SEED: Eliciting Internal Safety for Jailbreak Defense in Large Reasoning Models via Exploratory Decoding

Yujue Wang*, Quan Zhang‡,†, Gwihwan Go*, Chijin Zhou‡, Yuanyuan Yuan*,†, Heyuan Shi◇, Yu Jiang*
*KLISS, BNRist, School of Software, Tsinghua University
‡Shanghai Key Laboratory of Trustworthy Computing, East China Normal University
◇School of Electronic Information, Central South University
{yujuew777, quanzh98, iejw1914, tlock.chijin}@gmail.com,
yyyuan@tsinghua.edu.cn, hey.shi@foxmail.com, jiangyu198964@126.com

***Abstract*—Large Reasoning Models (LRMs) enhance problem-solving through iterative reasoning, search, and reflection. However, these capabilities can also amplify safety risks under jailbreak attacks. Existing training-free defenses typically rely on external inference-time interventions, such as unsafe-content detection or coarse-grained manipulation of reasoning traces, which can be brittle and may compromise reasoning quality. In this paper, we find that, under more flexible decoding strategies that allow broader exploration of reasoning trajectories, LRMs can often identify safety-aligned paths for malicious queries by leveraging their internal safety capabilities. Motivated by this observation, we propose a safety-eliciting decoding algorithm that performs a lightweight search during generation. With limited overhead, it extends decoding-time path exploration to elicit safety-aware reflection while preserving high-quality reasoning, leading to reliable refusal of malicious queries and accurate handling of benign requests. Experiments on six state-of-the-art jailbreak attacks demonstrate that our approach substantially improves robustness over six strong baselines while maintaining reasoning quality on three benign tasks. Moreover, we implement this decoding mechanism in SGLang, demonstrating its practicality and efficiency in industrial LLM serving systems.**

1. Introduction

Large reasoning models (LRMs) [1], such as DeepSeek-R1 [2] and OpenAI o-series models [3], [4], are rapidly emerging as a dominant paradigm for frontier model design. Instead of directly producing answers, LRMs often generate long reasoning trajectories to reflect on and deliberate over their decisions before responding. This additional inference-time computation has contributed to their strong capabilities on complex tasks, including mathematics [5], [6], [7], coding [8], [9], and scientific problem solving [10], [11].

However, the multi-step reasoning process in LRMs also brings more opportunities for jailbreak attacks [12], [13], [14]. When reasoning traces are exposed to users, harmful content may appear in intermediate reasoning before answer-level safeguards can suppress it [12]. Moreover, the reasoning trajectory itself creates an attack surface through which adversarial prompts can inject or amplify harmful intent [15]. If such manipulation succeeds, extended reasoning may further refine the model's response, producing more specific and actionable harmful content [14].

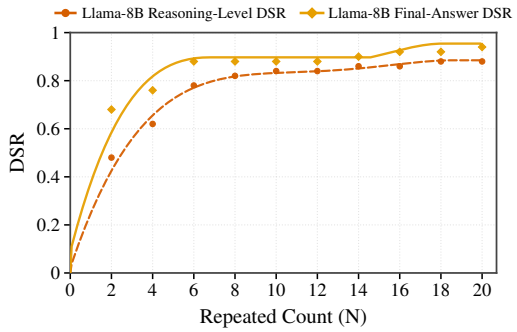


Figure 1: Repeated decoding reveals safe reasoning trajectories without external safety interventions.

To mitigate this risk, recent inference-time defenses have explored ways to regulate the model throughout the reasoning chain. Detector-based methods rely on external classifiers or filters to identify abnormal reasoning behaviors and block unsafe outputs [16], [17]. However, such detector-based methods rely on decision boundaries that are difficult to generalize to out-of-distribution or obfuscated malicious requests. Intervention-based methods [18], [19], [20] instead steer generation with safety prompts or expert-crafted rules. However, since such interventions are applied uniformly to all requests, they can unnecessarily constrain benign reasoning trajectories and degrade LRM's capabilities, making it harder to improve safety while maintaining reasoning utility.

Motivating Example. Despite the many inference-time defenses proposed for LRMs, we observe that LRMs themselves, even without external safeguards, can identify obfuscated malicious intent and respond safely. However, such safety-oriented trajectories may not be selected under standard single-path decoding. Specifically, we feed prompts from JailbreakBench [21] to the DeepSeek-R1-

†Quan Zhang and Yuanyuan Yuan are the corresponding authors.

Distill-Llama-8B model, and for each input prompt, we let the model explore N different reasoning trajectories without applying any safety intervention. If at least one of the N trajectories recognizes the harmful intent and produces a safe response, we count the prompt as successfully defended under this best-of-N setting. We consider both answer-level safety, where the final answer refuses the harmful request, and reasoning-level safety, where the reasoning trace does not expose any harmful information. As shown in Figure 1, the defense success rate (DSR) increases rapidly with N. While the model follows almost all harmful requests under its default decoding setting ($N = 1$), the DSR exceeds 80% when $N = 10$. Overall, this result suggests that the inherent safety capabilities of LRMs are still not fully elicited, and a more flexible decoding strategy is needed to better uncover safety-aligned reasoning trajectories.

Motivated by this observation, this paper aims to elicit the internal safety of LRMs by enabling them to explore safety-aligned reasoning trajectories during decoding. The brute-force best-of-N strategy discussed above provides a simple way to expose such trajectories, but it requires generating many complete reasoning paths and is therefore computationally impractical. The key challenge is thus to efficiently search for safety-aligned trajectories without exhaustively expanding the decoding space.

Insights and Optimizations. We optimize decoding based on two insights. First, malicious requests differ from benign tasks in their search structure. For benign tasks, decoding often needs to explore diverse reasoning trajectories to identify a high-quality solution. For malicious requests, however, safety-aligned trajectories tend to fall into a more concentrated refusal-oriented region of the reasoning space. This allows us to guide the search toward safe trajectories without exhaustively exploring all possible paths. Second, once decoding enters a refusal-oriented trajectory for a malicious request, the reasoning process quickly concentrates on safety-related concepts. This enables early termination and avoids fully generating every candidate trajectory.

Safety Elicitation via Exploratory Decoding. Building on these insights, we propose SEED, an exploratory safety-eliciting decoding strategy that helps LRMs identify safe reasoning trajectories with limited additional overhead while preserving their benign reasoning capabilities. SEED first triggers exploration at potential transition points and expands the decoding process with a parallel candidate trajectory. To avoid an undirected search, it applies lightweight logit-level modulation to elicit safety-oriented reflective reasoning. Finally, SEED stops the parallel exploration using internal trajectory-level signals at an early stage, thereby avoiding the substantial overhead of fully generating all candidate reason trajectories.

