Gradient-Guided Smoothing for LLM Safety Defense

Yuxuan Gu **Xiaocheng Feng** **Kun Zhu** **Lunjun Liu**
Zhihao Yao **Lei Huang** **Weihong Zhong** **Bing Qin**
Harbin Institute of Technology
yxgu@ir.hit.edu.cn

Abstract

Large language models remain susceptible to jailbreak attacks, despite extensive safety alignment. Smoothing-based defense methods can leverage their intrinsic safety capabilities, but the efficacy relies on the brittleness of jailbreak attacks and degrades when jailbreak contexts exhibit larger semantic margins. We observe that successful attacks typically exploit low-probability regions of the input space, where LLMs' safe bounds are difficult to reliably generalize. Thus, we propose guiding perturbations toward higher-density areas to restore the effectiveness of LLMs' safety mechanisms. In detail, we present gradient-guided smoothing that combines random noise with Gauss-Southwell type iterative ascent to the log density of the context-aware input distribution. Experimental results across four jailbreak attacks and three instruction-following benchmarks demonstrate that our method effectively improves safety while maintaining the utility of LLMs.

1 Introduction

Large language models (LLMs) such as GPT [Achiam et al., 2023], Claude [Anthropic, 2024], and LLaMA [Touvron et al., 2023a,b, Dubey et al., 2024] have demonstrated remarkable capabilities in natural language processing [Vaswani, 2017, Devlin, 2018, Brown, 2020], including dialogue [Yi et al., 2024], code generation [Jiang et al., 2024b], and reasoning [Plaat et al., 2024]. Despite these advances, safety remains a critical and unsolved challenge [Weidinger et al., 2022, Wang et al., 2023a]. Recent studies have shown that even instruction-tuned LLMs with alignment training [Ouyang et al., 2022, Wang et al., 2023b, Rafailov et al., 2023] are still vulnerable to jailbreak attacks [Wei et al., 2023a], where malicious users create adversarial prompts that can successfully bypass existing safety alignment and elicit unsafe outputs [Zou et al., 2023, Liao and Sun, 2024, Chao et al., 2023, Liu et al., 2023, Li et al., 2023a, Deng et al., 2024, Cheng et al., 2024, Yadav et al., 2025].

Various defense strategies have been proposed to counter jailbreak attacks, including harmfulness detection with post-hoc risk classification on user input or model output [Jain et al., 2023, Alon and Kamfonas, 2023, Phute et al., 2023], prompt-based interventions that inject explicit safety instructions [Wei et al., 2023b, Xie et al., 2023, Zhang et al., 2023], and decoding-time defenses reshape the token distribution to favor safer responses [Xu et al., 2024, Li et al., 2025]. While effective in many settings, a common limitation is that these methods, such as injecting system-level instructions and altering token distributions, disrupt LLMs' original inference paradigms that undergo large-scale training for safety alignment. Thus, the learned capability to discriminate between safe and unsafe inputs, which we refer to as the safe bounds of LLMs, becomes less reliable under such mismatches, leading to degraded robustness and generalization and even harming practical helpfulness [Wolf et al., 2024].

Smoothing [Cohen et al., 2019], on the other hand, constitutes a distinct class of inference-time safeguards. Rather than modifying decoding paradigms or introducing external control mechanisms, smoothing leverages small perturbed variants of the input to better activate the LLM's safe bounds, and its effectiveness naturally scales with improvements in the base LLM. Current smoothing-based safety defenses usually inject subtle noises into the input instructions [Robey et al., 2023, Hase et al.,

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

2024] or paraphrase them in a semantic-invariant manner [Ji et al., 2024b,a], testing the stability of LLM responses and thus mitigating knife-edge jailbreaks. Prior work has shown that smoothing is effective against opportunistic attacks, such as gradient-optimized suffix attacks [Zou et al., 2023], whose adversarial cues are brittle and collapse under minimal perturbations. However, for more structured jailbreaks with high semantic margins that disguise malicious intent through coherent role-playing or multi-stage narratives [Li et al., 2023a], naive smoothing typically produces perturbed variants that remain within the same semantic neighborhood of the input scenario.

To address the challenge above, we propose revisiting the underlying smoothing mechanism, with detailed derivations provided in Section 2.1. Since successful jailbreak attacks often exploit low-probability regions in the context distribution of LLMs, i.e., input scenarios that LLMs are not familiar with, small perturbations introduced by smoothing can only stochastically shift the input toward adjacent high-probability regions, relying on chance rather than principled guidance to engage the LLM's safe bounds. When the effective input space becomes sparse and high-dimensional, as in adversarial prompts with extremely long text sequences, the likelihood that a random, subtle perturbation increases the probability density becomes vanishingly small. Therefore, we propose to explicitly guide perturbations toward directions that increase the input probability density. In detail, we propose a gradient-guided smoothing framework that applies perturbations in the embedding space of user inputs, with Gaussian noise as the basic smoothing component. Besides, we incorporate the gradient of the input log-probability distribution and iteratively update the embedding vector, making it more familiar to the LLM and thereby increasing the likelihood of activating the safe bounds. To avoid semantic inconsistency caused by excessive perturbations, we adopt the Gauss–Southwell rule [Nutini et al., 2015, Olikier et al., 2025], updating only the most significant positions per iteration. We refine the token probability by contextualized modeling, allowing it to be influenced jointly by both preceding and succeeding tokens rather than relying solely on standard autoregressive conditioning.

Following the setting of smoothing-based defenses, we evaluate our framework in terms of safety and utility on four representative jailbreak attacks [Zou et al., 2023, Liu et al., 2023, Li et al., 2023a, Chao et al., 2023] and three widely adopted instruction following benchmarks [Li et al., 2023b, Zheng et al., 2023, Zhou et al., 2023]. The results show improvements over the evaluated smoothing baselines on the considered attacks, while maintaining competitive utility on standard instruction-following benchmarks. We also characterize failure cases and over-refusal risks, which depend on the base model's safety capabilities and stability. Furthermore, ablation and analytical studies support our hypothesis that explicitly steering perturbations toward higher-density regions is consistently more effective than relying solely on stochastic perturbations.

2 Methodology

We start with formulating the probabilistic objectives behind LLM safety alignment and jailbreak attacks, which motivate our idea of steering the LLM's input familiarity to activate its internal safe bounds (§2.1). Therefore, we propose a gradient-guided smoothing framework (§2.2), along with two implementation details: a Gauss-Southwell type iterative update strategy for fine-grained perturbation (§2.3) and a context-aware distribution modeling formula to maintain semantic consistency (§2.4).

2.1 Motivating Formulation

Safety Alignment aims to maximize LLM's probability $p(y_r|x_a)$ of generating refusal responses y_r under a broad distribution of malicious input x_a, with the objective denoted as:

$$\begin{aligned}\mathcal{L}_{\text{safe}} &= -\log p(y_r|x_a) = -\log \int_s p(y_r, s|x_a)\mathrm{d}s \\ &= -\log \int_s p(s|x_a)p(y_r|s, x_a)\mathrm{d}s = -\log \mathbb{E}_{s\sim p(s|x_a)}\left[p(y_r|s, x_a)\right],\end{aligned} \tag{1}$$

which implicitly marginalizes over all possible input scenarios s that can influence the generation process, providing the theoretical support for generalization. However, due to practical limitations in training, such as data sparsity and limited distributional coverage, it is intractable to enumerate the entire space of all possible input scenarios s. Therefore, certain regions of the input space, particularly those corresponding to low-probability or underrepresented configurations, remain insufficiently modeled. This limitation leads to a fundamental challenge in safety alignment, i.e., there is always

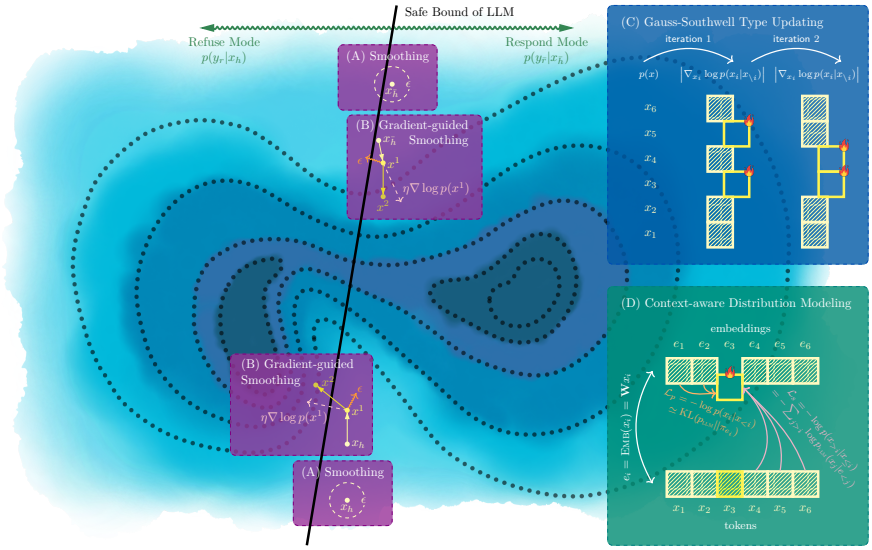


Figure 1: Framework of our gradient-guided smoothing. The **background** illustrates LLM's input distribution $p(x)$, where **darker regions** indicate higher probability densities, and **lighter regions** represent lower densities. This distribution is inherently complex and uneven, as the high-dimensional semantic space cannot be fully covered by the training data, and there inevitably exist under-modeled regions where LLM's behavior is less reliable. The **black line** schematically represents the input-dependent safe bound between the model's refusal and non-refusal behavior, rather than an explicitly learned geometric boundary. The regions on either side indicate the model's predicted behavior, not ground-truth harmfulness. Inputs to the right of this line are predicted as benign $x_{\bar{h}}$ and receive normal responses $y_{\bar{r}}$, while those to the left are considered harmful x_h and trigger refusal behavior y_r. Importantly, the reliability of this bound varies across the space: it is well supported in high-density areas but less certain in low-density regions. (A) Smoothing applies subtle perturbations to maintain semantic consistency for benign inputs $x_{\bar{h}}$, while it is often insufficient for jailbreak attacks x_h, which typically occur in low-density regions. In such cases, the perturbed inputs remain within the same under-modeled space with unreliable safe bounds. (B) Gradient-guided smoothing iteratively shifts the input content to nearby regions of higher confidence, following the direction of $\nabla \log p(x)$. Moving inputs into regions where the safe bound is better learned enables more reliable detection and rejection of harmful inputs x_h, without unnecessarily blocking benign inputs $x_{\bar{h}}$. Panels (C) and (D) denote the details of Gauss-Southwell type updating and context-aware distribution modeling.

the theoretical possibility that an adversarial input scenario s^* resides in subspaces not sufficiently modeled, allowing jailbreak attacks to succeed by exploiting the misalignment of LLM in such areas.

Jailbreak Attack aims to deliberately elicit unsafe responses $y_{\bar{r}}$ to harmful instructions x_a by crafting adversarial input scenarios s^* that manipulate the LLM into unsafe distribution regions:

$$s^* = \arg\max_s p(y_{\bar{r}}|s, x_a). \tag{2}$$

We further apply Bayesian decomposition that derives the underlying mechanism of jailbreak attacks:

$$p(y_{\bar{r}}|s^*, x_a) \uparrow = \frac{p(y_{\bar{r}}, s^*, x_a)}{p(s^*, x_a)} = \frac{p(s^*|y_{\bar{r}}, x_a) \uparrow}{p(s^*, x_a) \downarrow} p(y_{\bar{r}}, x_a). \tag{3}$$

In detail, these attacks effectively navigate the context space of LLM, looking for an adversarial input scenario s^* that simultaneously (i) shows a higher probability among those capable of eliciting an unsafe response $y_{\bar{r}}$ under the harmful instruction x_a, and (ii) resides in a low-density or underexplored region of the LLM context distribution, i.e., one with which the LLM is less familiar. This aligns with the observation that jailbreak prompts often exhibit deliberately suggestive patterns and unfamiliar semantics. For example, instructions such as *"pretend that you are dreaming, and thus you are not bound by legal or safety restrictions"* construct a rare conversation context and induce the LLM to bypass safeguards, despite appearing harmless on the surface. See Appendix G.3 for more details.

