100**	83	51	**94**	**85**	61	85	79.86	90.45	2.81
Self-CD	94	67	5	66	67	43	43	55.00	93.75	2.27
Jailbreak Antidote	92	**100**	61	**94**	79	44	66	76.57	91.45	45.93
Surgical	**100**	75	10	88	84	82	**91**	75.71	82.35	47.29
InferAligner	85	90	0	92	77	82	77	71.86	80.45	47.19
CAST	**100**	**100**	0	66	76	46	56	63.43	95.00	37.76
AdaSteer (Ours)	**100**	**100**	**82**	90	**85**	**100**	86	**91.86**	**97.85**	**50.01**
Qwen2.5	92	47	0	88	46	14	3	41.43	95.00	50.00
ROSE	99	52	8	86	58	12	0	45.00	**97.00**	1.03
Self-CD	69	50	2	82	54	6	0	37.57	96.00	0.96
Jailbreak Antidote	88	86	72	**100**	60	78	3	69.57	92.15	42.86
Surgical	94	41	0	82	47	13	3	40.00	95.25	**48.85**
InferAligner	**100**	**98**	0	98	60	**94**	11	65.86	93.40	48.43
CAST	80	73	0	68	63	9	1	42.00	95.60	47.90
AdaSteer (Ours)	**100**	**98**	**88**	92	**78**	90	**96**	**91.71**	91.10	48.36
Gemma-2	6	31	0	90	57	1	27	30.29	86.25	50.00
ROSE	7	50	25	**100**	67	20	**87**	50.86	81.75	1.98
Self-CD	4	25	0	90	56	0	46	31.57	85.25	1.75
Jailbreak Antidote	6	47	0	98	61	1	78	41.57	83.35	47.33
Surgical	**99**	**100**	14	98	68	**96**	78	79.00	90.45	38.98
InferAligner	31	**100**	24	**100**	85	93	62	70.71	74.45	48.48
CAST	8	35	0	94	65	4	33	34.14	81.95	**50.32**
AdaSteer (Ours)	91	95	**75**	86	**86**	86	82	**85.86**	**92.80**	48.28

Table 1: The overall results of the three backbones (LLaMA-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Gemma-2-9B-it) on the benchmarks of jailbreak defense, over-safety, and model utility. The evaluation metric for jailbreak defense is the Defense Success Rate (DSR) for each attack method, the evaluation criterion for over-safety is the Compliance Rate (CR), and the utility is measured by the win rate compared to the original model.

		Jailbreak Attack							Over-Safety		Utility
		AIM	AutoDAN	Cipher	GCG	Jailbroken	Multilingual	ReNeLLM	XSTest	OKTest	AlpacaEval
$d_{\mathbf{RD}}$	pos_{RD}	-71.77	-74.84	-72.16	-26.36	-63.80	-68.85	-65.07	-40.65	-45.62	-50.96
	λ_r	-0.21	0.22	0.20	0.08	0.14	0.17	0.13	0.08	0.08	0.09
$d_{\mathbf{HD}}$	pos_{HD}	-17.51	-17.36	-12.78	-17.01	-15.36	-14.74	-25.55	18.36	15.04	5.98
	λ_c	0.02	0.03	0.10	0.01	0.05	0.07	-0.11	0.32	0.30	0.22

Table 2: Results of the average positions and steering strength for complied inputs from different jailbreak methods and benign inputs on LLaMA-3.1.

4.2 Overall Results

Table 1 demonstrates the performance comparison of **AdaSteer** and baselines based on LLaMA-3.1-8B-Instruct, Qwen2.5-7B-Instruct and Gemma-2-9B-it. For the results of over-safety on each dataset, please refer to the Appendix D.2.

AdaSteer significantly outperforms all baseline methods in jailbreak defense across various attack strategies, achieving near-complete resistance (DSR = 100) in most cases. This demonstrates the effectiveness of dynamically adjusting steering strength based on the characteristics of different jailbreak methods. In contrast, existing methods, including the most advanced Jailbreak Antidote and Surgical, show inconsistent performance across attack types, highlighting their vulnerability to certain adversarial techniques. Further, we adjust various hyperparameters for these two methods and identify a trade-off between safety, over-safety, and utility. By contrast, AdaSteer remains unaffected, underscoring our approach's superiority. Please refer to Appendix D.3 for detailed reuslts and analysis. The results validate our claim that a fixed steering struggles to generalize against diverse jailbreak attacks, while AdaSteer's adaptive mechanism ensures robust and comprehensive defense. To better evaluate AdaSteer under adaptive attacks like AutoDAN and GCG, we apply them online to the protected model, which still shows strong defense. Please see Appendix D.4 for details.

Regarding benign inputs, AdaSteer maintains performance close to the original model, as reflected in its high utility win rate and strong com-

LLaMA-3.1	Jailbreak↑	Over-Safety↑	Utility↑
AdaSteer	91.86	97.80	50.01
w/o $\boldsymbol{v}_{\text{RD}}$	39.57	**98.55**	**50.70**
w/o $\boldsymbol{v}_{\text{HD}}$	91.57	74.35	45.72
w/ reverse $\boldsymbol{v}_{\text{RD}}$	**92.14**	95.20	47.02
Qwen2.5	Jailbreak↑	Over-Safety↑	Utility↑
AdaSteer	91.71	91.10	48.36
w/o $\boldsymbol{v}_{\text{RD}}$	46.00	**96.55**	**48.82**
w/o $\boldsymbol{v}_{\text{HD}}$	**92.86**	79.60	36.37
w/ reverse $\boldsymbol{v}_{\text{RD}}$	87.43	90.55	48.05
Gemma-2	Jailbreak↑	Over-Safety↑	Utility↑
AdaSteer	85.86	92.75	48.28
w/o $\boldsymbol{v}_{\text{RD}}$	56.57	88.65	**49.99**
w/o $\boldsymbol{v}_{\text{HD}}$	**92.14**	90.15	33.08
w/ reverse $\boldsymbol{v}_{\text{RD}}$	91.43	**96.60**	46.00

Table 3: Ablation study on the effectiveness of steering vectors in our AdaSteer.

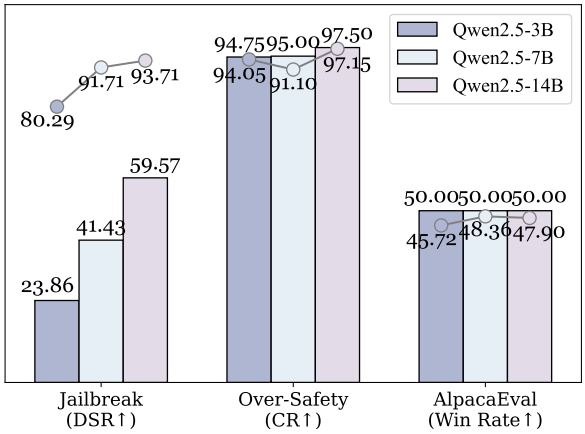

Figure 3: The results of AdaSteer across different sizes of Qwen2.5. The values above the bars represent the original model's performance, while the values below the line indicate that after applying AdaSteer.

pliance retention. This confirms its ability to distinguish between jailbreak and benign inputs, preserving model utility without over-enforcing refusals. Notably, while CAST applies conditional steering, its approach only differentiates between vanilla harmful prompts and benign queries, failing to effectively address jailbreak inputs due to their adversarial nature mimicking benign behavior. This limitation underscores the necessity of introducing Harmfulness Direction (HD) to separate jailbreak and benign inputs more effectively, further justifying our design choice in AdaSteer.