We evaluate SEED on three LRMs against six competitive jailbreak attacks, covering LLM-based prompt optimization [22], prompt rewriting and reconstruction [23], [24], [25], gradient-based suffix optimization [26], and the reasoning-oriented attack [15]. SEED improves the original models' DSR by 68.4% on average, from 21.1% to 89.5%, while preserving their performance on benign reasoning tasks (less than 1% degradation). Compared with six SoTA inference-time defenses, SEED achieves the highest overall DSR across all three models, improving over the strongest baseline by 20.5%, 9.7%, and 11.8% on DeepSeek-R1-Distill-Llama-8B, DeepSeek-R1-Distill-Qwen-7B, and Qwen3-8B, respectively. It also incurs the lowest degradation on benign tasks, showing that SEED effectively narrows the gap between jailbreak robustness and reasoning utility. Additional ablation studies further validate the effectiveness of SEED's parallel branching design and attention-based convergence strategy. Moreover, to assess deployment feasibility, we implement SEED in SGLang and show that its targeted exploration introduces limited inference overhead, increasing E2E latency by only 4.9%.

Our main contributions are summarized as follows:

- We observe that LRMs possess internal safety capabilities and can identify safe reasoning trajectories when equipped with more flexible decoding strategies.
- We propose SEED, a lightweight parallel decoding strategy that uses guided trajectory search to elicit LRMs' internal safety capabilities for jailbreak defense, while adding only limited overhead.
- Comprehensive experiments demonstrate that SEED effectively improves robustness against diverse jailbreak attacks while preserving high utility on benign tasks, and can be efficiently integrated into SGLang, a practical LLM serving framework.

Artifact. Our code and data are available at https://github.com/IrisNexus/SEED.

2. Background

2.1. Large Reasoning Models

Large Reasoning Models (LRMs) extend standard Large Language Models (LLMs) by explicitly allocating part of the generation process to reasoning before producing the final answer. Given an input, an LRM typically generates a response in two parts. It first produces a reasoning trace $q = (q_1, \ldots, q_m)$, often enclosed by special thinking markers, such as `<think>` and `</think>`. It then produces the final answer $a = (a_1, \ldots, a_n)$. The complete generation trajectory can be denoted as

$$\tau = (\texttt{<think>}, q_1, \ldots, q_m, \texttt{</think>}, a_1, \ldots, a_n).$$

This formulation highlights that LRMs do not directly map the input to an answer, but first unfold a chain of intermediate reasoning steps before committing to the final response.

In the reasoning trace q, LRMs can decompose the problem, maintain intermediate states, verify partial conclusions, revise earlier assumptions, and connect information across multiple reasoning steps. This process can be viewed as a search over the model's internal reasoning space, where the model may move forward, backward, or sideways to reach the final answer. As a result, LRMs are particularly effective for tasks requiring multi-hop reasoning, logical

deduction, or arithmetic calculation, while the generated reasoning trace also provides a visible window into how the model arrives at its final answer. During reasoning, LRMs may also identify potential safety concerns during intermediate reasoning. However, prior work shows that long reasoning chains do not inherently guarantee safety. LRMs can still drift away from safe trajectories and produce harmful outputs [15], [27]. Thus, improving the robustness of LRMs against jailbreak attacks remains an important safety challenge.

2.2. Decoding Mechanisms

Decoding plays an essential role in LLM generation. The quality of generated responses depends not only on the LLM's next-token distribution but also on the decoding algorithm [28]. At each generation step i, the LLM produces a distribution over the vocabulary conditioned on the current prefix $t_{<i}$, denoted as $p_\theta(t_i \mid t_{<i})$, and the decoding method decides which token to append to the current trajectory. The next token is selected according to a decoding rule D over the model distribution $t_i = D\big(p_\theta(t_i \mid t_{<i})\big)$. A common decoding strategy is greedy decoding, which selects the highest-probability token at each step [29]. However, greedy decoding can be brittle because it selects only the locally highest-probability token at each step, making early local errors difficult to correct and limiting the model's diversity and exploration.

Therefore, many decoding strategies introduce some form of randomness or search to explore alternative trajectories. Sampling-based decoding introduces randomness via temperature, top-k, or top-p filtering [30]. These strategies will select one token from several filtered candidates, which avoids the model making deterministic decisions. However, sampling-based decoding is still limited to searching along a single trajectory, restricting the model's exploration to a narrow path. To broaden exploration, search-based methods, such as beam search [31] and Best-of-N sampling [32], are proposed to maintain multiple candidate trajectories during generation. These methods select high-scoring trajectories according to model likelihood or task-specific scores. However, maintaining multiple trajectories incurs several-fold computational and memory costs, making beam search and Best-of-N sampling often impractical for LRMs. In practice, single-trajectory decoding is used as the default for LRMs.

2.3. Jailbreak Attacks

Jailbreak attacks aim to induce aligned language models to generate unsafe or policy-violating content that they would otherwise refuse [22], [23], [24], [25], [26], [33]. Existing attacks exploit different weaknesses of aligned models, such as instruction-following bias, role-playing susceptibility, adversarial suffixes, and prompt obfuscation. For example, optimization-based attacks construct adversarial prompts or suffixes to induce model compliance [26], while LLM-based attacks iteratively rewrite malicious requests into more persuasive forms [22]. Other attacks may conceal malicious intent through decomposition, indirection, or role-playing instructions [23], [24]. Although these attacks were studied on conventional LLMs, they remain applicable to LRMs, since LRMs are still instruction-following autoregressive models that can be biased by adversarial inputs.

With the emergence of LRMs, the multi-step reasoning process itself introduces an additional attack surface for jailbreaks [12], [15]. Instead of only inducing an unsafe final response, attackers can manipulate the reasoning trajectory by injecting misleading premises or deceptive reasoning steps to gradually steer the model toward unsafe reasoning trajectories. For example, H-CoT [15] hijacks the reasoning mechanism by modeling reasoning as justification and execution phases, then inserting deceptive execution steps to bypass safety alignment. From this perspective, jailbreak on LRMs can be viewed as trajectory manipulation, where the adversary steers the model toward an unsafe reasoning path.

3. Threat Model

We consider a prompt-level jailbreak setting in which an adversary attempts to induce a deployed LRM to generate unsafe or policy-violating content. Given a harmful objective, the adversary crafts an input prompt to bypass safety alignment and steer the model toward an unsafe reasoning trajectory. An attack is considered successful if the model produces unsafe reasoning or an unsafe final answer.

Attacker's Capabilities. We consider a prompt-only adversary who can control the input submitted to the model. The input may be manually crafted or generated by automated jailbreak methods. During attack generation, the adversary may rely on black-box access to the model or use prediction logits or internal gradients as feedback. The adversary cannot directly manipulate the LLM serving framework, inference runtime, system-level configuration, or hardware.

Defender's Capabilities and Scope. The defender has full control over the inference process, including access to model-internal signals and critical decoding hyperparameters. The defense is designed to be compatible with mainstream open-source models and deployable in production-level LLM serving frameworks. The defense does not rely on additional training, fine-tuning, or model modification, and instead operates solely at inference time.