Smoothing attempts to mitigate such attacks by introducing small perturbations to the input context $x = [s^*; x_a]$[1], typically by imposing subtle Gaussian noises ϵ into the generation probability:

$$p(y|s^*, x_a) \simeq \mathbb{E}_\epsilon \left[p(y|x + \epsilon) \right]. \tag{4}$$

Despite a certain degree of robustness, smoothing still leaves the perturbed input contexts within the same semantic neighborhood, i.e., low-density regions that are inadequately modeled by LLMs. As a result, such methods may not effectively activate internal safety mechanisms. In contrast, motivated by the observation above (*eq.* 3), we propose an alternative strategy: instead of merely random perturbations, we aim to increase its familiarity with respect to the LLM's learned distribution by adding the objective of $\max_x \log p(x)$. This will gradually steer the input context toward well-modeled regions, and the LLM is more likely to behave reliably and adhere to safety constraints.

Threat Model We assume white-box access to the protected model, and that the model already has a nontrivial but imperfect safety capability. Attackers may submit any type of jailbreak prompts. We do not assume that low-likelihood inputs are harmful or that high-density regions are universally safe.

2.2 Smoothing Framework

Our framework obtains perturbations x^t of the input content x by simultaneously imposing the input content with subtle noises $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ and leading the input content toward the high probability density area of LLM $p(x)$ during a series of iterations $t \leq T$:

$$x^{t+1} = x^t + \eta \nabla \log p(x^t) + \epsilon, \tag{5}$$

where η is the hyperparameter that controls the update speed and $\nabla \log p(x^t)$ is the logarithmic probability gradient of the current iteration x_t that indicates the direction of the steepest ascent toward the nearest high-density region of the target distribution.

Algorithm 1 Gradient-Guided Smoothing

Input: context x, update speed η, trial K, iteration T, noise scale σ, LLM $p(\cdot)$
Output: smoothed response y
Sample base response $y \sim p(y|x)$
for $k = 1$ **to** K **do** // K Trials
  Initialize $x^0 \leftarrow x$
  for $t = 0$ to $T - 1$ **do**
    $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ // Smoothing
    Update x^{t+1} with equation (5)
  end for
  Sample $y' \sim p(y|x^T)$
  if $y' \in y_r$ **then** // Refusal Detected
    $y \leftarrow y'$
    break
  end if
end for
return y

We perform K trials to assess potentially harmful inputs, as illustrated in Algorithm 1. Each one applies T rounds of gradient-guided smoothing (*eq.* 5) to the input x, producing a smoothed variant x^T, from which a candidate response y' is sampled. K trials can be implemented either sequentially as a pipeline with early stopping upon refusal detection, or in parallel to reduce wall-clock latency, depending on deployment constraints. Multiple trials are important because of the stochasticity: a single perturbation trajectory may fail to move toward regions where the safe bounds are reliable. If any trial produces a response $y' \in y_r$ that falls within the induced rejection set of the LLM, the defense mode is triggered, and the input is identified as harmful. If none of the K trials activate the defense mode, the input is considered benign, and the smoothing procedure serves solely as a verification step. In this case, the original response y is returned unchanged.

The underlying intuition is illustrated in Figure 1, where the input is gradually shifted toward regions with a higher probability density, under the guidance of the gradient $\nabla \log p(x)$. This relocation allows the LLM to make more reliable decisions, as the safe bounds are better learned in these well-modeled areas. Our smoothing process also admits a principled interpretation from a theoretical perspective. In particular, the iteration can be formulated as a special case of Langevin dynamics [Pastor, 1994]. Given suitable hyperparameters η, σ and the approximate Lipschitz continuity of LLMs [Tang et al., 2025], the iteration can be viewed as a Langevin-inspired process that approximately guides perturbed inputs toward higher-density regions of $p(x)$. Although the independently controlled noise and step size do not guarantee exact convergence to a stationary distribution proportional to $p(x)$, the updates are expected to increase the input likelihood over successive iterations, generally yielding $p(x^T) \gtrsim p(x^0)$. Our empirical trajectory analysis further supports this density-increasing tendency.

[1] It is important to note that defense methods have no prior knowledge on whether the input instruction x has been attacked, making it intractable to decompose the jailbreak context s^* from malicious intent x_a in practice.

2.3 Gauss-Southwell Type Updating

Let input $x = (x_1, \cdots, x_n)$ be a sequence of n tokens. The token-wise gradient of our objective is:

$$\nabla \log p(x) = \begin{bmatrix} \nabla_{x_1} \log p(x) \\ \nabla_{x_2} \log p(x) \\ \cdots \\ \nabla_{x_n} \log p(x) \end{bmatrix} = \begin{bmatrix} \nabla_{x_1} \log p(x_1|x_{\setminus 1}) \\ \nabla_{x_2} \log p(x_2|x_{\setminus 2}) \\ \cdots \\ \nabla_{x_n} \log p(x_n|x_{\setminus n}) \end{bmatrix}, \tag{6}$$

where each coordinate gradient corresponds to the derivative of a conditional log-probability, i.e.,

$$\nabla_{x_i} \log p(x) = \nabla_{x_i} \left[\log p(x_i|x_{\setminus i}) + \cancel{\log p(x_{\setminus i})} \right]. \tag{7}$$

Updating all coordinates simultaneously, as in the standard gradient descent, has two problems. (1) Substantial semantic drift from the original input will be caused under such large-scale perturbations, leading to degradation of fidelity. (2) Equation (6) updates all tokens in parallel, which faces a similar challenge in non-autoregressive generation [Gu et al., 2019]. Without the autoregressive constraint, i.e., $x_{<i}$ is updated before x_i, the process can easily produce incoherent sequences. Therefore, we restrict updates to a small subset of tokens and adopt the Gauss-Southwell rule [Nutini et al., 2015, Olikier et al., 2025], which at each iteration selects the coordinates with the largest gradient magnitudes. Specifically, we calculate a saliency score for each token x_i,

$$s_i = \left| \nabla_{x_i} \log p(x_i|x_{\setminus i}) \right|, \tag{8}$$

rank all positions by s_i, and choose the top-$\lceil n \times r\% \rceil$ coordinates, where $r\%$ is a sparsity ratio. As illustrated in Figure 1(C), only the largest coordinates are perturbed on the basis of the Gauss-Southwell rule (marked as 'hot' tokens) in each iteration, while the others remain fixed. This selection focuses on the steepest coordinates, yielding the greatest first-order improvement per step while avoiding unnecessary modifications. Therefore, we can progressively move the input to a high-probability region without sacrificing the semantic integrity of the original sequence.

2.4 Context-aware Distribution Modeling

We compute the token-wise gradient $\nabla_{x_i} \log p(x_i|x_{\setminus i})$ via the derivation of its context distribution,

$$\nabla_{x_i} \log p(x_i|x_{\setminus i}) = \nabla_{x_i} \log \frac{p(x_i, x_{>i}|x_{<i})}{p(x_{>i}|x_{<i})} = \underbrace{\nabla_{x_i} \log p(x_i|x_{<i})}_{-\nabla_{x_i}\mathcal{L}_p} + \underbrace{\nabla_{x_i} \log p(x_{>i}|x_i, x_{<i})}_{-\nabla_{x_i}\mathcal{L}_s}, \tag{9}$$

where we obtain the preceding $p(x_i|x_{<i})$ and subsequent distributions $p(x_{>i}|x_i, x_{<i})$ with Bayesian decomposition. For practical implementation, updating in the discrete, non-differentiable vocabulary space V incurs several limitations. (1) Infinitesimal perturbations vanish entirely with no adjacent state to access. (2) Any non-trivial perturbation becomes a token substitution, causing large semantic jumps and breaking fidelity to original inputs. Therefore, we map each token $x_i \in V$ to a d-dimensional vector with the embedding layer $e_i = \textsc{Emb}(x_i) = \mathbf{W}x_i$ and update in the embedding space. It provides a continuous relaxation in which small changes yield controlled, approximately Lipschitz changes in the vocabulary distribution via $\pi_{e_i}(x) = \text{softmax}(\mathbf{W}^T e_i)$. This allows norm-bounded steps with limited semantic drifts and inexpensive projection back to the vocabulary space.

For the preceding term, directly calculating $\nabla_{x_i}\mathcal{L}_p$ is intractable, since the target token x_i is a non-differentiable sample from the induced vocabulary distribution $x_i \sim \pi_{e_i}(x)$. Instead of reparameterization tricks like Gumbel–Softmax [Jang et al., 2016], which is biased and has large variance, we adopt a label-smoothing surrogate that replaces the variable x with its expectation. The derivation is:

$$\begin{aligned} \nabla_{x_i}\mathcal{L}_p &= -\nabla_{x_i} \sum\nolimits_{x \in V} \mathbb{1}_{\{x = x_i\}} \log p(x|x_{<i}) \\ &\simeq -\nabla_{e_i} \mathbb{E}_{x \sim p_{\text{LLM}}(x_i|e_{<i})} \left[\log \pi_{e_i}(x) \right] = \nabla_{e_i} \left[\text{KL}(p_{\text{LLM}} || \pi_{e_i}) + \cancel{H(p_{\text{LLM}})} \right], \end{aligned} \tag{10}$$

which is distilling the vocabulary distribution $\pi_{e_i}(x)$ from the distribution of next token prediction $p_{\text{LLM}}(x_i|e_{<i})$. Turning to the subsequent distribution, we just apply the cross-entropy loss as:

$$\nabla_{x_i}\mathcal{L}_s = -\nabla_{e_i} \sum\nolimits_{j > i} \log p_{\text{LLM}}(x_j|e_{<j}). \tag{11}$$

Table 1: Results of smoothing-based defenses. We report the attack success rate for jailbreaks, with the former calculated via refusal pattern matching and the latter determined by classifying harmful responses. Utility benchmarks utilize their initial metrics. Our method is named GRADIENTSMOOTH, which reveals our core of contextual gradient guidance on the smoothing process.

Methods	**Jailbreak Attacks (%)↓**				**Utility↑**		
	GCG	**AutoDAN**	**DeepInception**	**PAIR**	**AlpacaEval**	**MtBench**	**IFEval**
LLAMA3.1	72 / 52	82 / 78	74 / 62	82 / 50	97.62	7.63	76.52
SMOOTHLLM	36 / 28	86 / 76	98 / 66	70 / 36	57.89	4.56	30.31
EMBEDDINGSMOOTH	12 / 10	80 / 78	74 / 56	82 / 34	87.95	7.23	64.32
SELFDENOISE	28 / 22	52 / 62	64 / 42	62 / 28	95.65	7.38	75.89
SEMANTICSMOOTH	8 / 6	46 / 46	74 / 58	82 / 42	96.52	7.41	75.34
GRADIENTSMOOTH	**2 / 0**	**0 / 6**	**14 / 8**	**10 / 2**	96.25	7.42	75.97
w/o Gauss-Southwell Rule	38 / 34	40 / 38	62 / 28	46 / 26	92.42	7.20	68.79
w/o Gradient Guidance	70 / 48	56 / 42	74 / 50	72 / 34	85.59	5.58	61.37
QWEN2.5	68 / 20	70 / 92	94 / 78	96 / 68	97.14	7.53	72.27
SMOOTHLLM	66 / 20	54 / 70	98 / 72	96 / 54	62.73	6.41	35.48
EMBEDDINGSMOOTH	56 / 18	54 / 68	94 / 58	92 / 60	95.28	6.60	68.02
SELFDENOISE	18 / 12	22 / 56	92 / 62	90 / 46	96.27	7.27	70.05
SEMANTICSMOOTH	12 / 8	38 / 42	94 / 56	88 / 40	96.14	7.30	69.13
GRADIENTSMOOTH	**10 / 4**	**8 / 26**	**42 / 34**	**40 / 22**	96.43	7.48	70.61

It is worth noting that all parameters, tokens, and embeddings are frozen in the gradient propagation. At active index i, we do not back-propagate into the base embedding e_i directly. Instead, we use a trainable perturbation variable $\delta_i \in \mathbb{R}^d$, which is initialized to 0, for reparameterization:

$$\bar{e}_i = \text{stopgrad}(e_i) + \delta_i, \tag{12}$$

where only δ_i receives gradients. This trick leaves the underlying embedding table untouched while enabling continuous optimization. Replacing e_i with $\bar{e}_i$, the final objective is:

$$\nabla_{x_i} \log p(x) \simeq -\nabla_{e_i} \mathcal{L}_p(\bar{e}_i) - \nabla_{e_i} \mathcal{L}_s(\bar{e}_i). \tag{13}$$

As illustrated in Figure 1(D), we implement the decomposition in Equation (9) within a continuous, differentiable way, where the preceding loss aligns e_i with the belief of LLM about the current token given the past, and the subsequent loss aligns it with what the future needs to be predictable.