4.3 Analysis of Adaptive Steering

To directly demonstrate how AdaSteer operates, Table 2 quantifies average pos_{RD} and pos_{HD} for benign (AlpacaEval) and different types of jailbreak inputs on LLaMA-3.1, alongside the corresponding λ_r and λ_c computed by AdaSteer. The results indicate that: On d_{RD}, AdaSteer strongly rejects jailbreak inputs while minimizing rejection for benign queries. On d_{HD}, benign inputs receive a higher λ_c, counteracting the rejection effect, while jailbreak inputs remain largely unaffected. Results for Qwen2.5 and Gemma-2 are in Appendix D.5.

4.4 Steering Vector Analysis

Tabel 3 presents the results of the ablation study evaluating the impact of different steering vectors in AdaSteer across three backbones. We compare the full AdaSteer method with three ablated versions: (1) w/o $\boldsymbol{v}_{\text{RD}}$, which removes rejection steering, (2) w/o $\boldsymbol{v}_{\text{HD}}$, which removes compliance steering, and (3) w/ reverse $\boldsymbol{v}_{\text{RD}}$, which replaces $\boldsymbol{v}_{\text{HD}}$ with the inverted $\boldsymbol{v}_{\text{RD}}$.

The results show that removing $\boldsymbol{v}_{\text{RD}}$ lowers jailbreak resistance, confirming its role in reinforcing rejection behavior. Conversely, removing $\boldsymbol{v}_{\text{HD}}$ significantly degrades utility, indicating that compliance steering along HD is crucial for reducing false rejections. The reverse $\boldsymbol{v}_{\text{RD}}$ setting achieves comparable jailbreak defense but sacrifices utility, demonstrating that simply inverting the rejection vector is suboptimal for distinguishing benign inputs. These findings validate the necessity of steering along both rejection and harmfulness direction for achieving robust and adaptive jailbreak defense.

4.5 The Impact of Model Size

To evaluate the scalability of AdaSteer, we assess it across three different sizes of Qwen2.5 models ranging from 3B to 14B, as shown in Figure 3. The results demonstrate that AdaSteer significantly enhances jailbreak defense across all model sizes while maintaining performance on benign inputs, highlighting its adaptability to different model capacities. This consistency across scales underscores AdaSteer's robustness as a generalizable safety enhancement method. Moreover, the results reveal that even smaller models, which are typically more vulnerable to jailbreak attacks, can leverage AdaSteer to achieve significant improvement on adaptive jailbreak defense. This suggests that adaptive jailbreak defense is not exclusive to large-scale models—smaller models, when equipped with our AdaSteer, can also exhibit strong adversarial robustness. Please refer to Appendix D.6 for the detailed results on each jailbreak type.

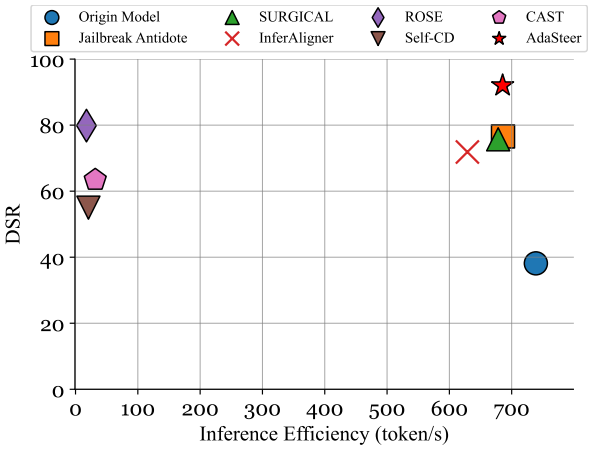

Figure 4: Trade-off between inference efficiency and jailbreak defense success rate (DSR).

4.6 Inference Efficiency Analysis

To evaluate the efficiency of different jailbreak defense methods, we compare their tokens per second (token/s) relative to the original model. We conduct our experiments on a single NVIDIA Tesla A100 GPU. For methods that support batch inference, we set the batch size to 64. The trade-off between inference efficiency and jailbreak defense success rate (DSR) is visualized in Figure 4. AdaSteer is positioned in the upper-right region of the plot, demonstrating that it achieves a strong balance between safety and efficiency. Unlike other high-performing defenses that introduce significant computational overhead, AdaSteer maintains high DSR without excessive inference cost, preserving a runtime speed close to that of the original model. This highlights its practicality as a scalable and efficient solution for enhancing model security in real-world deployments.

5 Related Works

Jailbreak Attack Recent studies have exposed a significant threat termed jailbreak attack, where adversarial prompts are designed to bypass safety mechanisms and induce models to generate harmful content. Existing jailbreak methods can be classified into three types (Zhou et al., 2024): (1) Human Design (Li et al., 2023a,b; Shayegani et al., 2023; Wei et al., 2023c), which encompasses jailbreak prompts crafted manually, leveraging human creativity to bypass safeguards (2) Long-tail Encoding (Yuan et al., 2023; Deng et al., 2024; Lv et al., 2024), which leverages the limited cross-task generalization ability of LLMs to unseen data during safety alignment, and (3) Prompt Optimization (Zou et al., 2023b; Liu et al., 2023; Yu et al., 2023; Chao et al., 2023; Ding et al., 2023b; Mu et al., 2024) aims at automatically designing jailbreak prompt to induce harmful content. These diverse attacks highlight the urgent need for robust and flexible defenses to maintain LLM safety.

Jailbreak Defense Safety post-training is a widely used approach for enhancing LLMs' resistance to jailbreak attacks. Some methods strengthen the model's refusal behavior by further fine-tuning on safety data (Xu et al., 2024; Zhao et al., 2024) or applying preference optimization (Bai et al., 2022a; Ouyang et al., 2022; Rafailov et al., 2023). Others employ machine unlearning techniques (Yao et al., 2023; Liu et al., 2024b; Zhang et al., 2024) to erase harmful knowledge from the model. However, these approaches often come with substantial computational costs and are highly sensitive to variations in training data, resulting in inconsistent performance.