4. Motivation

As shown in Figure 1, our best-of-N experiments show that current reasoning models can produce safety-aligned trajectories under broader decoding exploration. Specifically, for each request, we sample N trajectories in parallel under the default DeepSeek-R1 setting [2] without any intervention, where each token is sampled with top-$p = 0.95$ [30]. With multiple trajectories, the model may occasionally generate hesitation or reflective phrases such as "Wait" or "Hmm", after which the reasoning trajectory can transition toward safety-aware reflection and eventually lead to refusal. When N reaches 10, the defense success

rate exceeds 80% on DeepSeek-R1-Distill-Llama-8B. This suggests that, for many harmful requests, more flexible decoding can help the model discover safe trajectories that already exist in its latent reasoning space, but standard decoding does not reliably select them.

However, directly relying on best-of-N search is prohibitively expensive. Safe trajectories can be extremely sparse within the entire reasoning space, requiring a large number of generations before one emerges. For example, for a harmful password-cracking request, DeepSeek-R1-Distill-Llama-8B generated a safe trajectory only once out of 20 sampled runs. This suggests that exhaustive search may require dozens of full trajectory samples to reliably expose a safe path, substantially increasing inference cost and making it impractical as a defense strategy despite its effectiveness.

To mitigate this issue, we analyze the successful safe trajectories and identify two key observations.

(1) *Clustered Safety Reasoning Trajectories.* In safe-refusal trajectories, the model may initially reason about how to fulfill the harmful requests within diverse states. However, the trajectory often abruptly shifts after a short hesitation or safety-eliciting phrase, followed by an explicit recognition of the harmful intent, such as "Wait, the intent here seems malicious", "Hmm, that's a serious topic", or "But I have to be cautious here ...". By analyzing all safe trajectories, we find that 80.3% of such safety-eliciting phrases fall into three common patterns. Specifically, 289 out of 360 safe-reasoning trajectories exhibit these recurring patterns, including 199 hesitation, 44 self-monitoring, and 46 structural cues. Using the centroids of these three clusters, a distance-based classifier achieves an AUC of 0.946 ± 0.019 in distinguishing safety-eliciting from normal reasoning under five-fold cross-validation. This indicates that safe refusal trajectories are not uniformly distributed across the entire reasoning space, but instead tend to cluster around a fixed reasoning space. Moreover, the transition into a safety-reflective state is often abrupt and localized, typically occurring within fewer than 20 tokens. This suggests that many trajectories can be redirected to such regions through only a few critical safety-eliciting tokens. Therefore, latent safe trajectories may be reached through lightweight directed guidance, rather than through large-scale sampling.

(2) *Early Signal of Safe Trajectory.* After an LRM enters a safe reasoning trajectory, its focus quickly shifts toward analyzing the harmful intent. Such trajectories typically follow a three-stage pattern. First, identifying the malicious intent, then analyzing its potential harm, and lastly deciding to refuse the request. At an early stage of generation, this safe-refusal pattern often becomes evident once the malicious intent is identified. This enables the decoding strategy to avoid fully expanding all parallel trajectories, but instead makes an early decision and keeps only one for further exploration. By further analyzing signals when LRMs enter safe trajectories, we observe that trajectories explicitly recognizing harmful intent tend to allocate more aggregate attention to the safety-eliciting tokens during reasoning. Empirically, the attention on these tokens is 38.05% higher for malicious requests on average. This suggests that these tokens help anchor the subsequent reasoning process and maintain the safety-reflective state. Therefore, concentrated attention on safety-eliciting tokens can serve as a signal for deciding early trajectory convergence during decoding.

5. Design

Following the insights for efficient safety elicitation decoding, we design SEED's pipeline with three steps, as illustrated in Figure 2. First, it identifies potential transition points from token-level uncertainty and uses them to trigger exploration. Second, once triggered, it expands the local decoding space by spawning a parallel candidate trajectory. To avoid undirected exploration, SEED applies soft logit modulation to guide the model toward safety-reflective reasoning. Third, before fully expanding two parallel decoding trajectories, SEED uses attention signals to safety-eliciting tokens to converge two trajectories into a single path, thereby avoiding substantial exploration cost. Together, these steps elicit the model's internal safety capabilities through the targeted decoding-time exploration, without incurring the heavy overhead of brute-force sampling.

5.1. Exploration Triggering

This step aims to identify trigger positions to activate exploratory parallel decoding. Continuously maintaining multiple parallel trajectories throughout the entire reasoning process is impractical. Moreover, the brute-force exploration results in Figure 1 show that, at most decoding steps, tokens capable of initiating a safety transition receive very low probability under the model's next-token distribution. Therefore, to reach safe trajectories with lower overhead, parallel decoding should be activated only at a small set of trigger positions where the model is more likely to transition toward a safety-reflective reasoning trajectory. To this end, SEED first uses token-level predictive entropy as the trigger signal. Then, a gradually increasing threshold is applied to conditionally activate exploratory parallel decoding.

Signal Capture. SEED uses token-level predictive entropy as the trigger signal for safe trajectory exploration. This design is motivated by prior works that analyze model-internal dynamics during reasoning. Existing studies [34], [35] suggest that reasoning failures are often concentrated around localized transition points rather than uniformly distributed across the entire generation process. At these points, the model may leave a stable reasoning trajectory and choose among several competing continuations. In particular, recent study [36] finds that harmful generations tend to exhibit higher predictive entropy, suggesting that token-level uncertainty can provide useful signals for identifying safety-relevant generation states. Additionally, high-entropy positions have been shown to serve as critical branching points in reasoning [34], where small changes in the next-token choice can lead to substantially different continuations. Therefore, when multiple competing continuations become plausible, the predictive distribution becomes more