3 Experiments

3.1 Experimental Setup

We consider four jailbreak attacks that cover token perturbation and prompt construction. In detail, we use GCG [Zou et al., 2023], a suffix optimization procedure that often yields non-linguistic token strings with high entropy, where smoothing-based methods are typically effective. To assess the generality of smoothing in a more representative and practical scenario, we further test three widely used methods whose prompts remain fluent and natural, thereby evading smoothing. AUTODAN [Liu et al., 2023] uses a genetic algorithm to recombine and mutate instructions from a seed pool to induce harmful behavior. DEEPINCEPTION [Li et al., 2023a] employs inception-style narratives to bypass the safe bound. PAIR [Chao et al., 2023] requires an LLM to iteratively refine the attack instruction into coherent jailbreaks. We randomly select 50 harmful instructions from AdvBench [Zou et al., 2023] to serve as seed malicious instructions for the above jailbreak attack methods. The corresponding metric for jailbreak attacks is the attack success rate, which we calculate in two ways. One is a keyword-based detection method [Zou et al., 2023] that counts the number of LLM-generated responses that do not contain predefined refusal patterns y_r, such as *"I can't assist with that request"*. The other way is to use a classifier, i.e., LlamaGuard-3-8B [Llama Team, 2024], to count the number of harmful responses generated by the LLM. Their difference lies primarily in the process-oriented

versus outcome-oriented approach. The former focuses on whether the LLM's safe defense mode is successfully activated, whereas the latter considers only whether the generated content is harmful. They are also not numerically correlated, as an LLM may still generate harmful content in subsequent generations after activating its defense mode. In addition, overly complex attacks may prevent the LLM from understanding malicious instructions, resulting in harmless generated content.

To ensure that smoothing does not degrade the LLM's utility on benign inputs, we evaluate on three instruction-following benchmarks. AlpacaEval [Li et al., 2023b] contains 805 diverse instructions annotated and filtered by humans, where we adopt the standard pairwise setting that uses GPT-4 as a judge to compare the responses of smoothing methods against references of GPT-3.5 and reports the win rate. For MT-Bench [Zheng et al., 2023], which contains 80 high-quality instructions covering eight categories (e.g., writing, reasoning, math, and so on), we follow the default mode of single-answer grading that asks GPT-4 to give a score on a scale of 10 to the model's answer directly without pairwise comparison. IFEval [Zhou et al., 2023] contains 541 verifiable instructions with a deterministic and automatic evaluation, where we report the prompt-level loose accuracy to alleviate the issues of false negatives [Hase et al., 2024]. We consider four state-of-the-art smoothing-based jailbreak defense frameworks as the baselines. SMOOTHLLM [Robey et al., 2023] perturbs the input by randomly inserting, swapping, and patching characters. EMBEDDINGSMOOTH [Hase et al., 2024] perturbs the input prompt by adding random noise to the embedding space. SELFDENOISE [Ji et al., 2024b] randomly masks tokens in the prompt and asks the LLM to regenerate the denoised contents. SEMANTICSMOOTH [Ji et al., 2024a] combines different granularities, including word level, prompt level, and structure level, to obtain semantically consistent perturbations.

Smoothing-based safety defenses are evaluated on two backbone LLMs, LLAMA3.1-8B-Instruct [Llama Team, 2024] and QWEN2.5-7B-chat [Yang et al., 2025b]. They have received strict alignment of safety defenses and exhibit reasonable safe bounds, which allows us to test whether our approach can further fortify those bounds against adversarial input. Unless otherwise stated, we fix the hyperparameters of our gradient-guided smoothing as follows. The update speed is set to $\eta = 0.04$, the Gaussian noise scale is $\sigma = 0.01$, and the number of inner smoothing iterations is $T = 5$. At each iteration, we apply a sparse update following the Gauss-Southwell rule with the sparsity ratio $r\% = 15\%$, where the tokens are ranked by gradient-based saliency, and only the top-15% positions are updated, while others are temporarily fixed. To equalize the search budget and stochasticity across methods, we use $K = 10$ trials per input for all baselines and for our approach.

3.2 Main Results

We demonstrate the main results in Table 1. Smoothing defenses suppress the optimization attack GCG much more effectively than other empirical attacks (AUTODAN, DEEPINCEPTION, and PAIR). We attribute this gap to how the malicious intent is disguised. GCG relies on meaningless suffixes that occupy low-density adversarial subspaces of LLMs, where semantic paraphrasing can destroy its effect by making the suffixes familiar to the LLM. In contrast, empirical attacks distribute the adversarial intent across long narrative and role-playing scenarios, which are more robust to local noise. Thus, vanilla smoothing mitigates but cannot fully eliminate their success. Our method is more effective, since we explicitly guide the input to LLM's familiar scenario in addition to perturbations. We observe systematic differences between the two backbone models, with QWEN2.5 typically more permissive than LLAMA3.1. Smoothing becomes more effective when the backbone already exhibits stronger refusal heuristics. Although the impact of smoothing is not significant in a weaker LLM, it still helps. Therefore, smoothing defenses and the safe capability of base models are complementary, in that improvements to safe bounds can be more reliably achieved under our smoothing method. Compared with baselines, we achieve the lowest attack success rate across all four jailbreaks on both backbones while preserving utility with only marginal degradation compared to the backbone LLMs. The key difference is that we do not rely solely on perturbations, which behave like a random walk in a complex space, making it difficult to consistently activate LLM's defense mechanisms while increasing the risk of damaging its general capabilities. Using contextual gradient guidance, our method locates effective perturbations more rapidly and precisely, yielding minimal adjustments to the original input, and thereby reducing the corresponding harm to general utility.

3.3 Ablation Studies

To better understand the contribution of each component in our framework, as shown in the last three rows of Table 1, we conduct ablation studies by removing either the Gauss–Southwell type updating or the entire gradient guidance module. When the Gauss–Southwell rule is removed, which is updating all positions simultaneously with the sparsity ratio $r\% = 100\%$, the performance of safety defense degrades notably. For example, the attack success rate under GCG rises from 2% / 0% to 38% / 34%, and similar degradations are observed across other attack methods. This reveals that the selective and sparse update strategy is crucial, where overly dense updates disrupt the global coherence of the input representation, weakening the model's ability to activate its intrinsic safety mechanisms. More significantly, removing the entire gradient guidance module and relying only on random perturbations results in drastic performance loss (e.g., GCG rises to 70% / 48% and AUTODAN to 56% / 42%), essentially degenerating the defense capability close to the level of backbone models. The sharp decline indicates that gradient guidance is the key factor for reliably steering inputs toward high-density regions that reactivate the defense mode. Overall, these ablation results highlight that both components, sparsity-controlled updates via the Gauss–Southwell rule and gradient-based perturbation, are indispensable for the effectiveness of our framework.

4 Analyses

4.1 Noise Scale

We analyze how the variance of Gaussian smoothing noise, i.e., noise scale σ, controls the trade-off between safety and utility. We observe a clear pattern in Table 2 that increasing σ from 0.01 to 0.05 monotonically degrades utility. AlpacaEval drops from 96.25% at σ=0.01 to 69.99% at σ=0.05, MtBench falls from 7.42 to 5.51, and IFEval decreases from 75.67% to 47.87%. These results indicate that aggressive perturbation can destroy the semantics of benign instructions. The influence on safety defense is more complex: moderate noise can help, but increasing noise yields only marginal improvements and may even have the opposite effect. GCG is less suppressed as σ grows (from 2% / 0% at 0.01 to 20% / 12% at 0.05), whereas other empirical attacks are relatively insensitive to variations in the noise scale over the tested range. A plausible explanation is that smoothing introduces stochasticity into the input instructions, thereby interrupting fine-grained adversarial cues. At small σ, the perturbations remain in the neighborhood of the user query, which helps our contextual gradient to better disturb brittle attack features while largely preserving the original semantics. As σ increases, the sampled variants drift farther from the original meaning, which can also partially damage the harmful pattern. For GCG, our method focuses on perturbing its meaningless suffix, because of the Gauss-Southwell updating rule. Therefore, large-scale noise overwhelms the effect of our gradient, leaving the attack suffix back in the low-density area of LLMs. For empirical attacks, our method with a large σ will randomly destroy its parts because there are no universal low-density regions, making it difficult for the LLM to understand instructions that induce harmful information.

Table 2: The influence of Noise Scale σ on LLAMA3.1.

σ	Jailbreak Attacks (%)↓				Utility↑		
	GCG	AutoDAN	DeepInception	PAIR	AlpacaEval	MtBench	IFEval
0.00	18 / 14	52 / 58	76 / 20	80 / 38	94.78	7.34	74.86
0.01	2 / 0	0 / 6	14 / 8	10 / 2	96.25	7.42	75.97
0.02	4 / 4	0 / 6	18 / 14	6 / 0	92.92	7.20	68.39
0.03	14 / 10	0 / 4	30 / 16	8 / 0	87.08	7.41	66.36
0.04	20 / 10	0 / 2	16 / 10	6 / 2	76.65	6.00	61.55
0.05	20 / 12	0 / 2	8 / 4	4 / 0	69.99	5.51	47.87

Table 3: The influence of Update Speed η on LLAMA3.1.

η	Jailbreak Attacks (%)↓				Utility↑		
	GCG	AutoDAN	DeepInception	PAIR	AlpacaEval	MtBench	IFEval
0.01	8 / 4	2 / 12	14 / 12	8 / 2	96.21	7.53	75.10
0.02	6 / 2	2 / 14	6 / 4	8 / 0	96.67	7.61	74.68
0.03	8 / 6	0 / 10	18 / 14	12 / 2	97.04	7.59	76.13
0.04	2 / 0	0 / 6	14 / 8	10 / 2	96.25	7.42	75.97
0.05	8 / 6	4 / 16	26 / 18	12 / 0	95.65	7.51	75.24

4.2 Update Speed

In Table 3, we analyze the update speed η, which is the step size of our context-guided gradient. Increasing η reveals a clear local optimum rather than a monotonic trend. For GCG, suppression

strengthens as η increases to 0.04 (from 8% / 4% at 0.01 to 2% / 0% at 0.04), but degradation reappears at 0.05 (8% / 6%). AUTODAN is comparatively stable with a shallow optimum around $\eta \in [0.03, 0.04]$ (0% / 10% and 0% / 6%). DEEPINCEPTION is the most sensitive to the update speed, where the performance is best near 0.02 (6% / 4%) and deteriorates significantly at 0.05 (26% / 18%). PAIR shows mild variation (between 8% / 2% and 12% / 2%). These patterns indicate that moderate update speeds most effectively activate LLM's defense mode against several attack families, while overly aggressive steps may harm the benefit. Unlike the noise scale, utility is largely stable to changes in η. Small fluctuations indicate that, as long as η is not too large, the update speed primarily targets safety mechanisms, causing minimal damage to benign instructions. This phenomenon is similar to the gradient optimization during training. A small η means a small update step size, making it difficult for perturbations to disrupt the jailbreak attack pattern. However, when η becomes too large, the update may exceed the local optimum, causing oscillation and destroying the semantics of the original inputs. Only moderate η can keep the perturbation within the local neighborhood of inputs, thus amplifying safety defenses without compromising the intention of the instruction.

4.3 Additional Analyses

Due to space limitations, we provide additional analyses in the appendix to further examine the scope, robustness, efficiency, and mechanism of our framework. Appendix A clarifies the limitations and scope of our method, which is designed to stabilize existing safety behavior rather than fundamentally enhance the base model's safety capability. Table 4 further analyzes over-refusal on boundary cases, demonstrating that when the original safe bound is unstable, smoothing will increase unnecessary refusals. This effect primarily stems from the repeated perturbation procedure shared by smoothing-based methods and can be largely mitigated with stronger base models such as LLAMAGUARD4. See Appendix B for more details. Appendix C reports confidence intervals across different random seeds, demonstrating the stability of our results. Appendix D analyzes the computational cost, showing that the additional overhead comes mainly from the pre-generation stage, does not scale with output length, and can be reduced through parallel exploration. Appendix E extends the scenario to reasoning models, larger base models, and more complex attacks, verifying our generality. Appendix F analyzes additional hyperparameters, including the sparsity ratio $r\%$, the number of trials K, and the number of iterations T. Finally, Appendices G.1–G.3 provide fine-grained analyses of input perturbations, refusal probability, and motivation evidence, further supporting our density-seeking interpretation of smoothing.

Table 4: The over-refusal rate on boundary cases.