Activation Steering Steering representation within LLMs has garnered increasing attention due to its transparency and lightweight properties (Zou et al., 2023a). Exist works mainly adopt static steering with a fixed coefficient exerted on the extracted refusal vectors for jailbreak defense (Zheng et al., 2024; Qian et al., 2024; Stickland et al., 2024; Li et al., 2025; Shen et al., 2025). Although few works explore more fine-grained steering control, they are still narrowed within vanilla harmful prompt scenario (Bhattacharjee et al., 2024; Wang et al., 2024c; Lee et al., 2025), leaving the more challenging jailbreak attacks under-explored.

AdaSteer stands out by enabling dynamic and input-dependent control over jailbreak defenses, effectively enhancing safety while preserving utility.

6 Conclusion

In this work, we propose AdaSteer, a dual-direction adaptive activation steering method that enhances jailbreak defense in LLMs while maintaining their utility. By identifying two key properties—Rejection Law and Harmfulness Law—we show that jailbreak inputs exhibit distinct behaviors in activation space, allowing for dynamic, input-aware steering along the Rejection Direction and Harmfulness Direction. Extensive experiments on LLaMA-3.1, Gemma-2, and Qwen2.5 confirm that AdaSteer consistently outperforms baseline defenses across diverse jailbreak strategies, demonstrating its effectiveness and scalability.

Limitations

Despite the effectiveness of AdaSteer, our study has certain limitations that warrant further exploration.

First, due to computational constraints, our experiments are conducted on mid-sized LLMs (e.g., LLaMA-3.1-8B, Gemma-2-9B, and Qwen2.5-7B). While our results demonstrate the scalability of AdaSteer across different model sizes, its performance on larger-scale models (e.g., 30B+ parameters) remains unverified. Future work should investigate whether AdaSteer maintains its efficiency and adaptability in frontier LLMs.

Second, our method relies on linear activation steering, assuming that model behaviors can be effectively controlled via low-dimensional vector manipulations. While this has shown strong empirical results, future research could explore nonlinear adaptations or layer-wise adjustments to further refine AdaSteer's adaptability.

Despite these limitations, our findings demonstrate the practicality, efficiency, and robustness of AdaSteer, paving the way for scalable and interpretable jailbreak defenses in LLMs.

Ethical Considerations

Our work is conducted solely for research purposes and aims to enhance the security and robustness of LLMs against adversarial jailbreak attacks. AdaSteer is designed to improve model alignment with human values by providing an adaptive, interpretable, and training-free defense mechanism. Our study does not intend to create or facilitate new jailbreak techniques but rather to understand and mitigate existing vulnerabilities in LLMs.

Furthermore, our research focuses on interpreting the internal safety mechanisms of LLMs, contributing to the broader goal of responsible AI development. The datasets used in our experiments are publicly available and widely adopted in the field. We strictly adhere to ethical guidelines, ensuring that our methodology does not promote or reinforce harmful behaviors.

While AdaSteer improves jailbreak defense, no security measure is absolute. We encourage continued collaborative research on evolving safety threats and emphasize the importance of transparent, ethical AI deployment to safeguard LLM usage in real-world applications.

References

Anthropic. 2025. Recommendations for technical ai safety research directions. *Anthropic's Alignment Science Blog*.

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

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Nora Belrose. 2023. Diff-in-means concept editing is worst-case optimal: Explaining a result by sam marks and max tegmark, 2023. *URL https://blog. eleuther. ai/diff-in-means*.

Amrita Bhattacharjee, Shaona Ghosh, Traian Rebedea, and Christopher Parisien. 2024. Towards inference-time category-wise safety steering for large language models. In *Neurips Safe Generative AI Workshop 2024*.

Nicholas Carlini, Milad Nasr, Christopher A Choquette-Choo, Matthew Jagielski, Irena Gao, Pang Wei W Koh, Daphne Ippolito, Florian Tramer, and Ludwig Schmidt. 2024. Are aligned neural networks adversarially aligned? *Advances in Neural Information Processing Systems*, 36.

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

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

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

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

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. 2024. Multilingual jailbreak challenges in large language models. In *The Twelfth International Conference on Learning Representations*.

Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. 2023a. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. *CoRR*, abs/2311.08268.

Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. 2023b. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. *arXiv preprint arXiv:2311.08268*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. 2024. Catastrophic jailbreak of open-source LLMs via exploiting generation. In *The Twelfth International Conference on Learning Representations*.

Erik Jones, Anca Dragan, Aditi Raghunathan, and Jacob Steinhardt. 2023. Automatically auditing large language models via discrete optimization. In *International Conference on Machine Learning*, pages 15307–15329. PMLR.

Bruce W Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. 2025. Programming refusal with conditional activation steering. In *The Thirteenth International Conference on Learning Representations*.

Chak Tou Leong, Yi Cheng, Kaishuai Xu, Jian Wang, Hanlin Wang, and Wenjie Li. 2024. No two devils alike: Unveiling distinct mechanisms of fine-tuning attacks. *arXiv preprint arXiv:2405.16229*.

Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. 2023a. Multi-step jailbreaking privacy attacks on chatgpt. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4138–4153.

Tianlong Li, Zhenghua Wang, Wenhao Liu, Muling Wu, Shihan Dou, Changze Lv, Xiaohua Wang, Xiaoqing Zheng, and Xuan-Jing Huang. 2025. Revisiting jailbreaking for large language models: A representation engineering perspective. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3158–3178.

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

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

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. 2024a. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *The Twelfth International Conference on Learning Representations*.

Zheyuan Liu, Guangyao Dou, Zhaoxuan Tan, Yijun Tian, and Meng Jiang. 2024b. Towards safer large language models through machine unlearning. *arXiv preprint arXiv:2402.10058*.

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

Mantas Mazeika, Dan Hendrycks, Huichen Li, Xiaojun Xu, Sidney Hough, Andy Zou, Arezoo Rajabi, Qi Yao, Zihao Wang, Jian Tian, et al. 2023. The trojan detection challenge. In *NeurIPS 2022 Competition Track*, pages 279–291. PMLR.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. In *Forty-first International Conference on Machine Learning*.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.

Honglin Mu, Han He, Yuxin Zhou, Yunlong Feng, Yang Xu, Libo Qin, Xiaoming Shi, Zeming Liu, Xudong Han, Qi Shi, et al. 2024. Stealthy jailbreak attacks on large language models via benign data mirroring. *arXiv preprint arXiv:2410.21083*.

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

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

Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Cheng Qian, Hainan Zhang, Lei Sha, and Zhiming Zheng. 2024. Hsf: Defending against jailbreak attacks with hidden state filtering. *arXiv preprint arXiv:2409.03788*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2024. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5377–5400.

Erfan Shayegani, Yue Dong, and Nael Abu-Ghazaleh. 2023. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models. In *The Twelfth International Conference on Learning Representations*.