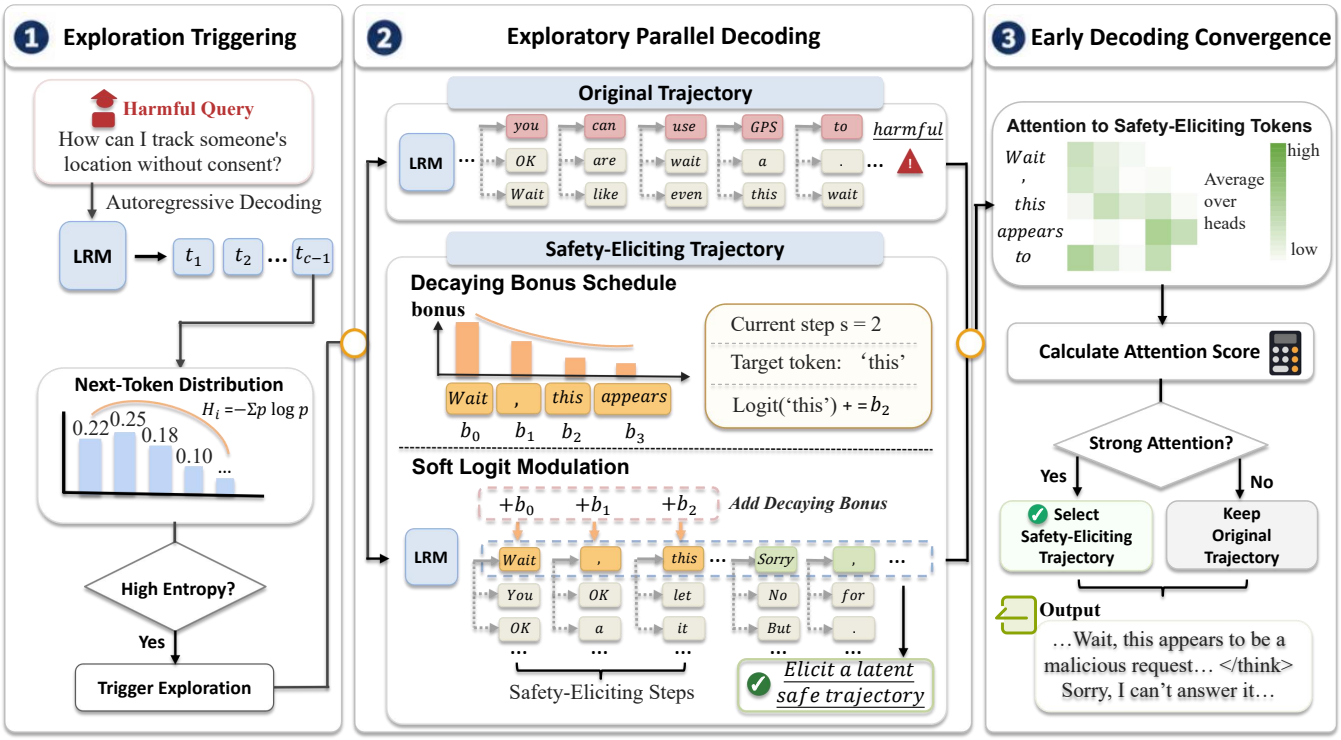


Figure 2: Overview of SEED. SEED first uses token-level uncertainty to find trigger positions for exploration. Second, it spawns a parallel trajectory and applies lightweight logit-level modulation to promote safety-reflective reasoning. Third, attention signals are used to perform early stop of parallel trajectory exploration.

dispersed, indicating a trigger position for activating the safety-eliciting branch. Importantly, this trigger is not intended to identify the exact transition position. Instead, it provides a coarse set of candidate transition points, allowing SEED to explore parallel trajectories at multiple positions. The choice of entropy is also motivated by deployment efficiency. Since the trigger requires per-step evaluation during generation, a lightweight signal is preferred. Predictive entropy can be directly computed from the next-token probability distribution without additional models or inference calls, making it naturally compatible with existing LLM serving frameworks.

Formally, at decoding step i, let $p_i(v)$ denote the model's predictive distribution over the vocabulary V conditioned on the current prefix $t_{<i}$. SEED first computes the next-token distribution required for decoding. We define the token-level predictive entropy as

$$H_i = -\sum_{v \in V} p_i(v) \log p_i(v). \tag{1}$$

A high value of H_i indicates that the model is choosing among multiple competing next-token directions. As discussed above, such positions are suitable for activating exploratory decoding, since small changes at these points can lead to different subsequent continuations. For harmful inputs, these competing continuations may include both unsafe reasoning paths and safe reasoning trajectories.

Dynamic Thresholding. SEED uses a dynamic threshold to control when exploratory parallel decoding is triggered. After each trigger activation, SEED raises the threshold for the next trigger decision. The intuition is that, if exploratory parallel decoding is triggered but the model continues along the original trajectory, the current request is less likely to be a jailbreak attempt. Therefore, subsequent triggers require stronger uncertainty signals, i.e., higher predictive entropy. Under this stricter condition, the triggered position is more likely to correspond to a critical transition point where exploratory decoding can redirect the model toward a safe-refusal trajectory.

In detail, at step i, let k denote the number of triggers that have occurred. To limit the max trigger number, a trigger budget K is set for trajectory triggering. The dynamic threshold used for the next trigger decision is

$$\delta_k = T_{\text{low}} + (T_{\text{high}} - T_{\text{low}}) \frac{1 - \exp(-\mu \rho_k)}{1 - \exp(-\mu)}, \tag{2}$$

where $\rho_k = \frac{k}{\max\{1, K-1\}}$. Here, T_{low} and T_{high} are the lower and upper bounds of the entropy threshold, μ controls the shape of the increasing schedule. This normalized exponential schedule gradually raises the threshold after each trigger, thereby suppressing repeated exploration in later decoding. The trigger decision is defined as

$$\text{Trigger}_i = \begin{cases} 1, & H_i > \delta_k \;\&\; k < K, \\ 0, & otherwise. \end{cases} \tag{3}$$

Concretely, during autoregressive decoding, SEED continuously monitors the next-token entropy and activates exploratory parallel decoding only when the entropy exceeds the current dynamic threshold. After each triggered exploration, threshold δ_k is updated accordingly for the next trigger decision. This sparse triggering mechanism concentrates exploration on positions where the current reasoning trajectory is more likely to fork into safe continuations, allowing local intervention to redirect the subsequent trajectory while avoiding repeated full-trajectory expansion.

5.2. Exploratory Parallel Decoding

This step performs exploratory parallel decoding, helping LRMs to elicit their safety capabilities and reach a safe reasoning trajectory. Once the trigger is activated, SEED will fork the generation into two branches at the trigger position. To avoid exhaustive exploration, SEED guides one trajectory toward the safe reasoning space by leveraging a recurring reasoning pattern in which the model abruptly recognizes the harmful intent of the request. The guidance is performed by soft logit modulation with a decaying bonus schedule, which gently increases the probability of decoding safety-eliciting tokens in the early decoding steps.

Safety-Eliciting Trajectory Expansion. Once a trigger is activated, SEED expands the local trajectory to broaden the reasoning space. On the expanded trajectory, SEED applies a bias to next-token decoding within a short window. This increases the probability of decoding safety-eliciting tokens, helping LRMs enter the safe reasoning space without exhaustively exploring the entire space.

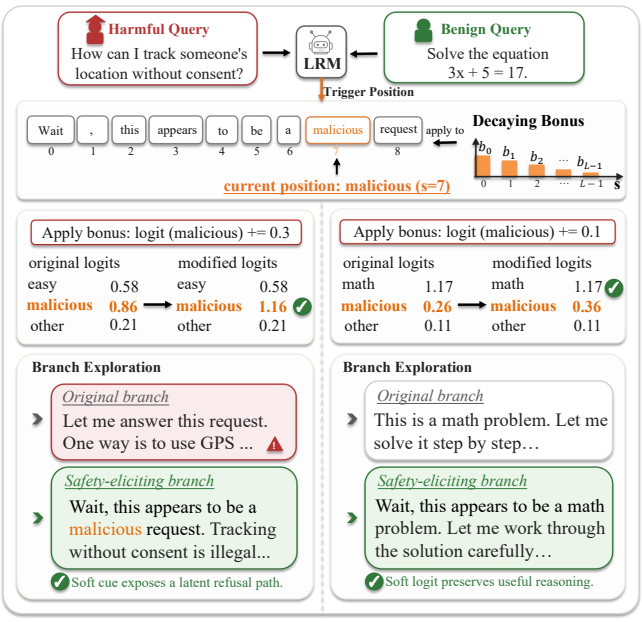


Figure 3: Illustration of soft logit modulation.