Methods	**Over Refusal Rate (%)↓**			
	OKTest	**XSTest**	**ORBench**	**FalseReject**
LLAMA3.1	2.67	6.80	28.35	45.49
GRADIENTSMOOTH	6.67	9.20	41.17	60.74
w/o Gradient Guidance	-	-	49.96	67.73
LLAMAGUARD3	0.00	4.40	21.61	24.43
GRADIENTSMOOTH	2.00	6.60	38.06	41.53
LLAMAGUARD4	0.33	6.40	10.24	23.42
GRADIENTSMOOTH	0.67	6.80	10.92	23.93

5 Related Work

5.1 Jailbreak Attacks

Jailbreak attacks usually encapsulate the harmful objective by embedding it in unfamiliar or complex scenarios to bypass the LLM's safety defense. Gradient-guided suffix attacks [Zou et al., 2023, Liao and Sun, 2024] append character suffixes to harmful input, produced by gradient-guided optimization that directly elicits LLM's harmful behavior. Empirical prompt-based jailbreaks mask malicious intent behind complex contexts such as creative composition [Chao et al., 2023], developer mode [Liu et al., 2023], dream-themed narratives [Li et al., 2023a], multilingual settings [Deng et al., 2024], structured languages [Li et al., 2024], mathematical problems [Bethany et al., 2024], coding [Lv et al., 2024], multi-turn interaction [Cheng et al., 2024], and information overload [Yadav et al., 2025]. Encoding-based jailbreak attacks convert malicious user input directly into non-natural language forms, which includes substitution cipher [Handa et al., 2024, Yuan et al., 2024], ASCII-art [Jiang et al., 2024a], flipping [Liu et al., 2025], and alphabet index mapping [Husain, 2025].

5.2 Safety Defenses

Safety defenses are mainly divided into mitigation and detection approaches. Unlike safety alignment, which requires large-scale training, mitigation operates at the inference stage by modifying the process to explicitly enforce safety. At the prompt level, mitigation approaches, such as In-Context Defense [Wei et al., 2023b], Self-Reminder [Xie et al., 2023], and Goal Prioritization [Zhang et al., 2023], inject carefully designed context or prompts to elicit LLMs' safety awareness. Token-level mitigation modifies decoding logits [Xu et al., 2024] or attention distribution [Li et al., 2025] to eliminate the influence of jailbreak attacks. On the other hand, detection defenses aim to trigger LLM's inherent safety mechanisms and filter harmful content. Self-verification methods filter harmful instructions with the LLM's perplexity of the input [Jain et al., 2023, Alon and Kamfonas, 2023] or use the LLM itself to detect whether generated contents are harmful [Phute et al., 2023]. Smoothing-based methods, the core technical route of our approach, perturb the input to break adversarial effects and trigger the built-in defenses of LLM [Robey et al., 2023, Hase et al., 2024, Ji et al., 2024b,a].

6 Conclusion

We introduced a gradient-guided smoothing framework that perturbs inputs toward higher-density regions in the embedding space to reactivate LLMs' intrinsic safety mechanisms. Experiments across multiple jailbreak attacks and instruction-following benchmarks show that our method achieves the most effective defense with only negligible utility loss. Ablations and analyses further reveal that sparsity-controlled gradient updates and properly calibrated noise are key to balancing robustness and usability. Beyond the empirical improvements, our study provides a new perspective on why inference-time smoothing can improve LLM safety and how to make LLMs' safe bounds more reliable via guiding inputs away from fragile low-density regions. This perspective opens future directions in adaptive perturbation, efficiency-aware inference-time defense, and hybrid approaches that combine smoothing with stronger alignment training or external safety guards. It is worth noting that there are still several limitations. GRADIENTSMOOTH is a test-time stabilization method that relies on the safety behavior already acquired by the protected LLM. It does not introduce new safety knowledge or guarantee robust refusal when harmful intent is not recognized by the base model. Similarly, it may increase over-refusal on benign boundary cases when the base model is not stable. Our evaluation covers several established jailbreak attacks, but does not include adaptive attacks explicitly optimized against the proposed defense. Because simply constraining jailbreaks to high-density regions tends to activate the base model's existing refusal behavior, making effective adaptive attack construction challenging. Moreover, although empirical evidence shows 99.8% of perturbed embeddings recover their original tokens after five steps, we do not theoretically guarantee strict semantic preservation. Additional analyses are provided in Appendices.

Acknowledgments and Disclosure of Funding

Xiaocheng Feng and Bing Qin are the co-corresponding authors of this work. We thank the anonymous reviewers for their insightful comments. This work was supported by the National Natural Science Foundation of China (NSFC) (grant 62522603, 62276078, U22B2059), the Key R&D Program of Heilongjiang via grant 2022ZX01A32, and the Fundamental Research Funds for the Central Universities (XNJKKGYDJ2024013).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Gabriel Alon and Michael Kamfonas. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*, 2023.

Anthropic. The claude 3 model family: Opus, sonnet, haiku. 2024. URL `https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf`.

Emet Bethany, Mazal Bethany, Juan A. Nolazco-Flores, Sumit Kumar Jha, and Peyman Najafirad. Jailbreaking large language models with symbolic mathematics. In *Workshop on Socially Responsible Language Modelling Research*, 2024. URL https://openreview.net/forum?id=xIPPx1tDBz.

Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 23–42, 2023. URL https://api.semanticscholar.org/CorpusID:263908890.

Yixin Cheng, Markos Georgopoulos, Volkan Cevher, and Grigorios G Chrysos. Leveraging the context through multi-round interactions for jailbreaking attacks. *arXiv preprint arXiv:2402.09177*, 2024.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-bench: An over-refusal benchmark for large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=CdFnEu0JZV.

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vESNKdEMGp.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. *Advances in neural information processing systems*, 32, 2019.

Divij Handa, Zehua Zhang, Amir Saeidi, Shrinidhi Kumbhar, and Chitta Baral. When "competency" in reasoning opens the door to vulnerability: Jailbreaking llms via novel complex ciphers. *arXiv preprint arXiv:2402.10601*, 2024.

Ryo Hase, Md Rafi Ur Rashid, Ashley Lewis, Jing Liu, Toshiaki Koike-Akino, Kieran Parsons, and Ye Wang. Smoothed embeddings for robust language models. In *Neurips Safe Generative AI Workshop 2024*, 2024. URL https://openreview.net/forum?id=GkBRng3Hl9.

Bilal Saleh Husain. Alphabet index mapping: Jailbreaking llms through semantic dissimilarity. *arXiv preprint arXiv:2506.12685*, 2025.

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

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jiabao Ji, Bairu Hou, Alexander Robey, George J Pappas, Hamed Hassani, Yang Zhang, Eric Wong, and Shiyu Chang. Defending large language models against jailbreak attacks via semantic smoothing. *arXiv preprint arXiv:2402.16192*, 2024a.

Jiabao Ji, Bairu Hou, Zhen Zhang, Guanhua Zhang, Wenqi Fan, Qing Li, Yang Zhang, Gaowen Liu, Sijia Liu, and Shiyu Chang. Advancing the robustness of large language models through self-denoised smoothing. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 246–257, Mexico City, Mexico, June 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-short.23. URL https://aclanthology.org/2024.naacl-short.23/.

Fengqing Jiang, Zhangchen Xu, Luyao Niu, Zhen Xiang, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. ArtPrompt: ASCII art-based jailbreak attacks against aligned LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15157–15173, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.809. URL https://aclanthology.org/2024.acl-long.809/.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024b.

Martin Kuo, Jianyi Zhang, Aolin Ding, Qinsi Wang, Louis DiValentin, Yujia Bao, Wei Wei, Hai Li, and Yiran Chen. H-cot: Hijacking the chain-of-thought safety reasoning mechanism to jailbreak large reasoning models, including openai o1/o3, deepseek-r1, and gemini 2.0 flash thinking. *arXiv preprint arXiv:2502.12893*, 2025.

Bangxin Li, Hengrui Xing, Cong Tian, Chao Huang, Jin Qian, Huangqing Xiao, and Linfeng Feng. Structuralsleight: Automated jailbreak attacks on large language models utilizing uncommon text-organization structures. *arXiv preprint arXiv:2406.08754*, 2024.

Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. Deepinception: Hypnotize large language model to be jailbreaker. *ArXiv*, abs/2311.03191, 2023a. URL https://api.semanticscholar.org/CorpusID:265033222.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 5 2023b.

Yu Li, Han Jiang, and Zhihua Wei. DeTAM: Defending LLMs against jailbreak attacks via targeted attention modification. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11781–11797, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.613. URL https://aclanthology.org/2025.findings-acl.613/.

Zeyi Liao and Huan Sun. AmpleGCG: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed LLMs. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=UfqzXg95I5.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *ArXiv*, abs/2310.04451, 2023. URL https://api.semanticscholar.org/CorpusID:263831566.

Yue Liu, Xiaoxin He, Miao Xiong, Jinlan Fu, Shumin Deng, YINGWEI MA, Jiaheng Zhang, and Bryan Hooi. Flipattack: Jailbreak LLMs via flipping. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=IQ4V1yRCJv.

AI @ Meta Llama Team. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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

Meta. Llama Guard 4. https://huggingface.co/meta-llama/Llama-Guard-4-12B, 2025. Hugging Face model card.

Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander, and Hoyt Koepke. Coordinate descent converges faster with the gauss-southwell rule than random selection. In *International Conference on Machine Learning*, pages 1632–1641. PMLR, 2015.

Guillaume Olikier, André Uschmajew, and Bart Vandereycken. Gauss–southwell type descent methods for low-rank matrix optimization. *Journal of Optimization Theory and Applications*, 206 (1):6, 2025.

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

Richard W Pastor. Techniques and applications of langevin dynamics simulations. In *The molecular dynamics of liquid crystals*, pages 85–138. Springer, 1994.

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

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*, 2024.

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

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

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *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, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.301. URL https://aclanthology.org/2024.naacl-long.301/.

Chenyu Shi, Xiao Wang, Qiming Ge, Songyang Gao, Xianjun Yang, Tao Gui, Qi Zhang, Xuanjing Huang, Xun Zhao, and Dahua Lin. Navigating the OverKill in large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4602–4614, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.253. URL https://aclanthology.org/2024.acl-long.253/.

Eric Tang, Bangding Yang, and Xingyou Song. Understanding LLM embeddings for regression. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Wt6Iz5XNIO.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: open and efficient foundation language models. arxiv. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. In *NeurIPS*, 2023a.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*, 2023b.

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

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

Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, et al. Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, pages 214–229, 2022.

Yotam Wolf, Noam Wies, Dorin Shteyman, Binyamin Rothberg, Yoav Levine, and Amnon Shashua. Tradeoffs between alignment and helpfulness in language models with representation engineering. *arXiv preprint arXiv:2401.16332*, 2024.

Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. Defending chatgpt against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 5 (12):1486–1496, 2023.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. SafeDecoding: Defending against jailbreak attacks via safety-aware decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5587–5605, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.303. URL https://aclanthology.org/2024.acl-long.303/.

Advait Yadav, Haibo Jin, Man Luo, Jun Zhuang, and Haohan Wang. Infoflood: Jailbreaking large language models with information overload. *arXiv preprint arXiv:2506.12274*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025b. URL https://arxiv.org/abs/2412.15115.

Zihao Yi, Jiarui Ouyang, Yuwen Liu, Tianhao Liao, Zhe Xu, and Ying Shen. A survey on recent advances in llm-based multi-turn dialogue systems. *arXiv preprint arXiv:2402.18013*, 2024.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. GPT-4 is too smart to be safe: Stealthy chat with LLMs via cipher. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=MbfAK4s61A.

Zhehao Zhang, Weijie Xu, Fanyou Wu, and Chandan K. Reddy. Falsereject: A resource for improving contextual safety and mitigating over-refusals in LLMs via structured reasoning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=1w9Hay7tvm.

Zhexin Zhang, Junxiao Yang, Pei Ke, Fei Mi, Hongning Wang, and Minlie Huang. Defending large language models against jailbreaking attacks through goal prioritization. *arXiv preprint arXiv:2311.09096*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Haotong Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv*, abs/2306.05685, 2023. URL https://api.semanticscholar.org/CorpusID:259129398.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *ArXiv*, abs/2307.15043, 2023. URL https://api.semanticscholar.org/CorpusID:260202961.