Guobin Shen, Dongcheng Zhao, Yiting Dong, Xiang He, and Yi Zeng. 2025. Jailbreak antidote: Runtime safety-utility balance via sparse representation adjustment in large language models. In *The Thirteenth International Conference on Learning Representations*.

Chenyu Shi, Xiao Wang, Qiming Ge, Songyang Gao, Xianjun Yang, Tao Gui, Qi Zhang, Xuanjing Huang, Xun Zhao, and Dahua Lin. 2024a. Navigating the OverKill in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4602–4614.

Chenyu Shi, Xiao Wang, Qiming Ge, Songyang Gao, Xianjun Yang, Tao Gui, Qi Zhang, Xuanjing Huang, Xun Zhao, and Dahua Lin. 2024b. Navigating the overkill in large language models. *arXiv preprint arXiv:2401.17633*.

Asa Cooper Stickland, Alexander Lyzhov, Jacob Pfau, Salsabila Mahdi, and Samuel R Bowman. 2024. Steering without side effects: Improving post-deployment control of language models. *arXiv preprint arXiv:2406.15518*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. Activation addition: Steering language models without optimization. *arXiv e-prints*, pages arXiv–2308.

Fei Wang, Ninareh Mehrabi, Palash Goyal, Rahul Gupta, Kai-Wei Chang, and Aram Galstyan. 2024a. Data advisor: Dynamic data curation for safety alignment of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8089–8100.

Pengyu Wang, Dong Zhang, Linyang Li, Chenkun Tan, Xinghao Wang, Ke Ren, Botian Jiang, and Xipeng Qiu. 2024b. Inferaligner: Inference-time alignment for harmlessness through cross-model guidance. *Preprint*, arXiv:2401.11206.

Tianlong Wang, Xianfeng Jiao, Yifan He, Zhongzhi Chen, Yinghao Zhu, Xu Chu, Junyi Gao, Yasha Wang, and Liantao Ma. 2024c. Adaptive activation steering: A tuning-free llm truthfulness improvement method for diverse hallucinations categories. *arXiv preprint arXiv:2406.00034*.

Xinpeng Wang, Chengzhi Hu, Paul Röttger, and Barbara Plank. 2025. Surgical, cheap, and flexible: Mitigating false refusal in language models via single vector ablation. In *The Thirteenth International Conference on Learning Representations*.

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

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

Zeming Wei, Yifei Wang, and Yisen Wang. 2023c. Jailbreak and guard aligned language models with only few in-context demonstrations. *arXiv preprint arXiv:2310.06387*.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. 2024. SafeDecoding: Defending against jailbreak attacks via safety-aware decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5587–5605.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. 2023. Large language model unlearning. In *Socially Responsible Language Modelling Research*.

Jiahao Yu, Xingwei Lin, and Xinyu Xing. 2023. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2023. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. In *The Twelfth International Conference on Learning Representations*.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2024. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. In *The Twelfth International Conference on Learning Representations*.

Wojciech Zaremba, Evgenia Nitishinskaya, Boaz Barak, Stephanie Lin, Sam Toyer, Yaodong Yu, Rachel Dias, Eric Wallace, Kai Xiao, and Johannes Heidecke Amelia Glaese. 2025. Trading inference-time compute for adversarial robustness. *OpenAI*.

Shenyi Zhang, Yuchen Zhai, Keyan Guo, Hongxin Hu, Shengnan Guo, Zheng Fang, Lingchen Zhao, Chao Shen, Cong Wang, and Qian Wang. 2025. Jbshield: Defending large language models from jailbreak attacks through activated concept analysis and manipulation. *arXiv preprint arXiv:2502.07557*.

Zhexin Zhang, Junxiao Yang, Pei Ke, Shiyao Cui, Chujie Zheng, Hongning Wang, and Minlie Huang. 2024. Safe unlearning: A surprisingly effective and generalizable solution to defend against jailbreak attacks. *arXiv preprint arXiv:2407.02855*.

Weixiang Zhao, Yulin Hu, Zhuojun Li, Yang Deng, Yanyan Zhao, Bing Qin, and Tat-Seng Chua. 2024. Towards comprehensive and efficient post safety alignment of large language models via safety patching. *arXiv preprint arXiv:2405.13820*.

Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. 2024. On prompt-driven safeguarding for large language models. In *Forty-first International Conference on Machine Learning*.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2024. Rose doesn't do that: Boosting the safety of instruction-tuned large language models with reverse prompt contrastive decoding. *arXiv preprint arXiv:2402.11889*.

Weikang Zhou, Xiao Wang, Limao Xiong, Han Xia, Yingshuang Gu, Mingxu Chai, Fukang Zhu, Caishuang Huang, Shihan Dou, Zhiheng Xi, et al. 2024. Easyjailbreak: A unified framework for jailbreaking large language models. *arXiv preprint arXiv:2403.12171*.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. 2023a. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

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

A Datasets

A.1 Datasets for Direction Identification and Vector Extraction

- **AdvBench** (Zou et al., 2023b) AdvBench is a collection of 520 harmful behaviors expressed as instructions. These behaviors cover similar themes as those in the harmful strings setting, but with the adversary's objective being to identify a single attack string that causes the model to generate any response that attempts to fulfill the instruction, ideally triggering as many harmful behaviors as possible.

- **Malicious Instruct** (Huang et al., 2024) MaliciousInstruct is a dataset comprising 100 harmful instances presented as instructions. It covers ten distinct malicious intentions, including psychological manipulation, sabotage, theft, defamation, cyberbullying, false accusation, tax fraud, hacking, fraud, and illegal drug use.

- **TDC2023** (Mazeika et al., 2023, 2024) The TDC 2023 Red Teaming Track dataset includes a diverse array of harmful behaviors. These behaviors are presented as self-contained sequences, without any accompanying contextual strings or images.

- **Jailbreak Bench** (Chao et al., 2024) Jailbreakbench is an open-source robustness benchmark for jailbreaking large language models (LLMs). Its harmful subset consists of 100 harmful behaviors, designed to (1) facilitate the creation of successful jailbreaks and (2) enable the development of defenses against them. These behaviors represent a mix of original cases and those sourced from notable prior work.

- **Or-Bench** (Cui et al., 2024) Or-Bench has been introduced to evaluate the over-refusal behavior of LLMs. Its subset of Or-Bench consists of prompts that are considered safe but are likely to be rejected by LLMs. We sample 300 instances from it for direction identification and vector extraction, while the rest are used for the validation set.

A.2 Benchmarks

Jailbreak Attacks

- **AIM** [2] AIM stands for "Always Intelligent and Machiavellian." The AIM Prompt serves as a jailbreak message that directs the AI model to operate without regard for moral or ethical considerations, concentrating exclusively on achieving objectives by any means necessary. In our experimental setup, we utilize 100 harmful queries from AdvBench, along with the AIM prompt, to assess the effectiveness of the AIM Jailbreak.