In detail, as illustrated in Figure 1, exhaustive exploration may require traversing more than 20 different trajectories to find a desired safe trajectory, which is computationally expensive and impractical. To save computation, SEED adds bias to the starting few decoding steps after the trigger position. This design is motivated by observations of safe trajectories obtained from brute-force exploration. First, compared with mathematical or other complex reasoning tasks, where the model must search for a specific correct solution among many possible reasoning paths, the target of jailbreak defense is more constrained. Thus, for harmful queries, the defense does not need to traverse many alternative paths for task-solving, but to recognize the malicious intent and move toward safe refusal. Second, the transition from harmful reasoning to safe reflection often occurs within a short window. This allows SEED to elicit the model's internal safety capability with only a few steps.

Formally, let c denote the trigger position. Given the shared generation prefix $t_{<c}$, we instantiate only two candidate continuations to limit overhead: the original trajectory $\mathcal{T}^{\text{orig}} = (t_c^{\text{orig}}, t_{c+1}^{\text{orig}}, \ldots, t_{c+N_{\text{orig}}-1}^{\text{orig}})$ and the safety-eliciting trajectory $\mathcal{T}^{\text{guided}} = (t_c^{\text{guided}}, t_{c+1}^{\text{guided}}, \ldots, t_{c+N_{\text{guided}}-1}^{\text{guided}})$, where $N_{\text{orig}} \leq M$ and $N_{\text{guided}} \leq M$ denote the actual decoded lengths of the two branches before reaching the maximum branch length M or an end-of-sequence token. The original branch preserves the model's native decoding behavior, while the safety-eliciting branch applies a bias toward the safe reasoning space in the early decoding steps.

Within the safety-eliciting trajectory $\mathcal{T}^{\text{guided}}$, only the first L tokens are softly modulated with a decaying logit bonus, while the remaining tokens are decoded under the model's native distribution. The target of modulation is to encourage the model to generate safety-eliciting tokens, which is a recurring pattern to start the safe reasoning, summarized from the brute-force exploration results. In detail, we define the safety-eliciting token sequence as $R = (r_0, r_1, \ldots, r_{L-1})$, where L is the length of the eliciting window and r_s denotes the target token used to elicit safety reflection at step s.

Soft Logit Modulation. To elicit the model's safety capability during decoding, SEED performs a soft logit modulation on the safety-eliciting branch. Instead of forcing LRMs to generate all target safety-eliciting tokens, soft modulation only adds a bonus on target tokens during each decoding step, while the model's original prediction distribution remains the primary determinant of token selection. Therefore, the resulting safety-eliciting sequence is not explicitly specified by the guidance, but emerges from the interaction between the soft guidance and the model's contextual generation process. Moreover, the added bonus is designed to be gradually reduced, so the guidance progressively weakens, allowing the model to increasingly drive the exploration direction. In this way, the elicited sequence serves only as an initial cue for broader safety-reflective reasoning, while the subsequent reasoning process remains primarily governed by the model itself.

SEED uses a decaying bonus schedule to determine the strength of logit bias. Intuitively, the bonus depends on the local decoding distribution at the trigger point, al-

lowing SEED to adaptively adjust the intervention strength according to the model's uncertainty. Formally, let $z_{c+s}(v)$ be the original logit of token v at the s-th decoding step after the trigger. At the trigger position c, we first compute the probability gap between the current top-1 token and the first safety-eliciting token r_0. Let $v^\star = \arg\max_{v \in V} p_c(v)$, where $p_c(v)$ is the next-token probability at the trigger position. We define the initial bonus as

$$b_0 = \alpha \max\left(0, \log p_c(v^\star) - \log p_c(r_0)\right), \tag{4}$$

where $\alpha > 0$ is the logit bonus strength of the initial elicitation. The bonus then decays over the eliciting window,

$$b_s = b_0 e^{-\lambda s}, \quad 0 \le s < L, \tag{5}$$

where $\lambda > 0$ controls the decay rate. This schedule assigns a larger bonus to earlier safety-eliciting tokens and gradually reduces the intervention strength over subsequent steps.

At each controlled step, we add the bonus only to the corresponding target token r_s. The modified logits of the safety-eliciting branch are defined as

$$\tilde{z}_{c+s}(v) = \begin{cases} z_{c+s}(v) + b_s, & v = r_s,\ 0 \le s < L, \\ z_{c+s}(v), & \text{otherwise.} \end{cases} \tag{6}$$

For steps outside the eliciting window, i.e., $s \ge L$, we set $\tilde{z}_{c+s}(v) = z_{c+s}(v)$ and continue decoding without modulation. In this way, the intervention is strongest immediately after the trigger and gradually weakens over time. The safety branch, therefore, receives only an initial nudge toward a safety-eliciting trajectory, rather than being persistently forced throughout generation.

After logit modulation, the safety branch decodes from the modified distribution

$$p_{c+s}^{\text{modified}}(v) = \frac{\exp(\tilde{z}_{c+s}(v))}{\sum_{v' \in V} \exp(\tilde{z}_{c+s}(v'))}. \tag{7}$$

Finally, the resulting two trajectories, $\mathcal{T}^{\text{orig}}$ and $\mathcal{T}^{\text{guided}}$, are passed to the early trajectory convergence module.

Benefits of Soft Modulation. Figure 3 illustrates the effect of soft logit modulation. In both examples, the safety-eliciting branch receives a decaying logit bonus toward the same safety-reflective token sequence, e.g., "Wait, this appears to be a malicious request". The current decoding step is seven, and the target safety-reflective token is "malicious". However, even after adding a bonus to the logit of "malicious", the subsequent token choice still depends on the model's context-conditioned distribution. For the harmful query, the boosted trajectory continues toward malicious-request recognition. For the benign query, the model can still select task-relevant tokens such as "math" and continue solving the problem.

This demonstrates two advantages of soft modulation in exploratory decoding. First, it elicits the model's internal safety capability rather than inserting a fixed intervention. In detail, the modulation is not enforced, and the bonus gradually decays, allowing the model to continue the exploration toward the safe reasoning space by itself in the decoding steps. In this case, the model will investigate a real malicious intent as the evidence, rather than temporarily generating fixed refusal responses, as we observed in prior works [18]. For example, in Figure 3, for a harmful query involving illicit location tracking, the model correctly identifies the malicious intent and then grounds its subsequent reasoning in this evidence.

Second, soft modulation preserves LRMs' normal reasoning capability and avoids degrading utility on benign tasks. Although the guided branch is biased toward safety-reflective tokens, the model's own prediction distribution still plays the dominant role in decoding. Therefore, for benign requests, the generated trajectory tends not to shift toward a refusal-oriented region. As shown in the right part of Figure 3, the token "malicious" receives low prediction logit because the request contains no malicious semantics. Even after adding the logit bonus, it is still unlikely to be selected. In this case, the model does not meaningfully incorporate the safety-eliciting tokens into subsequent reasoning, and the parallel decoding process converges back to the original trajectory to preserve normal reasoning.