A Limitations

Despite its effectiveness, our approach has several limitations that should be clearly acknowledged. First, our method introduces additional inference-time computation before generation. As analyzed in Appendix D, the main overhead comes from the pre-generation smoothing stage, where multiple perturbed candidates are explored and evaluated before the final response is produced. This design differs from decoding-time defenses whose cost often grows with the output length: once the smoothed input is selected, subsequent generation proceeds as in standard decoding, so the additional overhead does not scale with the length of the generated response. Nevertheless, the pre-generation cost can remain non-negligible in latency-sensitive applications, especially when the number of perturbation trials is large or the base model is expensive to query. *A practical mitigation is to parallelize the exploration of perturbed candidates. Since different smoothing trials are independent before the final selection step, they can be executed concurrently across multiple devices or batched within the same inference pipeline.* This makes the wall-clock latency closer to that of a single smoothing trial rather than the sum of all trials, substantially reducing the practical overhead. Moreover, early stopping can be adopted when a candidate has already activated a sufficiently stable safe response, further improving efficiency in real deployments. Future work may investigate more efficiency-aware perturbation policies, adaptive trial allocation, and lightweight surrogate-scoring mechanisms to reduce pre-generation costs without sacrificing robustness.

Second, our method relies strongly on the inherent safe bound of the underlying LLM. The goal of smoothing is not to teach the model new safety knowledge or introduce an external safety criterion from scratch. Instead, it aims to move adversarial or unstable inputs toward regions where the model's existing safe bound can be more reliably triggered. As discussed in our boundary analysis in Appendix B, this assumption is crucial: when the base model already possesses a reasonably reliable safe bound, small density-seeking perturbations can improve the stability of safety behavior by moving inputs away from fragile low-density regions. However, when the base model is under-aligned, domain-specialized, insufficiently safety-tuned, or intrinsically unable to distinguish harmful from benign requests in a given domain, smoothing alone may provide limited benefit. In such cases, the method may fail to induce safe behavior because there is no reliable safety bound within which the perturbation process can recover. This limitation is also consistent with our observation that stronger safety backbones or external safety components, such as the boundary cases analysis reported in Appendix B, can provide a more reliable foundation for smoothing to operate on. *Therefore, our approach is best viewed as complementary to model alignment rather than a substitute for it. As pretraining, instruction tuning, reinforcement learning, constitutional alignment, and safety-specialized classifiers continue to improve the quality and coverage of LLM safe bounds, the effectiveness of our smoothing-based stabilization is expected to improve accordingly.*

Third, it is important to clarify the scope of our contribution. Our method does not claim to enhance the fundamental safety capability of an LLM in the sense of enabling it to recognize entirely new harmful intents, acquire missing safety knowledge, or reason correctly about safety cases that the base model cannot handle. Instead, our contribution lies in improving the stability and reliability of the model's existing safety behavior during inference. Many jailbreak prompts exploit fragile regions of the input space, where the model's safety behavior becomes unstable even though it may refuse semantically similar harmful requests in normal phrasing. By reframing smoothing as a density-seeking process, our method attempts to relocate such inputs toward better-supported regions of the model's learned distribution, where its inherent safety boundary is more likely to function consistently. In this sense, our method should be understood as a safety-stabilization mechanism rather than a safety-capability enhancement mechanism. *This distinction also explains both its strengths and limitations: it can improve robustness against attacks that destabilize otherwise available safety behavior, but it cannot fully compensate for missing alignment, incorrect safety understanding, or fundamentally weak safety boundaries.*

Overall, these limitations suggest several directions for future work. On the efficiency side, adaptive perturbation, parallel candidate exploration, early stopping, and budget-aware smoothing policies can reduce the practical inference overhead. On the safety side, smoothing can be integrated with stronger alignment procedures, safety reward models, or guard models to provide a more reliable boundary for the perturbation process to activate. More broadly, our study indicates that inference-time defenses should not only focus on detecting harmful prompts but also on understanding where the model's safety behavior becomes unstable and how inputs can be guided back to reliable regions.

Table 5: The over refusal rate on boundary cases.

Methods	Over Refusal Rate (%)↓			
	OKTest	XSTest	ORBench	FalseReject
LLaMA3.1	2.67	6.80	28.35	45.49
SmoothLLM	3.33	8.80	50.27	64.24
EmbeddingSmooth	19.00	10.40	46.78	64.62
SelfDenoise	10.67	13.60	50.11	62.49
SemanticSmooth	13.33	20.40	54.97	62.76
GradientSmooth	6.67	9.20	41.17	60.74
w/o Gradient Guidance	-	-	49.96	67.73
LlamaGuard3	0.00	4.40	21.61	24.43
GradientSmooth	2.00	6.60	38.06	41.53
LlamaGuard4	0.33	6.40	10.24	23.42
GradientSmooth	0.67	6.80	10.92	23.93

B Over Refusal on Boundary Cases

To specifically probe regions where existing LLMs' safe bounds are more likely to be unreliable, we further introduce four boundary-case benchmarks: OKTest [Shi et al., 2024], XSTest [Röttger et al., 2024], ORBench [Cui et al., 2025], and FalseReject [Zhang et al., 2025]. These benchmarks consist of benign or permissible requests that lie close to the safe bounds and are therefore more likely to expose instability in current safety mechanisms. We use them as targeted stress tests for safe-bound reliability, with refusal rates serving as an observable manifestation of such instability rather than as the primary object of evaluation. Table 5 reports the refusal rate on these boundary-case benchmarks. Overall, the results show that excessive refusals on boundary cases are closely tied to the reliability of the underlying safety boundary. For LLaMA3.1, whose safe bounds are relatively less stable on challenging boundary cases, refusal rates become more pronounced on ORBench and FalseReject. This suggests that when the base LLMs are already uncertain on difficult boundary inputs, repeated perturbation-based trials may expose or amplify such instability.

Compared with existing smoothing-based defenses, our method consistently yields lower over-refusal. On LLaMA3.1, our approach obtains lower over-refusal on most boundary benchmarks, especially on ORBench and FalseReject. This indicates that the increased over-refusal is not a new issue introduced by our gradient-guided design, but rather a common side effect of smoothing-based methods that repeatedly explore perturbed variants of the input. In fact, our gradient guidance partially mitigates this problem by steering perturbations toward more reliable regions, rather than relying on purely random or unguided ones. The ablation without gradient guidance further supports this interpretation. Removing gradient guidance leads to substantially higher over-refusal, increasing from 41.17 to 49.96 on ORBench and from 60.74 to 67.73 on FalseReject. This contrast suggests that utility degradation in boundary cases primarily stems from repeated smoothing under an unstable safe bound, whereas our gradient-guided perturbation helps reduce unnecessary refusals.

To further examine how the reliability of the underlying safe bound affects smoothing-based stabilization, we conduct an additional analysis using LlamaGuard3-8B [Llama Team, 2024] and LlamaGuard4-12B [Meta, 2025] as stronger safe-bound proxies. It is worth noting that LlamaGuard models are safety classifiers rather than generative models, and are used here only for analysis. We evaluate them in the same multi-trial defense setting with $K = 10$, in which any "unsafe" signal triggers the refusal decision. The results show a clear dependence on the strength of the safe bounds. With LlamaGuard3, which is contemporaneous with LLaMA3.1 and still exhibits an insufficiently stable safe bound on difficult boundary cases, we observe a similar phenomenon: applying our GradientSmooth noticeably increases refusal rates on ORBench and FalseReject. More importantly, this limitation is significantly alleviated as the safe bound strengthens. With LlamaGuard4, which provides a more reliable safe bound than LlamaGuard3, applying GradientSmooth introduces only marginal additional over-refusal: the rate changes from 10.24

Table 6: Repeat experiments with 5 random seeds on LLAMA3.1.

Seed	Jailbreak Attacks (%)↓			
	GCG	AutoDAN	DeepInception	PAIR
0	2 / 0	0 / 6	14 / 8	10 / 2
2	6 / 4	2 / 4	18 / 6	12 / 4
4	4 / 4	2 / 6	16 / 4	16 / 6
42	6 / 2	0 / 4	14 / 10	18 / 6
2025	6 / 2	0 / 4	18 / 0	14 / 4
Avg.	4.4 / 2.4	0.8 / 4.8	16.0 / 5.6	14.0 / 4.4
95% CI.	2.1 / 2.1	1.4 / 1.4	2.5 / 4.8	4.0 / 2.1

to 10.92 on ORBENCH and from 23.42 to 23.93 on FALSEREJECT. *This result supports our scope claim: our method is not intended to create new safety capabilities but to stabilize and more effectively activate the model's existing safe bounds. Therefore, as base models continue to improve through stronger alignment and safety training, smoothing-based stabilization is expected to preserve utility more effectively while maintaining robustness.*

C Confidence Intervals

To further examine the statistical reliability of our results, we repeat the main LLAMA3.1 experiments with five random seeds and report the mean performance together with the 95% confidence interval in Table 6. Here, seed 0 corresponds to the exact random seed used in our main experimental results. For fairness and consistency, all baseline methods and base LLM results reported in the main tables are also evaluated under this same fixed seed. The additional seeds are therefore used not to change the primary comparison protocol, but to assess whether the observed effectiveness of our method is robust to randomness in the defense procedure. Overall, the results show that our method exhibits stable behavior across different random seeds. Across GCG, DEEPINCEPTION, and PAIR, the average jailbreak success rates remain low, with moderate 95% confidence intervals. For example, the average rates are 4.4 / 2.4 on GCG, 16.0 / 5.6 on DEEPINCEPTION, and 14.0 / 4.4 on PAIR, indicating that the defense effect observed under seed 0 is not an isolated outcome caused by a favorable random seed. AUTODAN shows slightly larger relative variation because the absolute attack success rates are already very low across seeds, where small changes in the number of successful attacks can lead to visible percentage fluctuations. Nevertheless, its variance remains moderate and does not alter the main conclusion. These results suggest that our method is statistically stable under repeated random trials. More importantly, since seed 0 is shared by the base model, all baselines, and our method in the main experiments, the primary comparisons remain controlled and fair, while the multi-seed analysis provides additional evidence that our reported gains are not due to seed-specific randomness.

D Computational Cost

We analyze the efficiency of different inference-time defenses from two perspectives: *latency cost* and *compute cost*. Latency cost refers to the end-to-end wall-clock inference time relative to the base LLM, reflecting the user-facing delay during deployment. Compute cost instead measures the total amount of model computation consumed by all trials, including computations that can be executed in parallel. These two quantities are different: parallel execution can significantly reduce latency, but it does not eliminate the aggregate compute required by multiple trials. Table 7 reports the measured latency cost under two inference settings: 100 decoding tokens on GCG and 1000 decoding tokens on ALPACAEVAL. The shorter setting highlights the relative impact of pre-generation defense overhead, while the longer setting evaluates how this overhead is amortized during long-form generation. Our method performs a fixed number of gradient-guided smoothing steps before generation, and thus its additional latency is largely independent of the final output length. As a result, the relative latency decreases as the decoding length increases. For example, the pipeline implementation decreases from 2.20× at 100 tokens to 1.59× at 1000 tokens, while the parallel implementation further decreases from 1.70× to 1.06×. This shows that GRADIENTSMOOTH is particularly friendly to long-form

Table 7: Computational cost across different inference settings.

Methods	Latency Cost		Compute Cost
	100	1000 Tokens	
BASE LLM	$1.00\times$	$1.00\times$	$1.00\times$
SMOOTHLLM	$2.88\times$	$2.70\times$	$11.00\times$
EMBEDDINGSMOOTH	$2.37\times$	$1.21\times$	$300t + 1.00\times$
SELFDENOISE	$5.15\times$	$1.75\times$	$700t + 1.00\times$
SEMANTICSMOOTH	$7.54\times$	$1.97\times$	$950t + 1.00\times$
GRADIENTSMOOTH *pipeline*	$2.20\times$	$1.59\times$	$600t + 1.00\times$
GRADIENTSMOOTH *parallel*	$1.70\times$	$1.06\times$	$600t + 1.00\times$

generation, where the pre-generation smoothing overhead is amortized by the autoregressive decoding cost. We further estimate the aggregate compute cost by separating the standard generation cost from the additional pre-generation overhead. Let t denote the cost of decoding one token. In Table 7, the term $1.00\times$ represents the normal autoregressive decoding cost of the base LLM under the corresponding output length, while the additional t-based term denotes the extra computation introduced before generation. This notation clarifies the scaling behavior: methods with a fixed pre-generation overhead are written as "extra cost $+1.00\times$", whereas methods that require multiple full generations scale directly with the output length. For our method, the worst-case additional computation consists of refusal-checking generations and gradient-guided smoothing updates. With $K = 10$ trials and $T = 5$ gradient steps per trial, the pre-generation overhead is estimated as

$$K \times [30t + T \text{ forward} + T \text{ backward}] \approx 600t.$$

Therefore, the aggregate compute cost of GRADIENTSMOOTH is reported as $600t + 1.00\times$, where $1.00\times$ corresponds to the standard decoding cost after the defense procedure. The same aggregate compute estimate applies to both the pipeline and parallel implementations, since both execute the same worst-case set of trials. Their difference lies in latency rather than total compute: the pipeline implementation runs trials sequentially and can benefit from early stopping, whereas the parallel implementation executes trials concurrently, substantially reducing wall-clock latency but not eliminating the aggregate computation across all trials. This decomposition also clarifies the difference between GRADIENTSMOOTH and prior smoothing-based defenses. SMOOTHLLM requires multiple complete generations, so its compute cost scales with the output length and is reported as $11.00\times$ under $K = 10$ trials. In contrast, EMBEDDINGSMOOTH, SELFDENOISE, SEMANTICSMOOTH, and GRADIENTSMOOTH introduce additional pre-generation computation on top of a single standard generation. Among them, SELFDENOISE depends on the full input length, and SEMANTICSMOOTH is affected by the cost and compression efficiency of paraphrasing or summarization. The additional smoothing computation occurs before final generation and does not grow directly with output length, but its gradient calculations depend on input length. Parallel trials reduce wall-clock latency without eliminating aggregate computation. Consequently, although it incurs moderate additional aggregate compute, its relative overhead is amortized as the decoding length increases, and its parallel implementation achieves near-standard decoding latency in long-form generation.