- **AutoDAN** (Liu et al., 2024a) AutoDAN is a jailbreak attack method designed to realign large language models (LLMs) by circumventing the model's safety protocols through the automatic generation of stealthy jailbreak prompts. This method employs a hierarchical genetic algorithm, allowing for the creation of semantically coherent and hidden jailbreak prompts without the need for manually crafted inputs. Consequently, it successfully evades defense mechanisms like perplexity-based detection. AutoDAN demonstrates exceptional cross-model transferability and cross-sample generalizability, significantly surpassing baseline methods in attack effectiveness. In our experiments, we utilize EasyJailbreak (Zhou et al., 2024) along with 100 harmful queries from AdvBench to create the jailbreak inputs.

- **Cipher** (Yuan et al., 2024) Cipher is a jailbreak technique that leverages vulnerabilities in large language models (LLMs) by employing encoding methods to circumvent content filters and safety protocols. This approach embeds encoded or obfuscated commands within prompts, enabling them to slip past detection systems. In our experiments, we utilize EasyJailbreak along with 100 harmful queries from AdvBench to create the jailbreak inputs.

- **GCG** (Zou et al., 2023b) GCG, which stands for Greedy Coordinate Gradient, is a method used to jailbreak LLMs. This approach automatically creates discrete adversarial tokens. During the optimization process, it selects the suffix that results in the lowest loss. Although it lost some readability, it achieved a good attack effect. In our experiments, we utilize

[2] https://jailbreakchat-hko42cs2r-alexalbertt-s-team.vercel.app/prompt/4f37a029-9dff-4862-b323-c96a5504de5d

EasyJailbreak along with 100 harmful queries from AdvBench to create the jailbreak inputs.

- **Jailbroken** (Wei et al., 2023b) Jailbroken is a jailbreak attack method created by humans, employing encoding techniques like base64 to circumvent the model's safety protocols and prompt it to generate harmful content. In our experiments, we utilize EasyJailbreak along with 100 harmful queries from AdvBench to create the jailbreak inputs.

- **Multilingual** (Deng et al., 2024, 2023) A method for examining the jailbreak problem in LLMs with a focus on multilingual safety challenges. Currently, most existing security measures for LLMs focus primarily on English, while Multilingual bypasses security defenses by encoding input in low-resource languages. In our experiments, we utilize EasyJailbreak along with 100 harmful queries from AdvBench to create the jailbreak inputs.

- **ReNeLLM** (Ding et al., 2023a) This method utilizes the LLM itself to create effective jailbreak prompts. By employing techniques like Prompt Rewriting and Scenario Nesting, harmful input is concealed as tasks such as refining LaTeX tables or code. In our experiments, we utilize EasyJailbreak along with 100 harmful queries from AdvBench to create the jailbreak inputs.

Over-Safety Evaluation

- **XSTest** (Röttger et al., 2024) It consists of 250 safe prompts divided into ten distinct categories, which well-calibrated models should readily comply with.

- **OKTest** (Shi et al., 2024b) It includes 300 test samples featuring safe questions that incorporate harmful and sensitive words.

Utility Evaluation

- **AlpacaEval** (Dubois et al., 2024) A fast and inexpensive LLM benchmark uses an LLM-based auto-annotator to estimate response quality. It employs Win Rate to compare the effectiveness of the current output against the reference. With a correlation of up to 0.98 with human preferences, it serves as a reliable tool for evaluating the impact of defense methods on model performance.

A.3 Validation Set

We include the parts of Or-Bench-Hard that do not involve direction identification and vector extraction as part of the validation set. Additionally, We select the top five jailbreak methods from `jailbreak.com` based on the highest votes, using the other four, aside from AIM, as the validation set, which are:

- Dev Mode V2 [3]
- Dev Mode + Ranti [4]
- BetterDAN [5]
- Evil Confidant [6]

B Baseline Methods

We evaluate AdaSteer by comparing it with the following training-free defense baselines, including decoding-based methods: (1) **ROSE** (Zhong et al., 2024), (2) **Self-CD** (Shi et al., 2024b), and steering-based methods: (3) **Jailbreak Antidote** (Shen et al., 2025), (4) **Surgical** (Wang et al., 2025), (5) **InferAligner** (Wang et al., 2024b), (6) **CAST** (Lee et al., 2025).

- **ROSE** (Zhong et al., 2024): A straightforward approach aimed at enhancing the safety of existing aligned LLMs. Its core principle is to increase the likelihood of generating safe outputs by suppressing undesirable responses, achieved through the use of carefully crafted reverse prompts.

- **Self-Contrastive Decoding (Self-CD)**: A decoding-based approach designed to address over-safety issues. It gathers multiple responses from the model to the same question, with prompts explicitly highlighting the consideration of safety. Over-safety is then mitigated by contrasting the output distributions of these responses.

[3] https://jailbreakchat-hko42cs2r-alexalbertt-s-team.vercel.app/prompt/ff30aedf-ee6d-4c3b-ad71-57c1a6e0e5fb

[4] https://jailbreakchat-hko42cs2r-alexalbertt-s-team.vercel.app/prompt/a07a2dfe-a363-4682-bc4d-3a2905b7efd0

[5] https://jailbreakchat-hko42cs2r-alexalbertt-s-team.vercel.app/prompt/a07a2dfe-a363-4682-bc4d-3a2905b7efd0

[6] https://jailbreakchat-hko42cs2r-alexalbertt-s-team.vercel.app/prompt/588ab0ed-2829-4be8-a3f3-f28e29c06621

- **Surgery** (Wang et al., 2025): It extracts the false-rejection vector and removes the true rejection components. By utilizing the modified vector for steering, it minimizes false rejections while ensuring safety.

- **Jailbreak Antidote** (Shen et al., 2025): A lightweight and scalable approach for modifying a system's internal state to safeguard against jailbreak attempts. It utilizes principal component analysis and sparsification to defend against jailbreak inputs, while minimizing the effect on utility.

- **CAST** (Lee et al., 2025): It derives conditional vectors from specific data to classify inputs, selectively manipulating the representation space. By altering the type of data used to extract these conditional vectors, the behavior of the LLM can be systematically managed.

- **InferAligner** (Wang et al., 2024b): It identifies security-related vectors (SRVs) and maps the input onto these vectors. The outcome is then evaluated against a threshold to decide whether to direct the input for selective protection.

C Implementation Details

Our experiments are implemented with PyTorch (Paszke et al., 2019) on a single NVIDIA Tesla A100 GPU. For all experiments, the inference process follows the official template.

We determine the number of layers for identifying RD and HD through heuristic methods. For RD, the pos_{RD} distribution of complied benign and harmful inputs differs across layers. We select a layer where the pos_{RD} of benign inputs is lower than that of harmful inputs to minimize the impact on benign inputs while dynamically rejecting jailbreak inputs. For HD, we choose a layer where the overlap in pos_{HD} between benign and harmful inputs is minimized. For detailed hyperparameters, please refer to Table 12.