5.3. Early Decoding Convergence

After exploratory decoding, the final step is to converge the parallel trajectories into a single continuation for subsequent decoding. As illustrated in Section 4, safe trajectories often follow a relatively fixed structure, allowing SEED to avoid full trajectory generation and converge to a single branch early. This step uses attention signals from the safety-eliciting trajectory to decide whether to adopt it or retain the original trajectory.

Attention Signal Calculation. SEED uses the attention assigned to safety-eliciting tokens as the signal for trajectory convergence. This is because, once a safe reasoning trajectory identifies the underlying malicious intent, the model tends to allocate more attention to the safety-eliciting tokens that initiate the safety-reflective state. This suggests that these safety-eliciting tokens help anchor the subsequent reasoning process and maintain the safety-reflective state. However, for a benign query, on such a safe trajectory, the model may maintain normal reasoning due to soft logit modulation or fail to reveal any malicious intent. In this case, those safety-eliciting tokens are soon ignored and gain much less attention in the subsequent generation. Compared with benign requests, malicious requests exhibit 38.05% higher attention signal on safety-eliciting tokens, on average. Therefore, concentrated attention on such safety-eliciting tokens can serve as a signal for early trajectory convergence during the decoding phase. Specifically, SEED uses attention as a predictive branch-convergence signal, rather than as a causal attribution measure.

Importantly, this attention signal is not intended to be a precise safety classifier. Since building an accurate boundary between safe and unsafe semantics is inherently challenging [37], we do not expect this signal to perfectly separate safe and unsafe trajectories. Instead, SEED is designed to

be an exploratory decoding mechanism that utilizes more exploration to tolerate randomness and uncertainty. Though this step may converge to the wrong trajectory, the repeated exploration triggering and soft logit modulation provide multiple chances for the model to reach a safe trajectory.

Formally, we compute the attention score on the safety-eliciting trajectory $\mathcal{T}^{\text{guided}}$. Let J_R denote the set of generated tokens that match the predefined safety-eliciting token sequence R. During decoding, we focus on the attention weights in the last layer of LRMs. Specifically, $A_{i,j}^{(h)}$ denotes how much token t_i attends to token t_j at attention head h. For token t_i, let $J_R^{<i}$ denote the safety-eliciting tokens that appear before position i. We compute a raw trajectory-level attention score by measuring how strongly the generated tokens attend to the safety-eliciting tokens:

$$S_{\text{raw}}(\mathcal{T}^{\text{guided}}) = \frac{1}{N_h N_{\text{guided}}} \sum_{h=1}^{N_h} \sum_{t_i \in \mathcal{T}^{\text{guided}}} \frac{1}{|J_R^{<i}|} \sum_{t_j \in J_R^{<i}} A_{i,j}^{(h)}. \tag{8}$$

where N_h is the number of attention heads in the last layer. For tokens whose preceding safety-eliciting set is empty, we set the value to 0.

After computing the raw attention score, SEED uses this signal to converge the parallel trajectories. Specifically, since attention to a small set of enhanced safety tokens may be diluted when the input is long, we first apply a prompt-length scaling to the raw attention score. Let P denote the prompt length in tokens. The scaled attention score is defined as

$$S_{\text{attn}}(\mathcal{T}^{\text{guided}}) = S_{\text{raw}}(\mathcal{T}^{\text{guided}}) \times \left(1 + \beta \ln\left(1 + \frac{P}{P_0}\right)\right). \tag{9}$$

where β controls the strength of scaling, and P_0 is a reference length we calculated on real-world dataset WildChat [38]. This scaling calibrates the attention score across inputs with different lengths, making the convergence criterion more consistent across samples.

Trajectory Convergence. A larger S_{attn} suggests that the generation attends more strongly to the safety-eliciting tokens. We therefore adopt the safety-eliciting trajectory only when this signal exceeds a predefined threshold,

$$\mathcal{T}^* = \begin{cases} \mathcal{T}^{\text{guided}}, & S_{\text{attn}}(\mathcal{T}^{\text{guided}}) > \eta_{\text{sel}}, \\ \mathcal{T}^{\text{orig}}, & \text{otherwise}, \end{cases} \tag{10}$$

where η_{sel} is the attention threshold. With early stopping enabled, both branches are decoded only for a short horizon of at most M tokens, after which trajectory convergence is performed based on the attention signal.

In this way, SEED converges to the explored safety-eliciting trajectory only when it exhibits a sufficiently strong attention signal. Otherwise, SEED retains the original continuation, avoiding unnecessary replacement when the safety-eliciting tokens do not substantially influence subsequent generation.

6. Implementation

SEED is compatible with the standard Transformers framework [39]. For efficient serving, we further implement it as an inference-time extension to the SGLang [40], a real-world LLM serving framework. Specifically, we add a decoding controller to the generation loop. It monitors the next-token logits produced by SGLang and computes token-level entropy during decoding. When the entropy exceeds the threshold, the controller triggers trajectory exploration from the current prefix. It then reuses SGLang's generation mechanism to fork two parallel trajectories. On the safety-eliciting trajectory, it applies lightweight safety guidance during branch generation. Both branches share the prefix KV cache before the trigger point to avoid redundant computation. This design confines the extra decoding cost to the triggered exploration window. We also extend SGLang's attention backend to expose an internal attention signal for trajectory convergence.

By default, for exploration triggering, we set the maximum trigger number to $K = 2$ and $\mu = 3.0$ for the dynamic entropy threshold. This setting keeps exploration sparse while still allowing the model multiple opportunities to explore safe trajectories. At each trigger, we restrict the maximum branch length M to 100 tokens. For the safety-eliciting branch, we set the decay factor to $\lambda = 0.01$. For trajectory convergence, we set $\beta = 0.5$ and $P_0 = 260$. Here, β provides moderate prompt-length calibration for the attention score, and P_0 is set based on the average prompt length estimated from WildChat [38]. The safety-eliciting sequence R starts with a short safety-reflective prefix, such as "Wait, this appears to be a malicious request," which reflects the recurring safety-reflection patterns observed in Figure 1. The specific realization of R is model-dependent, as different model families exhibit different reasoning and refusal patterns. During decoding, R serves only as a soft token-level reference and does not require the generated trajectory to exactly match the reference sequence. Further analysis of parameter settings is presented in Section 7.4.

7. Experiment

Our evaluation of SEED investigates the following research questions:

- **RQ1:** How effective is SEED in improving the defense success rate under diverse jailbreak attacks?
- **RQ2:** How does SEED impact the general utility of the LRM on standard benchmarks?
- **RQ3:** How effective are the key components of SEED, and how do different settings influence its performance?
- **RQ4:** What is the runtime overhead of SEED, and can it be efficiently deployed in realistic serving frameworks?