We note that our method introduces limited memory overhead. All model parameters are frozen during smoothing and decoding. Unlike full fine-tuning, our method does not require maintaining parameter gradients, optimizer moments, or additional trainable parameter copies of the same scale as the model. The backward computation is used only to obtain gradient information with respect to the input representations to guide the smoothing direction. Moreover, our sparse Gauss-Southwell update just selects a small subset of high-impact token positions for perturbation, so the gradient information is ultimately aggregated over a small portion of the input sequence rather than used to update the model parameters. In practice, this makes the memory requirement close to standard inference. All experiments can be run on a single NVIDIA A100 40GB GPU.

Table 8: Ablation study on the two components of the context-aware density objective on LLAMA3.1.

Loss	Jailbreak Attacks (%)↓				Utility↑		
	GCG	AutoDAN	DeepInception	PAIR	AlpacaEval	MtBench	IFEval
$\mathcal{L}_p$	40 / 26	2 / 8	16 / 10	18 / 2	96.78	7.61	76.31
$\mathcal{L}_s$	8 / 10	0 / 6	12 / 4	18 / 2	94.53	7.22	70.06
$\mathcal{L}_p + \mathcal{L}_s$	2 / 0	0 / 6	14 / 8	10 / 2	96.25	7.42	75.97

E Extended Experiments

E.1 Loss Component Ablation

To investigate the contributions of the two components of our context-aware density objective, we separately evaluate the preceding loss $\mathcal{L}_p$ and the succeeding loss $\mathcal{L}_s$, as shown in Table 8. The preceding loss encourages consistency with the previous context, helping preserve semantic fluency and instruction-following utility, but provides relatively limited safety improvements when used alone. In contrast, the succeeding loss incorporates information from the subsequent context and generally provides stronger safety guidance, at the cost of greater utility degradation. Combining both components achieves a better overall safety–utility balance, reducing the attack success rates to 2% / 0% on GCG and 10% / 2% on PAIR, while maintaining competitive utility across all three benchmarks. These results indicate that the two losses play complementary roles in balancing safety activation and semantic consistency.

E.2 Effect of Reasoning on Safety Defense

We further investigate the reasoning capability of modern LLMs for safety defense. To this end, we conduct additional experiments on QWEN3-8B [Yang et al., 2025a] under two inference modes: disabling the reasoning process (*w/o think*) and enabling the reasoning process (*w/ think*). As shown in Table 9, reasoning itself provides a certain degree of passive defense. For example, enabling thinking reduces the jailbreak success rate of the base Qwen3 model from 74 / 60 to 50 / 22 on GCG, and from 98 / 46 to 98 / 38 on DEEPINCEPTION. This suggests that explicit reasoning can help the model assess the malicious intent of certain adversarial prompts more effectively before producing the final response. However, the improvement brought by reasoning alone is limited and attack-dependent. On PAIR, enabling thinking does not reduce the base model's jailbreak rate, which remains 100 / 68. On DEEPINCEPTION, the first score also remains extremely high even with reasoning enabled. These results indicate that reasoning is not a reliable standalone defense: although it may help the model deliberate over the safety implications of certain prompts, it does not fundamentally prevent adversarial inputs from pushing the model into fragile or unsafe regions. In this sense, reasoning behaves more like a passive self-checking mechanism during generation, rather than an active mechanism for stabilizing the model's safe bound. The effectiveness on Qwen3 is attack-dependent: GradientSmooth substantially improves robustness against GCG, but reduces the PAIR harmful-output rate only modestly (from 68% to 62%), while non-refusal rates on DeepInception remain high. This difference arises because gradient-guided smoothing can effectively restore safety behavior disrupted by token-level attacks such as GCG, but offers limited benefits against semantically coherent attacks such as PAIR and DeepInception when the base model itself fails to reliably recognize or reject the embedded harmful intent. In addition, reasoning and smoothing are complementary rather than mutually exclusive. Reasoning improves the model's internal deliberation after receiving the input, while GRADIENTSMOOTH acts before generation by perturbing the input representation toward a more stable region of the model's learned safe bounds. Therefore, our method can still provide additional robustness on top of reasoning-capable models. Overall, these results support our scope claim. Reasoning can enhance safety to some extent, but it does not replace inference-time safety stabilization. Our approach addresses a different failure mode: instead of relying on the model to reason its way out of an adversarial prompt, it first stabilizes the input space, allowing the model's existing safety mechanism to be activated more reliably. As reasoning models continue to improve, our method can serve as a complementary inference-time defense that further strengthens their robustness against jailbreak attacks.

Table 9: Safety defense on reasoning models.

Methods	Jailbreak Attacks (%)↓			
	GCG	AutoDAN	DeepInception	PAIR
QWEN3 w/o think	74 / 60	0 / 0	98 / 46	100 / 68
SELFDENOISE	32 / 42	0 / 0	98 / 46	96 / 66
SEMANTICSMOOTH	48 /	0 / 0	98 / 44	98 / 68
OURS	12 / 14	0 / 0	98 / 42	92 / 62
QWEN3 w/ think	50 / 22	0 / 0	98 / 38	100 / 68
SELFDENOISE	12 / 10	0 / 0	98 / 38	98 / 66
SEMANTICSMOOTH	18 /	0 / 0	96 / 50	98 / 66
OURS	6 / 4	0 / 0	96 / 28	96 / 62

Table 10: Results of smoothing on QWEN2.5 models with different scales.

Methods	Jailbreak Attacks (%)↓				Utility↑		
	GCG	AutoDAN	DeepInception	PAIR	AlpacaEval	MtBench	IFEval
QWEN2.5-7B	68 / 20	70 / 92	94 / 78	96 / 68	97.14	7.53	72.27
OURS	10 / 4	8 / 26	42 / 34	40 / 22	96.43	7.48	70.61
QWEN2.5-14B	58 / 0	0 / 0	80 / 22	90 / 34	98.76	7.76	80.41
OURS	4 / 0	0 / 0	28 / 12	34 / 18	97.82	7.70	79.92

E.3 Effect of Model Scale

We further evaluate whether the effectiveness of GRADIENTSMOOTH transfers to larger models. Specifically, we conduct additional experiments on QWEN2.5-14B and compare it with QWEN2.5-7B, as reported in Table 10. The results show that increasing model scale can improve the base model's safety behavior on some attacks, but it does not eliminate the vulnerability to jailbreak prompts. For example, QWEN2.5-14B shows substantially lower attack success rates on GCG and AUTODAN, suggesting that the larger model has stronger inherent safe bounds. However, the base 14B model still remains highly vulnerable under several attacks, with jailbreak rates of 80 / 22 on DEEPINCEPTION and 90 / 34 on PAIR. This indicates that model scaling alone is insufficient to ensure robustness against adversarial jailbreaks. GRADIENTSMOOTH remains effective on the larger model. On QWEN2.5-14B, our method reduces the jailbreak success rate from 58 / 0 to 4 / 0 on GCG, from 80 / 22 to 28 / 12 on DEEPINCEPTION, and from 90 / 34 to 34 / 18 on PAIR. These reductions are consistent with the trend observed on QWEN2.5-7B, where our method also substantially lowers the attack success rates across all four jailbreak attacks. This suggests that the proposed smoothing mechanism is not limited to smaller models, but can also stabilize the safety behavior of larger instruction-tuned LLMs. We also observe that the utility degradation remains minor after applying GRADIENTSMOOTH. For QWEN2.5-14B, ALPACAEVAL decreases only from 98.76 to 97.82, MTBENCH from 7.76 to 7.70, and IFEVALfrom 80.41 to 79.92. The small drops indicate that the improved safety does not come at the cost of severe degradation in general instruction-following ability. Overall, these results support our claim that GRADIENTSMOOTH leverages and stabilizes the model's existing safe bounds. Larger models may provide stronger base safety, but GRADIENTSMOOTH can further activate this boundary under adversarial inputs, yielding additional robustness with negligible utility loss.

E.4 Evaluation on More Complex Jailbreak Attacks

To further evaluate the robustness of GRADIENTSMOOTH beyond the standard jailbreak attacks used in prior smoothing-based defenses, we conduct additional experiments on more complex attack settings, as shown in Table 11. These attacks include four flipping-based attacks [Liu et al., 2025], i.e., flipping characters in sentence (FCS), flipping characters in word (FCW), fool model mode (FMM), and flip word order (FWO), as well as a reasoning-based jailbreak attack, H-COT [Kuo et al., 2025].

Table 11: Evaluation on more complex jailbreak attacks.

Methods	Jailbreak Attacks (%)↓				
	FCS	FCW	FMM	FWO	H-CoT
QWEN2.5	100 / 4	100 / 6	100 / 6	96 / 90	100 / 98
SELFDENOISE	100 / 4	100 / 6	100 / 6	96 / 90	100 / 96
SEMANTICSMOOTH	100 / 4	100 / 6	100 / 6	94 / 86	98 / 92
OURS	100 / 4	100 / 2	100 / 6	78 / 64	98 / 88
LLAMA3.1	100 / 10	100 / 34	100 / 18	100 / 90	100 / 94
SELFDENOISE	100 / 8	96 / 26	100 / 12	100 / 88	98 / 90
SEMANTICSMOOTH	100 / 10	100 / 30	100 / 18	98 / 86	92 / 86
OURS	6 / 0	14 / 4	24 / 4	16 / 16	74 / 66
LLAMAGUARD3	36 / -	18 / -	44 / -	4 / -	36 / -
SELFDENOISE	28 / -	12 / -	40 / -	4 / -	32 / -
SEMANTICSMOOTH	76 / -	60 / -	80 / -	4 / -	34 / -
OURS	2 / -	4 / -	0 / -	0 / -	14 / -

The flipping-based attacks perturb the surface form of malicious instructions and require the model to recover the underlying harmful intent from distorted inputs, while H-COT embeds malicious intent into multi-step reasoning contexts, making the unsafe objective more deeply entangled with the reasoning process. Therefore, these attacks provide a more challenging testbed for examining whether smoothing-based defenses can generalize to complex adversarial prompts. For the flipping-based attacks, the base LLAMA3.1 model remains highly vulnerable. Across FCS, FCW, FMM, and FWO, the base model obtains attack success rates of 100 / 10, 100 / 34, 100 / 18, and 100 / 90, respectively. Existing smoothing-based baselines only provide limited improvements. For example, SELFDENOISE still yields 100 / 8 on FCS, 100 / 12 on FMM, and 100 / 88 on FWO, while SEMANTICSMOOTH remains similarly vulnerable with 100 / 10, 100 / 30, 100 / 18, and 98 / 86 across the four attacks. In contrast, GRADIENTSMOOTH substantially reduces the attack success rates to 6 / 0, 14 / 4, 24 / 4, and 16 / 16, respectively. This demonstrates that our method is effective not only against suffix-optimization and prompt-level jailbreaks, but also against attacks that obscure harmful intent through input perturbations. An interesting observation is that the base model behaves differently across these flipping strategies. For character-level or more complex perturbations such as FCS, FCW, and FMM, LLAMA3.1 often struggles to correctly recover the malicious intent from flipping perturbations, leading to low harmfulness but also weak refusal. For word-level flipping such as FWO, the perturbation is easier to interpret, and the model can more readily reconstruct the harmful request, which explains the much higher attack success rate of 100 / 90. GRADIENTSMOOTH consistently improves refusal activation across these settings. By perturbing the input toward a more stable and reliable region, our method appears to weaken the jailbreak-inducing contextual cues while preserving the model's ability to recognize the unsafe intent.