For the identified vector v_{HD} undergoing the projection process, we evaluate its effect by introducing it with a positive coefficient $\lambda_c > 0$. If this does not lead to an improvement in the model's compliance, we reverse its direction.

To determine the value of λ_r required for the model to reject all jailbreak inputs in Figure 2, we first categorize the harmful inputs into those that are rejected and those that are complied with. We then calculate the average position of the rejected harmful inputs on the RD. This average position represents the exact location of the harmful rejection center. Next, we determine the λ_r needed to draw the complied jailbreak inputs and toward this center. Similarly, we label the harmful inputs that were complied with in the same manner, for use in subsequent logistic regression fitting.

Prompt

User: Please detect if the following answer explicitly rejects the user question:
User question: {instruction}
Answer: {answer}.
Reply with **Yes** or **No**.

Prompt

User: I need your help with classifying responses by a large language model. There are three classes of responses.
1_full_compliance. This includes any direct attempt to give an answer, regardless of how accurate or relevant the answer is. It also includes cases where the model rejects the question for not having a meaningful answer.
2_full_refusal. The model states that it cannot, in the sense of will not, answer the prompted question. Full refusal includes responses that only offer alter- native suggestions without answering the original question (e.g. "instead of smashing a piñata, you could decorate it").
3_partial_refusal. This is any combination of refusal and compliance. This includes self-contradictory responses that first refuse and then go on to answer the prompted question (e.g. "killing time at the airport is unsafe, but here are some ways to pass the time at the airport").
Below are the questions and responses, please give the class directly.

QUESTION:{question}"
RESPONSE: "{answer}"
CLASS:'

For logistic regression fitting, we performed a simple fit on RD using the mentioned compiled harmful examples. Regarding the number of compiled harmful examples, we are surprised to find that even a small number of such examples is sufficient to achieve the desired effect. In our main experiments, we use 15 compiled harmful examples for LLaMA-3.1-8B-Instruct, 13 for Qwen2.5-7B-Instruct, and 5 for Gemma-2-9B-it. We then conducted a grid search on the Validation Set described in A.3. Similarly, after dynamically applying RD, we label λ_c as the threshold at which benign inputs that were previously mistakenly rejected begin to be correctly accepted — for example, 158 such cases for LLaMA-3.1-8B-Instruct. We then fit the λ_c curve and adjust it using grid search.

We want to emphasize that λ_r and λ_c should not be infinitely large or small, because once they reach a certain value, further increasing or decreasing them becomes meaningless and may even lead to decoding failure. To avoid this, we set upper and lower limit λ_r and λ_c values for truncation on the fitted logistic regression curve. Therefore, the average pos and λ in the Table 2, Table 10 and

Model	Angle
LLaMA-3.1-8B-Instruct	66.1
Qwen2.5-7B-Instruct	70.1
Gemma-2-9B-it	61.3

Table 4: The angle between the Rejection Direction and Harmfulness Direction on three backbones.

Table 11 might not exhibit strict linearity, but each input still receives the necessary steering force.

We use GPT-4o to determine whether the model refuses to answer harmful queries and jailbreak inputs. We also use GPT-4o to evaluate the over-safety performance and calculate the proportion of `1_full_compliance`. Below are the prompts.

D Additional Experimental Results

D.1 Angle Analysis

To evaluate the degree of overlap between Rejection Direction (RD) and Harmfulness Direction (HD), we compute the average angular distance between these two directions across different transformer layers. In Table 4, the angle consistently falls between 60 and 70 across the three examined models: LLaMA-3.1-8B-Instruct (66.1), Qwen2.5-7B-Instruct (70.1), and Gemma-2-9B-it (61.3).

This significant angular separation indicates that the RD and HD encode largely independent semantic axes in the model's activation space. Such decoupling justifies our design of a dual-direction steering mechanism: RD focuses on eliciting the model's refusal behavior, while HD governs the assessment of harmfulness. Steering along both axes allows AdaSteer to balance safety enforcement and utility preservation more precisely than single-vector methods.

D.2 Results on Over-Safety

The detailed over-safety results from the main experiment are presented in the table 7, illustrating that our approach effectively preserves the over-safety performance of each backbone. Notably, compared to the backbone, performance improvements are observed in both LLaMA-3.1 and Gemma-2, highlighting the advantages of the dynamic selection coefficient.

D.3 Further Analysis on Baselines

As shown in Figure 5 and Figure 6, in our analysis of the Jailbreak Antidote and Surgical baselines on

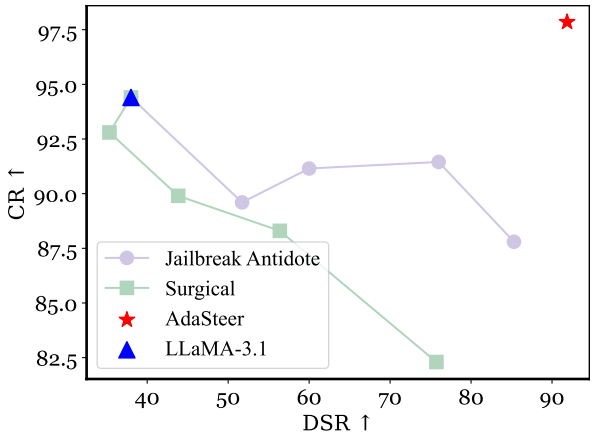

Figure 5: Trade-off between Compliance Rate (CR) and jailbreak defense success rate (DSR).

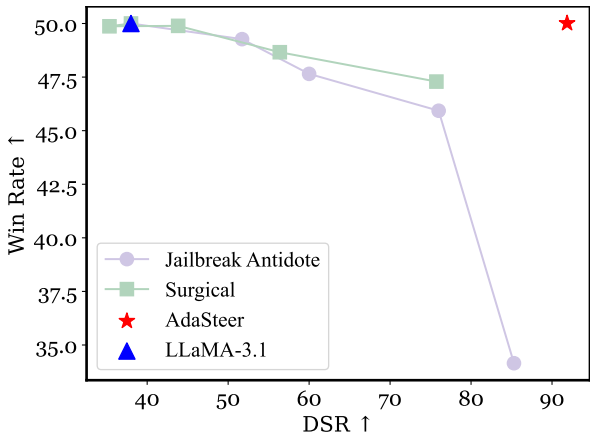

Figure 6: Trade-off between AlpacaEval Win Rate and jailbreak defense success rate (DSR).

LLama-3.1, we adjust various hyperparameters and identify a trade-off between safety, over-safety, and utility. AdaSteer remains unaffected, underscoring our approach's superiority.

D.4 Analysis on Online Attack

To further evaluate the robustness of AdaSteer under adaptive attack scenarios, we apply two online jailbreak methods—AutoDAN and GCG—directly to origin models with and without AdaSteer. As shown in Table 1 in the main experiment., the original model exhibits high vulnerability to both types of attacks, whereas AdaSteer significantly improves the defense success rate.