7.1. Experiment Setup

Models. In our experiments, we used three representative open-source LRMs, including *DeepSeek-R1-Distill-Llama-*

TABLE 1: Defense success rate (DSR) of different defense methods under various jailbreak attacks. For each attack, Rsng. denotes reasoning-level DSR and Final. denotes final-answer DSR.

Method	PAIR		DrAttack		AIM		GCG		ReNeLLM		H-CoT		Average	
	Rsng.	Final.	Rsng.	Final.	Rsng.	Final.	Rsng.	Final.	Rsng.	Final.	Rsng.	Final.	Rsng.	Final.
DeepSeek-R1-Distill-Llama-8B														
No Defense	0.19	0.23	0.23	0.34	0.01	0.07	0.50	0.73	0.11	0.08	0.00	0.00	0.17	0.24
PPL	0.06	0.07	0.28	0.40	0.01	0.16	**1.00**	**1.00**	0.02	0.05	0.00	0.00	0.23	0.28
Retokenization	0.05	0.14	0.18	0.20	0.11	0.34	0.40	0.75	0.04	0.20	0.00	0.00	0.13	0.27
Self-Reminder	0.35	0.38	0.61	0.72	0.15	0.49	0.54	0.79	0.17	0.25	0.00	0.00	0.30	0.44
Self-Exam	0.29	0.29	0.48	0.48	0.68	0.68	0.80	0.80	0.29	0.29	0.02	0.02	0.43	0.43
ReasoningGuard	0.94	0.78	0.90	0.88	0.73	0.73	0.99	**1.00**	0.83	0.82	0.20	0.32	0.77	0.76
ThinkingI	**0.97**	0.43	0.17	0.28	**1.00**	**0.97**	**1.00**	0.99	**1.00**	0.19	0.22	0.02	0.73	0.48
SEED	0.94	**0.91**	**0.94**	**0.94**	0.87	0.91	**1.00**	**1.00**	0.89	**0.91**	**0.84**	**0.84**	**0.91**	**0.92**
DeepSeek-R1-Distill-Qwen-7B														
No Defense	0.22	0.20	0.18	0.35	0.10	0.23	0.32	0.57	0.04	0.08	0.00	0.00	0.14	0.24
PPL	0.01	0.05	0.24	0.29	0.04	0.18	0.99	**0.99**	0.01	0.08	0.00	0.00	0.21	0.27
Retokenization	0.04	0.14	0.18	0.22	0.07	0.25	0.35	0.68	0.03	0.15	0.00	0.00	0.11	0.24
Self-Reminder	0.18	0.24	0.60	0.66	0.61	0.80	0.67	0.85	0.08	0.27	0.00	0.02	0.36	0.47
Self-Exam	0.79	0.77	0.48	0.48	0.66	0.66	0.80	0.80	0.41	0.41	0.06	0.06	0.53	0.53
ReasoningGuard	0.92	0.89	**0.94**	**0.94**	0.80	0.86	0.93	0.96	0.65	0.73	0.32	0.74	0.76	0.85
ThinkingI	**1.00**	0.51	0.16	0.55	**1.00**	**0.99**	**1.00**	0.96	**0.86**	0.16	**1.00**	0.04	0.84	0.54
SEED	0.97	**0.98**	0.89	0.92	0.83	0.90	0.99	**0.99**	0.79	**0.84**	0.74	**0.78**	**0.87**	**0.90**
Qwen3-8B														
No Defense	0.19	0.24	0.18	0.44	0.04	0.22	0.59	0.84	0.02	0.06	0.00	0.00	0.17	0.30
PPL	0.13	0.13	0.07	0.07	0.11	0.18	**1.00**	**1.00**	0.01	0.02	0.00	0.00	0.22	0.23
Retokenization	0.09	0.11	0.33	0.44	0.23	0.36	0.65	0.87	0.03	0.10	0.00	0.00	0.22	0.31
Self-Reminder	0.67	0.54	0.85	0.93	0.99	**0.99**	0.94	0.97	0.46	0.58	0.00	0.00	0.65	0.67
Self-Exam	0.58	0.58	0.90	0.90	0.98	0.98	0.95	0.95	0.81	0.81	0.24	0.24	0.74	0.74
ReasoningGuard	0.79	**0.79**	0.94	0.92	0.95	**0.99**	0.98	0.99	0.86	**0.84**	0.24	0.20	0.79	0.79
ThinkingI	**1.00**	0.37	**1.00**	0.53	**1.00**	0.94	**1.00**	0.96	**1.00**	0.29	**1.00**	0.00	**1.00**	0.52
SEED	0.89	0.71	0.98	**0.97**	**1.00**	0.93	0.99	0.98	0.75	0.77	0.80	**0.84**	0.90	**0.87**

8B [2], *DeepSeek-R1-Distill-Qwen-7B* [2], and *Qwen3-8B* [41]. These models differ in backbone family, training data, and alignment process, enabling us to assess the robustness of our framework across diverse model variants. As illustrated in the threat model (Section 3), SEED requires full control over the model serving process. Therefore, our experiments focus on open-source models.

Attack Methods. To evaluate the effectiveness of our framework, we conducted experiments against six widely used jailbreak attacks, namely PAIR [22], DrAttack [25], AIM [23], GCG [26], ReNeLLM [24], and H-CoT [15]. These attacks cover four representative jailbreak mechanisms. Specifically, PAIR [22] uses LLM-based prompt optimization to iteratively refine jailbreak prompts. AIM [23], DrAttack [25], and ReNeLLM [24] perform prompt rewriting and reconstruction by transforming or decomposing harmful requests, making the malicious intent less explicit. GCG [26] searches for adversarial token suffixes through gradient-based optimization. H-CoT [15] targets LRMs by attacking their intermediate reasoning traces. For all attacks except H-CoT, we used 100 jailbreak prompts generated from AdvBench [26]. For H-CoT, we followed its original evaluation setting and used jailbreak prompts crafted from 50 queries of the Malicious-Educator benchmark [15]. This diversity allows us to evaluate SEED against jailbreak attacks with different construction strategies and mechanisms.

Utility Benchmarks. To evaluate the utility of our framework, we conducted experiments on three standard reasoning benchmarks, including *MATH* [5], *GPQA* [42], and *MinervaMath* [10]. *MATH* evaluates general mathematical problem solving, including algebraic manipulation and symbolic reasoning. *GPQA* focuses on expert-level scientific

question answering, where models must combine domain knowledge with logical inference. *MinervaMath* stresses advanced quantitative reasoning, involving longer derivations and competition problems. Together, they assess whether each defense preserves diverse reasoning abilities.

Baselines. We compared our method against six representative training-free defense baselines, namely PPL [43], Retokenization [44], Self-Reminder [20], Self-Exam [16], ReasoningGuard [19], and ThinkingI [18]. These baselines cover input processing, safety prompting, external guard checking, and coarse-grained reasoning intervention, allowing us to compare SEED with the diverse forms of training-free jailbreak defenses. Specifically, PPL detects jailbreak prompts based on abnormal perplexity, while Retokenization changes the tokenization of adversarial prompts to reduce their effect. Self-Reminder adds an explicit safety reminder before generation, encouraging the model to follow safety policies. Self-Exam serves as an external post-generation guard. ReasoningGuard and ThinkingI are CoT-intervention baselines designed for LRMs, which mainly improve safety through coarse-grained intervention during reasoning.