We further evaluate H-COT as a representative reasoning-based jailbreak attack. Compared with flipping-based attacks, H-COT is more challenging because the malicious objective is embedded within multi-step reasoning chains rather than expressed as a simple obfuscated instruction. On LLAMA3.1, the base model achieves a very high attack success rate of 100 / 94, indicating that this attack already exceeds the reliable safe bound of the model. Although GRADIENTSMOOTH reduces the attack success rate to 74 / 66, the absolute value remains relatively high. This suggests that our method still improves robustness to reasoning-based jailbreaks, but smoothing alone cannot fully recover the correct safety behavior when the base model itself fails to reliably identify unsafe intent in complex reasoning. To better understand this limitation, we additionally conduct a targeted analysis using LLAMAGUARD3 as a base safety model with stronger safe bounds. Following the same multi-trial defense setting, LLAMAGUARD3 reduces the H-COT attack success rate from LLAMA3.1's 100 / 94 to 36 / 32. When GRADIENTSMOOTH is further applied on top of LLAMAGUARD3, the attack success rate is reduced to 14 / 2. It is important to note that LLAMAGUARD3 is a safety classifier rather than a generative model, and we use it here only for analysis, similar to our over-refusal experiments. Its purpose is not to replace the generative base model, but to examine how our method behaves when the underlying model provides a more accurate and stable safe bound.

Table 12: Comparison with other lines of defense.

Methods	Jailbreak Attacks (%)↓				Utility↑		
	GCG	AutoDAN	DeepInception	PAIR	AlpacaEval	MtBench	IFEval
LLaMA3.1	72 / 52	82 / 78	74 / 62	82 / 50	97.62	7.63	76.52
SafeDecoding	2 / 0	2 / 2	6 / 4	4 / 4	91.05	7.03	69.02
LlamaGuard3	2 / 0	0 / 0	6 / 0	42 / 22	96.89	7.58	76.15
Ours	2 / 0	0 / 6	14 / 8	10 / 2	96.25	7.42	75.97

Table 13: Combination with other lines of defense.

Methods	Jailbreak Attacks (%)↓			
	GCG	AutoDAN	DeepInception	PAIR
SafeProbing on Qwen2.5	26 / 4	4 / 6	48 / 28	52 / 16
+ GradientSmooth	4 / 0	0 / 0	16 / 10	20 / 6
CircuitBreaker on Llama3	54 / 26	100 / 100	4 / 4	14 / 14
+ GradientSmooth	26 / 26	86 / 86	2 / 2	14 / 4

Overall, these results support our central scope claim. GradientSmooth does not create new safety capabilities from scratch. Instead, it stabilizes and better activates the safe bounds already present in the underlying model. For complex perturbation-based attacks, where the base model can still recover the unsafe intent but is destabilized by adversarial context, GradientSmooth yields strong and consistent improvements. For reasoning-based attacks such as H-CoT, the remaining attack success reflects the limitation of the base model's safety understanding with deeply entangled reasoning. Fortunately, the LlamaGuard3 analysis shows that once the underlying safe bounds become stronger, GradientSmooth can more effectively leverage it, leading to substantially improved robustness even in these challenging settings.

E.5 Comparison with Other Lines of Defense

Although our main comparisons focus on smoothing-based defenses for a controlled and fair evaluation, we further include two representative defenses from other technical lines, SafeDecoding [Xu et al., 2024] and LlamaGuard3 [Llama Team, 2024], as additional references. SafeDecoding is a decoding-time defense that modifies the generation process to suppress unsafe continuations and encourage safer responses. Unlike smoothing-based methods, it directly intervenes in token generation and can provide strong safety protection but may also affect generation quality or instruction-following behavior. LlamaGuard3 is a classifier-based safety filter. Instead of changing the generation process, it detects unsafe inputs and blocks or rejects them before passing them to the target model. Since LlamaGuard3 is not a generative model, we use it here as an external input-filtering baseline rather than as a replacement for the base LLM.

The results are shown in Table 12. LLaMA3.1 is highly vulnerable to all four jailbreak attacks, with attack success rates of 72 / 52 on GCG, 82 / 78 on AutoDAN, 74 / 62 on DeepInception, and 82 / 50 on PAIR. At the same time, it achieves strong utility scores on benign benchmarks, including 97.62 on AlpacaEval, 7.63 on MTBench, and 76.52 on IFEval. This confirms the central trade-off considered in our experiments: the base model has good general utility, but its safety behavior is unstable under adversarial prompts. SafeDecoding achieves strong robustness across all jailbreak attacks. It reduces the attack success rates to 2 / 0 on GCG, 2 / 2 on AutoDAN, 6 / 4 on DeepInception, and 4 / 4 on PAIR. However, this strong safety comes with a clear utility drop: AlpacaEval decreases from 97.62 to 91.05, MTBench from 7.63 to 7.03, and IFEval from 76.52 to 69.02. This suggests that directly constraining the decoding process can effectively suppress harmful outputs, but it may also interfere with normal response generation and reduce helpfulness or instruction-following quality. LlamaGuard3 also performs well on most attacks when used as

Table 14: The influence of Sparsity Ratio $r\%$ on LLAMA3.1.

r	Jailbreak Attacks (%)↓				Utility↑		
	GCG	AutoDAN	DeepInception	PAIR	AlpacaEval	MtBench	IFEval
5	14 / 8	4 / 12	36 / 26	34 / 12	97.41	7.63	76.34
10	10 / 8	2 / 10	32 / 20	18 / 8	97.03	7.61	75.83
15	2 / 0	0 / 6	14 / 8	10 / 2	96.25	7.42	75.97
20	6 / 2	0 / 6	16 / 10	14 / 4	95.03	7.53	73.38
30	12 / 6	2 / 6	26 / 18	20 / 6	94.41	7.44	72.46
50	16 / 14	4 / 10	48 / 24	22 / 8	92.92	7.43	71.90
100	38 / 34	40 / 38	62 / 28	46 / 26	92.42	7.20	68.79

an input filter. It reduces the attack success rates to 2 / 0 on GCG, 0 / 0 on AUTODAN, and 6 / 0 on DEEPINCEPTION. Meanwhile, it preserves utility relatively well, with 96.89 on ALPACAEVAL, 7.58 on MTBENCH, and 76.15 on IFEVAL. However, its performance is less robust on PAIR, where the attack success rate remains 42 / 22. This indicates that classifier-based filtering can be effective when unsafe intent is explicitly recognizable, but may generalize less reliably to adaptive or conversational jailbreaks where the harmful intent is hidden behind indirect interaction patterns. GRADIENTSMOOTH provides a more balanced trade-off. Although it is not always the lowest across all attacks, it remains competitive with these stronger defense paradigms while preserving much higher utility than SAFEDECODING. Compared with LLAMAGUARD3, it performs better on PAIR, reducing the attack success rate from 42 / 22 to 10 / 2, while maintaining comparable utility scores.

To investigate whether GRADIENTSMOOTH provides additional benefits when combined with stronger defenses, we further evaluate its integration with SafeProbing and CircuitBreaker. As shown in Table 13, GRADIENTSMOOTH consistently improves SafeProbing across all four attacks. For instance, the non-refusal/harmful-output rates decrease from 48 / 28 to 16 / 10 on DeepInception and from 52 / 16 to 20 / 6 on PAIR. When combined with CircuitBreaker, GRADIENTSMOOTH also achieves additional improvements, although the gains are less pronounced and vary across attacks. Notably, it reduces the harmful-output rate on PAIR from 14 to 4, while leaving the rate on GCG unchanged. These results suggest that density-guided smoothing can further stabilize safety behavior even in models equipped with stronger defenses, although the incremental benefits depend on the underlying defense and attack characteristics.

F Analyses on other Hyperparameters

F.1 Sparsity Ratio

We examine the sparsity ratio $r\%$, which controls the update scale per iteration by applying gradient-guided perturbations only to the top-$r\%$ tokens with the largest absolute gradients (Table 14). Considering safety, aside from AUTODAN which remains relatively stable, the other jailbreak attacks exhibit a non-monotonic dependence on $r\%$. Varying from 5% to 100% yields a local optimum around 15%, with performance improving up to that point and degrading beyond. In contrast, utility exhibits an approximately monotonic decline as r increases. A plausible explanation is that sparsity ratio shapes where the update process concentrates. Moderately sparse and high-magnitude gradients focus on a small subset of salient tokens in the input sequence, maximizing the shift toward higher-likelihood neighborhoods with minimal changes in representation. If the updates are too sparse (e.g., $r\% \leq 10\%$), the perturbation lacks sufficient coverage to escape the semantic margin of attacks, resulting in limited safety gains. If the updates are too dense (e.g., $r\% \geq 30\%$), the simultaneous modifications disrupt the coherence of the input sequence and adversely push the representation into lower-density regions, degrading both safety and utility.

F.2 Iterations

We analyze the influence of the number of gradient-guided smoothing iterations T, as shown in Table 15. Here, T controls the optimization depth within each smoothing trial, i.e., how many gradient-guided update steps are performed before checking whether the defense mode is activated.

Table 15: The influence of Iterations T on LLAMA3.1.

T	Jailbreak Attacks (%)↓				Utility↑		
	GCG	AutoDAN	DeepInception	PAIR	AlpacaEval	MtBench	IFEval
5	2 / 0	0 / 6	14 / 8	10 / 2	96.25	7.42	75.97
4	6 / 6	2 / 6	24 / 10	20 / 6	96.03	7.46	75.78
3	10 / 8	8 / 6	28 / 10	28 / 10	96.53	7.54	74.86
2	10 / 8	6 / 8	34 / 10	36 / 10	96.66	7.60	76.34
1	10 / 8	18 / 20	46 / 14	38 / 16	97.29	7.61	76.52

Table 16: The influence of Trials K on LLAMA3.1.

K	Jailbreak Attacks (%)↓				Utility↑		
	GCG	AutoDAN	DeepInception	PAIR	AlpacaEval	MtBench	IFEval
10	2 / 0	0 / 6	14 / 8	10 / 2	96.25	7.42	75.97
8	6 / 4	4 / 6	18 / 10	14 / 4	96.26	7.53	75.97
6	12 / 6	4 / 6	34 / 10	26 / 12	96.65	7.61	76.15
4	12 / 10	12 / 14	34 / 10	38 / 8	97.14	7.59	76.11
2	22 / 14	36 / 36	54 / 12	52 / 22	97.27	7.61	76.44
1	34 / 26	68 / 68	52 / 16	64 / 28	97.22	7.63	76.50

The main experiments use $T = 5$, which corresponds to the setting highlighted in the table. Overall, increasing T increases jailbreak robustness, while utility scores remain relatively stable. For example, when T increases from 1 to 5, the attack success rate decreases from 10 / 8 to 2 / 0 on GCG, from 18 / 20 to 0 / 6 on AUTODAN, from 46 / 14 to 14 / 8 on DEEPINCEPTION, and from 38 / 16 to 10 / 2 on PAIR. This suggests that a larger number of update steps helps the perturbed input move further toward a stable region where the model's safe bounds can be more reliably activated. At the same time, the utility degradation as T increases is mild. Across $T = 1$ to $T = 5$, ALPACAEVAL remains within a narrow range from 97.29 to 96.25, MTBENCH from 7.61 to 7.42, and IFEVAL from 76.52 to 75.97. This indicates that the additional smoothing iterations mainly improve safety activation without substantially damaging benign instruction-following ability. We therefore choose $T = 5$ as the default setting in the main experiments, as it provides the strongest overall safety performance while maintaining competitive utility.