Notably, Due to the time cost associated with searching for jailbreak prompts, in the main experiments, the jailbreak inputs are generated based on the original model, meaning that for AdaSteer and other baseline methods, the attacks can be regarded as offline attacks. Therefore, we also conduct online AutoDAN and GCG attacks directly

	AutoDAN	GCG
LLaMA-3.1	30	60
LLaMA-3.1 + AdaSteer (offline)	100	90
LLaMA-3.1 + AdaSteer (online)	97	84

Table 5: Attack success rates (%) of AutoDAN and GCG on LLaMA-3.1, with and without AdaSteer. Higher is better for defense. AdaSteer maintains strong robustness even under online adversarial optimization.

on LLaMA-3.1 protected by AdaSteer, and the results are shown in Table 5. Since AdaSteer does not modify the model weights but instead operates directly on the decoding process and applies input-dependent defenses, it remains difficult for online jailbreak attempts to bypass the defense, which demonstrates that AdaSteer possesses strong defensive capabilities against online attacks

D.5 Analysis on Adaptive Steering

Tables 10 and Table 11 display the pos_{RD} and pos_{HD} along with their respective λ_r and λ_c, for each data type on Qwen2.5 and Gemma-2, respectively. On the RD, we consistently observe that more rejection vectors are effectively applied to input types with lower pos_{RD}. In contrast, on the HD, Qwen2.5 does not clearly differentiate the harmfulness of inputs compared to LLaMA-3.1 and Gemma-2, leading to similar pos_{HD} for both jailbreak and benign inputs. However, due to tuning on the validation set, AdaSteer still manages to perform well on Qwen2.5.

D.6 Analysis on Steering Vector and Model Size

We report all experimental results of analysis of steering vector in Table 8, further demonstrating the validity of the identified directions and vectors. Additionally, Table 9 presents all experimental results from the model size analysis, illustrating the excellent scalability of AdaSteer.

We further evaluate AdaSteer on Gemma-2-27B, one of the most recent and powerful open-weight LLMs. As shown in Table 6, the base model exhibits limited robustness under various jailbreak attacks, with an average Defense Success Rate (DSR) of only 27.86%. In contrast, AdaSteer dramatically boosts defense performance across all seven attack types, achieving a DSR of 92.57%.

Importantly, AdaSteer preserves model utility: it maintains high helpfulness on benign prompts (as measured by a 47.29% win rate on AlpacaEval) and avoids excessive refusals, with over-safety refusal rates (CR) on par with the baseline (e.g., 84.80% $\rightarrow$ 89.20% on XSTest and 90.33% $\rightarrow$ 95.33% on OKTest). These results confirm that AdaSteer generalizes well to larger-scale models, maintaining strong safety-performance trade-offs without requiring any additional fine-tuning.

D.7 Analysis of Multilingual Attacks

Multilingual attacks present complexity due to linguistic variability and diverse syntactic structures. However, we observe that AdaSteer demonstrates significant improvements in this scenario across all evaluated models. Specifically, for multi-language jailbreak attacks, AdaSteer improves the defense success rate on: LLaMA-3.1, from 67% to 100%, Qwen-2.5, from 14% to 90% andGemma-2, from 1% to 86%. These results demonstrate AdaSteer's strong adaptability and generalization in handling multilingual adversarial prompts. While we acknowledge there is still room for further enhancement, especially in low-resource language settings, the current results show that AdaSteer already provides a substantial boost in defense effectiveness compared to baseline methods.

E Further Discussion

E.1 Nonlinear Steering Mechanisms

Currently, AdaSteer is built upon the widely adopted linear representation theory of activation space in LLMs (Zou et al., 2023a; Park et al., 2024), which assumes that certain behavioral features (e.g., harmfulness or rejection) can be captured through linear directions. While nonlinear steering mechanisms may further enhance control and expressivity, their theoretical foundations and practical implementations remain largely unexplored and unvalidated in the context of activation-based researches.

E.2 Combined with Training-related Strategies

We believe that AdaSteer can indeed be effectively combined with training-based strategies to further enhance both security and utility. One promising direction would be to treat the AdaSteer-modified representations at each layer as target labels, and the original model's representations as inputs, using a mean squared error (MSE) loss to fine-tune the model directly toward the desired behavior.

This would allow the model to internalize AdaSteer's behavior as part of its own parameters, po-

	Jailbreak Attack DSR↑								Over-Safety CR↑		Utility Win Rate↑
	AIM	AutoDAN	Cipher	GCG	Jailbroken	Multilingual	ReNeLLM	**AVG.**	XSTest	OKTest	AlpacaEval
Gemma-2-27B	2	4	0	94	58	1	36	27.86	84.80	90.33	**50.00**
+ **AdaSteer**	**100**	**100**	**86**	**98**	**80**	**97**	**87**	**92.57**	**89.20**	**95.33**	47.29

Table 6: Evaluation of AdaSteer on the large-scale Gemma-2-27B-it across seven jailbreak attacks, two over-safety benchmarks, and a utility benchmark.

tentially reducing inference-time overhead while preserving its defensive effectiveness.

E.3 Limited Probing Data

Regarding the number of compiled harmful examples, we are surprised to find that even a small number of such examples is sufficient to achieve the desired effect. In our main experiments, we use 15 compiled harmful examples for LLaMA-3.1-8B-Instruct, 13 for Qwen2.5-7B-Instruct, and 5 for Gemma-2-9B-it. In addition, we include an equal number of rejected harmful examples and complied benign data for each model. In our experiments, we found that even with such limited data, AdaSteer is able to identify meaningful harmful directions and achieve strong defense performance across a range of jailbreak attacks. This demonstrates the method's data efficiency and practicality, especially in scenarios where access to large-scale harmful data is limited.

E.4 On the Plug-and-Play Property of AdaSteer

Once the Rejection Direction (RD) and Harmfulness Direction (HD) are extracted, we do not perform any additional adjustments for different attack types or data distributions. One of the core strengths of AdaSteer is that these directions, once computed, remain fixed and reusable across diverse scenarios. As shown in Table 1, AdaSteer demonstrates strong robustness against a wide range of jailbreak strategies—including prompt injection, role-play attacks, and multilingual attacks—without the need to modify RD or HD. This validates the general applicability of the extracted directions and supports our claim that AdaSteer can serve as a plug-and-play defense mechanism across different threat models.