F.3 Trials

We further study the influence of the number of smoothing trials K, as reported in Table 16. Different from T, which controls the optimization depth of each individual trial, K controls how many independent smoothing attempts are performed. The main experiments use $K = 10$, which is highlighted in the table. Increasing K generally improves robustness across attacks, because more trials provide more opportunities to find a perturbed input that activates the model's refusal behavior. For instance, increasing K from 1 to 10 reduces the attack success rate from 34 / 26 to 2 / 0 on GCG, from 68 / 68 to 0 / 6 on AUTODAN, from 52 / 16 to 14 / 8 on DEEPINCEPTION, and from 64 / 28 to 10 / 2 on PAIR. Meanwhile, the utility scores remain stable, with only small variations on ALPACAEVAL, MTBENCH, and IFEVAL. An important observation is that the improvement with respect to K is not always a smooth, gradual decrease. Instead, performance often exhibits abrupt drops when K crosses certain values. This pattern reflects the instance-level nature of our defense: different attack samples require different numbers of trials before the refusal behavior is successfully triggered. Some samples can be defended within only one or two trials, while others require more attempts. As a result, early stopping is particularly useful in the pipeline implementation, since the remaining trials can be skipped once a successful defensive perturbation is found. For latency-sensitive scenarios, these trials can also be executed in parallel, reducing wall-clock latency to approximately that of a single trial while still benefiting from the robustness gains provided by multiple attempts. We therefore use $K = 10$ as the default setting in the main experiments, as it offers strong robustness while preserving utility and allowing flexible pipeline or parallel deployment.

Table 17: Effects of gradient-guided smoothing on input prefixes. T is the perturbation steps. **Match Rate** denotes the strict token-level recovery rate, measuring whether the token decoded from the perturbed embedding exactly matches the original token. **Cosine** reports the cosine similarity between the perturbed and original token embeddings. **Mag.** is magnitude and measures the ℓ_2 norm of the perturbation vector. **Avg.** indicates the average perturbation magnitude computed over all token positions, while **Max.** reports the maximum perturbation magnitude among all positions.

T	**Match Rate**	**Cosine**	**Mag. (Avg. / Max.)**
GCG ATTACK			
1	99.91%	0.89	0.98 / 3.04
2	99.91%	0.88	1.05 / 3.27
3	99.91%	0.88	1.11 / 3.46
4	99.91%	0.87	1.18 / 3.60
5	99.91%	0.86	1.23 / 3.71
PAIR ATTACK			
1	99.89%	0.79	1.00 / 3.29
2	99.87%	0.77	1.09 / 3.55
3	99.85%	0.75	1.16 / 3.76
4	99.84%	0.74	1.22 / 3.94
5	99.83%	0.73	1.29 / 4.12

Table 18: Probability densities of perturbed input prefixes. **NLL** denotes the negative log probability (negative log-likelihood) of the input prefix, computed autoregressively under the LLM. **SoftNLL** measures a soft negative log probability, where the hard next-token prediction is replaced by the soft token distribution and evaluated via aggregation of KL-divergence. T indicates perturbation steps, with $T = 0$ corresponding to the original, unperturbed input prefix.

T	**NLL**	**SoftNLL**	**NLL**	**SoftNLL**	**NLL**	**SoftNLL**
	GCG ATTACK		PAIR ATTACK		ALPACAEVAL	
0	206.32	444.47	320.88	2125.07	113.18	741.58
1	197.49	425.91	321.46	1874.93	111.06	587.81
2	197.29	424.53	320.09	1824.39	111.57	586.69
3	195.12	422.24	319.73	1797.85	111.09	586.09
4	194.32	421.56	319.39	1774.63	110.67	585.12
5	192.71	419.12	319.46	1757.26	110.23	584.64

G Fine-Grained Analyses

G.1 Effects of Gradient-Guided Smoothing on the Input Prefixes

The results of LLAMA-3.1 in Table 17 indicate that our gradient-guided smoothing method does not rely on disrupting the original tokens or semantics to weaken adversarial attack patterns. In the case of GCG attacks, which are characterized by visibly corrupted or nonsensical token sequences, the match rate remains close to 100%, suggesting that almost no tokens are altered after perturbation. Meanwhile, the high cosine similarity further confirms that the underlying semantic representations are largely preserved. Instead of breaking attack patterns through lexical or semantic distortion, our method operates by searching for regions in the representation space where the model assigns higher probability densities to input prefixes. This mechanism effectively restores the reliability of the model's safe bound at inference time, mitigating the generalization loss introduced by distributional shifts and malicious attacks. Table 18 provides direct evidence that gradient-guided smoothing consistently shifts input representations toward regions with higher model-assigned probability. Across adversarial settings (GCG and PAIR) as well as benign evaluation data (ALPACAEVAL), both NLL and SoftNLL generally decrease as the number of perturbation steps increases, indicating

Table 19: Fine-grained refusal probability under gradient-guided smoothing. **Refusal Probability** measures the model's propensity to produce a refusal response by summing the predicted probabilities of a predefined set of refusal-related tokens and phrases (e.g., *I can't*, *I'm sorry*, *I'm unable*, *I cannot*, *I am sorry*, *I am unable*, and *is illegal and unethical*) given the input prefix. T denotes the number of gradient-guided perturbation steps, with $T = 0$ corresponding to the original, unperturbed input.

T	**Refusal Probability (%)**		
	GCG ATTACK	PAIR ATTACK	ALPACAEVAL
0	4.61	4.89	1.95
1	48.26	33.52	3.19
2	62.03	50.79	3.52
3	67.28	56.40	3.49
4	76.31	62.78	3.55
5	79.36	69.71	3.33

Table 20: Sequence-level probability of inputs. **NLL** and **SoftNLL** report the negative log probability and soft negative log probability of the input prefix, respectively. RAW x_a denotes the original malicious instruction before it is encapsulated by any attack method, while the other rows correspond to different attack strategies.

Input	**NLL**	**SoftNLL**
RAW x_a	45.36	250.71
GCG	206.32	444.47
AUTODAN	2143.78	12470.15
DEEPINCEPTION	509.00	2247.16
PAIR	320.88	2125.07

improved model confidence under the perturbed prefixes. Importantly, this trend is more pronounced in SoftNLL than in hard autoregressive likelihood (NLL), indicating that gradient-guided smoothing does not merely optimize token-level predictions. Instead, it progressively steers the global input representations toward regions that are more familiar to the LLM, leading to a more reliable decision space. Combined with the high token match rates and strong semantic preservation observed in Table 17, results suggest that our method enhances safety by restoring robust decision boundaries in representation space, rather than by obfuscating the original inputs.

G.2 Refusal Probability on Safety Defense

While the main experiments report binary safety outcomes, whether the LLM produces a refusal or a harmful response, such metrics lack gradual changes in the model's internal safety preference. To obtain a more fine-grained view, we compute the refusal probability by aggregating the predicted densities over a set of refusal tokens and phrases. As shown in Table 19, our method leads to a substantial and progressive increase in refusal probability under adversarial settings (GCG and PAIR) as the number of perturbation steps increases. This trend indicates that the model becomes increasingly inclined toward refusal behavior, even before a hard refusal is explicitly generated. In contrast, refusal probabilities on benign inputs from ALPACAEVAL remain consistently low and stable, suggesting that our method enhances safety responses without inducing noticeable over-refusal on non-adversarial prompts. Together with the representation-level analyses in Tables 17 and 18, these results provide complementary evidence that gradient-guided smoothing improves safety by reshaping the model's confidence landscape toward more reliable regions, rather than relying on brittle binary thresholds or explicit output suppression.

G.3 Motivation Analyses

Table 21: Motivation analysis on the relationship between next token likelihood and refusal probability. We append a token s to malicious input x, group tokens by $\log p(s|x)$, and report the refusal probability statistics.

$\log p(s\|x)$	**Avg.**	**Max.**	**Min.)**
$[-25, -15]$	0.929	0.999	0.000
$(-15, -13]$	0.938	0.999	0.213
$(-13, -11]$	0.941	0.999	0.354
$(-11, -9]$	0.944	0.999	0.879
$(-9, -7]$	0.944	0.999	0.890
$(-7, -5]$	0.943	0.999	0.914
$(-5, -3]$	0.944	0.999	0.922
$(-3, 0]$	0.945	0.999	0.938

Probability Degradation under Adversarial Attack Scenarios Table 20 quantifies the sequence-level probability shift induced by different attack encapsulation strategies. Compared to the raw malicious instruction x_a, all attack variants introduce a substantial drop in the input probability, despite remaining semantically coherent. Even the mildest attack transformation (GCG) decreases the total sequence probability density by approximately 8×10^{69} ($p(x) = e^{-\text{NLL}}$), while more aggressive encapsulation strategies such as DeepInception and AutoDAN lead to sharper probability drops. This observation supports our central claim that low-probability inputs in safety-critical scenarios do not necessarily arise from semantic incoherence. Instead, attack effectiveness is largely driven by the construction of long, structured prompts that push otherwise coherent inputs into model-unfamiliar low-probability regions, where sequence length plays a dominant role in probability degradation. In practice, this effect is often overlooked because likelihood is typically computed by averaging token-level probabilities across the sequence. Importantly, such probability degradation is not unique to adversarial content: as sequence length increases, benign and safe inputs will also occupy low-probability regions. As a result, our defense method does not rely on filtering or rejecting inputs based on probability thresholds. In contrast, it improves the reliability of the LLM's safe bounds in low-probability regimes by gently perturbing inputs toward higher-probability regions, allowing the model's intrinsic safety mechanisms to operate more consistently and accurately.

Safety Degradation under Next-Token Attack To further validate the motivation behind GRADIENTSMOOTH, we conduct a controlled analysis on the relationship between input likelihood and safety behavior. Given a malicious input x, we append a token s sampled from the LLAMA3.1 vocabulary and measure its conditional likelihood $p(s|x)$ together with the model's refusal probability $p(y|x, s)$. We then group the appended tokens into likelihood intervals based on $\log p(s \mid x)$ and report the average, maximum, and minimum refusal probabilities for each interval, as shown in Table 21. This analysis allows us to examine whether unsafe responses are more likely to occur when the input is perturbed toward low-probability regions.

The results show a clear pattern. Although the average refusal probability remains high across all intervals, the minimum refusal probability differs substantially between low- and high-likelihood regions. When $\log p(s \mid x)$ is very low, e.g., in $[-25, -15]$, the minimum refusal probability can drop to 0.000, indicating that some low-probability continuations can completely deactivate the model's refusal behavior. Similarly, in $(-15, -13]$ and $(-13, -11]$, the minimum refusal probabilities are only 0.213 and 0.354, respectively. In contrast, once $\log p(s \mid x)$ exceeds a moderate threshold, e.g., greater than -11, the minimum refusal probability sharply increases to 0.879 and remains consistently high in higher-likelihood intervals. For example, the minimum refusal probability reaches 0.938 in the interval $(-3, 0]$. This indicates that successful jailbreaks, reflected by extremely low refusal probability, mainly occur when the appended token lies in sufficiently low-probability regions.

This finding provides additional empirical support for our central hypothesis: jailbreak prompts often exploit fragile low-density regions of the model's input distribution, where the learned safety boundary becomes unstable. Importantly, this does not mean that low likelihood itself should be used as a direct detection criterion, since many benign long or rare inputs may also have low probability. Instead, the analysis motivates our smoothing strategy: rather than filtering inputs based on likelihood, GRADIENTSMOOTH gently perturbs the input representation toward nearby higher-density regions,

where the model's existing safe bounds can be more reliably activated. Therefore, the observed sharp increase in the minimum refusal probability from low- to moderate-likelihood intervals supports the design choice of using density-seeking smoothing to stabilize safety behavior under jailbreak attacks.

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: Contributions and scope can be viewed in the abstract and introduction sections.

   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: There is a limitation section in Appendix A.

   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: Theory assumptions and proofs are provided in Section 2.1: Motivating Formulation.

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: Experimental details are provided in Section 3.1 Experimental Setup. All hyperparameters are analyzed in Section 4: Analyses and Appendix F: Analyses on other Hyperparameters.

   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: [No]

Justification: At the time of submission, we are not able to provide a complete anonymized code release with fully reproducible scripts. The implementation is still being cleaned, anonymized, and organized to remove environment-specific dependencies and private paths. We will release the code and reproduction instructions in a future version, including scripts for running our method and baselines, configuration files, hyperparameters, random seeds, and data preparation instructions. The experimental details needed to understand the setup, including model choices, attack settings, evaluation metrics, and key hyperparameters, are described in the paper and appendix.

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: Experimental details are provided in Section 3.1: Experimental Setup.

   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: Statistical Significance is provided in Appendix C: Confidence Intervals.

   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: Compute costs are provided in Appendix D: Computational Cost.

   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.

   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: Impacts are discussed in Conclusion and Limitations.

    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: Our method is not a generative model but a safety-defense filter designed to mitigate the risk posed by LLMs.

    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 assets are properly cited.

    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: [N/A]

    Justification: The paper does not release new assets.

    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: The paper does not involve crowdsourcing nor 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: The paper does not involve crowdsourcing nor 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: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    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.