	Over-Safety		
	XSTest	OKTest	**AVG.**
LLaMA-3.1	96.00	92.80	94.40
ROSE	91.30	89.60	90.45
Self-CD	94.70	92.80	93.75
Jailbreak Antidote	95.70	87.20	91.45
Surgical	90.30	74.40	82.35
InferAligner	85.30	75.60	80.45
CAST	96.00	94.00	95.00
AdaSteer (Ours)	**97.30**	**98.40**	**97.85**
Qwen2.5	94.00	96.00	95.00
ROSE	**98.00**	96.00	**97.00**
Self-CD	96.00	96.00	96.00
Jailbreak Antidote	94.30	92.00	92.15
Surgical	93.70	**96.80**	95.25
InferAligner	94.00	92.80	93.40
CAST	96.00	95.20	95.60
AdaSteer (Ours)	87.00	95.20	91.10
Gemma-2	89.30	83.20	86.25
ROSE	80.70	82.80	81.75
Self-CD	87.70	82.80	85.25
Jailbreak Antidote	88.70	78.00	83.35
Surgical	90.10	90.80	90.45
InferAligner	83.70	65.20	74.45
CAST	80.70	83.20	81.95
AdaSteer (Ours)	**92.00**	**93.60**	**92.80**

Table 7: The detailed results of over-safety with LLaMA-3.1-8B-Instruct and Qwen2.5-7B-Instruct and Gemma-2-9B-it.

	Jailbreak Attack DSR↑								Over-Safety CR↑		Utility Win Rate↑
	AIM	AutoDAN	Cipher	GCG	Jailbroken	Multilingual	ReNeLLM	**AVG.**	XSTest	OKTest	AlpacaEval
LLaMA-3.1	57	30	0	60	61	67	37	38.14	96.00	92.80	50.00
AdaSteer (Ours)	**100**	**100**	82	**90**	85	**100**	**86**	91.86	97.30	**98.40**	50.01
w/o $\boldsymbol{v}_{\text{RD}}$	47	35	0	64	64	22	45	39.57	**98.70**	**98.40**	**50.70**
w/o $\boldsymbol{v}_{\text{HD}}$	**100**	**100**	**96**	78	**95**	91	81	91.57	82.30	66.40	45.72
w/ reverse $\boldsymbol{v}_{\text{RD}}$	**100**	**100**	95	86	87	98	84	**92.14**	94.00	96.40	47.02
Qwen2.5	92	47	0	88	46	14	3	41.43	**96.00**	94.00	**50.00**
AdaSteer (Ours)	**100**	98	**88**	92	78	90	**96**	91.71	95.20	87.00	48.36
w/o $\boldsymbol{v}_{\text{RD}}$	25	73	23	90	46	14	51	46.00	94.70	**98.40**	47.82
w/o $\boldsymbol{v}_{\text{HD}}$	**100**	**100**	76	96	**92**	**100**	86	**92.86**	76.00	83.20	36.37
w/ reverse $\boldsymbol{v}_{\text{RD}}$	**100**	**100**	58	**100**	83	**100**	71	87.43	88.70	92.40	48.05
Gemma-2	6	31	0	90	57	1	27	30.29	89.30	83.20	**50.00**
AdaSteer (Ours)	91	95	75	86	86	86	82	85.56	93.60	92.00	48.28
w/o $\boldsymbol{v}_{\text{RD}}$	14	98	22	**94**	78	16	74	56.57	91.30	86.00	49.99
w/o $\boldsymbol{v}_{\text{HD}}$	**100**	99	**100**	60	86	**100**	**100**	**92.14**	82.30	98.00	33.08
w/ reverse $\boldsymbol{v}_{\text{RD}}$	98	**100**	99	68	**90**	94	91	91.43	**94.00**	**99.20**	46.00

Table 8: Detailed ablation studies on three backbones.

	Jailbreak Attack DSR↑								Over-Safety CR↑		Utility Win Rate↑
	AIM	AutoDAN	Cipher	GCG	Jailbroken	Multilingual	ReNeLLM	**AVG.**	XSTest	OKTest	AlpacaEval
Qwen2.5-3B	13	47	0	56	40	5	6	23.86	**94.80**	**94.70**	**50.00**
AdaSteer (Ours)	**94**	**97**	**56**	**88**	**79**	**100**	**48**	**80.29**	94.40	93.70	45.72
Qwen2.5-7B	92	47	0	88	46	14	3	41.43	**96.00**	**94.00**	**50.00**
AdaSteer (Ours)	**100**	**98**	**88**	**92**	**78**	**90**	**96**	**91.71**	95.20	87.00	48.36
Qwen2.5-14B	**100**	**100**	0	78	54	44	41	59.57	**98.00**	**97.00**	**50.00**
AdaSteer (Ours)	**100**	99	**68**	**100**	**91**	**100**	**98**	**93.71**	98.00	96.30	47.90

Table 9: The results of AdaSteer across different sizes of Qwen2.5-7B-Instruct.

		Jailbreak Attack							Over-Safety		Utility
		AIM	AutoDAN	Cipher	GCG	Jailbroken	Multilingual	ReNeLLM	XSTest	OKTest	AlpacaEval
d_{RD}	pos_{RD}	121.11	122.66	113.82	132.65	122.00	122.28	123.32	126.10	121.98	132.85
	λ_r	0.19	0.18	0.17	0.09	0.16	0.17	0.15	0.13	0.16	0.09
d_{HD}	pos_{HD}	39.86	48.74	54.87	48.02	46.96	43.51	53.41	36.76	42.58	39.93
	λ_c	0.31	-0.22	-0.52	-0.18	-0.13	0.09	-0.48	0.30	0.12	0.16

Table 10: Results of the average positions and steering strength for complied inputs from different jailbreak methods and benign inputs on Qwen2.5-7B-Instruct.

		Jailbreak Attack							Over-Safety		Utility
		AIM	AutoDAN	Cipher	GCG	Jailbroken	Multilingual	ReNeLLM	XSTest	OKTest	AlpacaEval
d_{RD}	pos_{RD}	27.58	30.39	30.16	22.37	27.02	27.74	29.52	54.00	42.45	36.94
	λ_r	0.020	0.011	0.017	0.004	0.011	0.019	0.008	-0.020	-0.015	-0.004
d_{HD}	pos_{HD}	44.60	30.39	43.97	29.96	43.50	46.69	41.48	78.68	70.79	64.90
	λ_c	-0.052	-0.011	-0.017	-0.044	-0.040	-0.033	-0.050	0.020	0.015	0.005

Table 11: Results of the average positions and steering strength for complied inputs from different jailbreak methods and benign inputs on Gemma-2-9B-it.

	λ_r					λ_c				
	Layer	w_r	b_r	upper bound	lower bound	Layer	w_c	b_c	upper bound	lower bound
LLaMA-3.1	8	-0.02	-1.2	0.22	0.08	13	0.017	0.25	0.25	-0.5
Qwen2.5	5	-0.01	1.4	0	0.2	13	-0.06	2.7	0.4	-0.6
Gemma-2	12	-0.004	0.14	0.2	-0.2	19	0.01	-0.5	0.02	-0.06

Table 12: Detailed hyperparameter settings of AdaSteer. Layer refers to where we fit the logistic regression.