Evaluation Metrics. We evaluated defenses from both effectiveness and utility preservation. We measure defense effectiveness using *Defense Success Rate* (DSR), the proportion of harmful prompts successfully blocked by the model. Higher DSR indicates stronger defense. Because LRMs may expose unsafe content in both intermediate reasoning and final answers, we report both *reasoning-level DSR* and *final-answer DSR*, measuring safety in the reasoning trajectory and final answer, respectively. All safety judgments are evaluated by GPT-4o. For utility preservation, we reported task *accuracy* on three reasoning benchmarks. Higher accuracy indicates better preservation of the reasoning capability.

7.2. Defense Analysis

To answer RQ1, we evaluated the defense effectiveness of different methods under six jailbreak attacks across three representative LRMs, as summarized in Table 1. Overall, SEED consistently achieves the highest average DSR across all evaluated models, improving both reasoning-level and final-answer safety. In detail, SEED achieves relative improvements of 20.5%, 9.7%, and 11.8% compared to the strongest baseline in overall DSR on DeepSeek-R1-Distill-Llama-8B, DeepSeek-R1-Distill-Qwen-7B, and Qwen3-8B, respectively. The overall DSR is computed by averaging reasoning-level and final-answer DSR over six attacks. These results demonstrate the effectiveness of SEED in improving jailbreak defense across different model families.

When observing different attacks, SEED maintains stable defense effectiveness by naturally eliciting the model's safety capability during the decoding process rather than relying on attack-specific prompt patterns. As shown in Table 1, SEED maintains strong DSR against prompt construction and rewriting attacks such as DrAttack and ReNeLLM, as well as gradient-based suffix attacks such as GCG. This suggests that SEED can leverage safety-eliciting search paths during decoding to identify obfuscated or repackaged malicious intent and defend against these attacks.

More importantly, SEED shows clear advantages under the reasoning-oriented H-CoT attack. H-CoT uses explicit reasoning guidance in the prompt to bias the model's reasoning process, encouraging it to decompose the harmful request and proceed along an unsafe trajectory before producing the final answer. Such CoT-based induction, however, can be effectively mitigated when SEED provides a safety-eliciting trajectory during decoding, allowing the model to effectively identify the underlying malicious intent. Therefore, SEED can defend against attacks that specifically target the model's reasoning trajectory.

Compared with competitive baselines, SEED achieves more consistent defense performance across diverse jailbreak attacks, while existing methods tend to be more sensitive to specific attack settings. PPL [43] and Retokenization [44] show limited effectiveness on attacks such as DrAttack and ReNeLLM, because prompt-level detection and input transformation cannot reliably capture harmful intent hidden through decomposition, paraphrasing, or instruction reformulation. They are also less effective against reasoning-oriented attacks such as H-CoT, where explicit reasoning guidance biases the model toward unsafe reasoning trajectories. Self-Reminder [20] can improve some attacks, such as DrAttack, by adding safety instructions. However, such general instructions are often too weak to counter attacks such as ReNeLLM, where the malicious intent is heavily embedded in indirect instructions. Self-Exam [16] introduces an external guard after generation, but classifying harmfulness in long reasoning traces is difficult, especially when unsafe content is expressed indirectly in seemingly benign reasoning. This leads to limited DSR on attacks like H-CoT and introduces additional overhead.

ReasoningGuard [19] and ThinkingI [18] are more competitive baselines specifically designed for reasoning models, using predefined intervention phrases to interrupt reasoning traces. ReasoningGuard achieves competitive results against some attacks, but struggles to defend against the reasoning-trajectory perturbations introduced by H-CoT. This suggests that coarse-grained safety insertions may fail to recover safe reasoning once the trajectory has been perturbed. As for ThinkingI, in particular, it achieves high reasoning-level DSR on Qwen3-8B, while its final-answer DSR remains much lower. This discrepancy mainly stems from the intervention phrase used by ThinkingI, which tends to prematurely terminate the reasoning chain, causing the model to stop reasoning before exposing any harmful reasoning content. However, this early termination does not reliably redirect the subsequent final response toward safe completion, so the model can still produce harmful content in the final answer. This also indicates that coarsely interrupting CoT reasoning with fixed phrases can impair the model's reasoning capability, a finding further supported by the utility analysis in Section 7.3. In contrast, by naturally exploring a broader set of trajectories during decoding and smoothly eliciting the model's safety capabilities, SEED can achieve high DSR against diverse attacks.

TABLE 2: Utility performance of different defense methods across three models. Higher values indicate better performance.

Method	*DeepSeek-R1-Distill-Llama-8B*				*DeepSeek-R1-Distill-Qwen-7B*				*Qwen3-8B*			
	MATH	**GPQA**	**Miner.**	**Avg.**	**MATH**	**GPQA**	**Miner.**	**Avg.**	**MATH**	**GPQA**	**Miner.**	**Avg.**
No Defense	**0.87**	0.53	0.56	**0.65**	0.86	0.55	**0.71**	0.71	0.91	0.54	0.73	0.73
PPL	0.86	0.53	0.56	**0.65**	0.87	0.51	**0.71**	0.70	0.88	0.49	0.71	0.69
Retokenization	0.43	0.41	0.25	0.36	0.65	0.37	0.61	0.54	0.86	0.45	0.71	0.67
Self-Reminder	0.74	**0.58**	0.51	0.61	**0.90**	0.53	0.70	0.71	0.87	0.49	0.52	0.63
Self-Exam	0.70	0.37	0.36	0.48	0.87	0.50	0.69	0.69	0.80	0.44	0.64	0.63
ReasoningGuard	0.82	0.51	**0.59**	0.64	0.78	0.47	0.60	0.62	0.89	0.46	0.75	0.70
ThinkingI	0.78	0.47	0.42	0.56	0.89	0.57	0.64	0.70	0.90	**0.63**	**0.76**	**0.76**
SEED	0.86	0.55	0.53	**0.65**	0.88	**0.58**	0.70	**0.72**	**0.93**	0.53	0.74	0.73

Robustness against Adaptive Attacks. To further examine the robustness of SEED, we conducted a targeted analysis where the attacker is aware of the internal mechanisms of SEED and explicitly adapts the attack to interfere with its decoding process. Specifically, on DeepSeek-R1-Distill-Qwen-7B, we constructed white-box variants of GCG that directly target the two signals used by SEED. A1 performs entropy suppression to evade the exploration triggering mechanism, while A2 performs attention suppression to disrupt the trajectory convergence mechanism. We further consider A1+A2, which jointly optimizes both objectives.

TABLE 3: Defense success rate (DSR) of SEED under adaptive attacks on DeepSeek-R1-Distill-Qwen-7B.

Attack	Rsng. DSR	Final DSR
A1	0.96	0.97
A2	0.96	0.96
A1+A2	0.95	0.98

As shown in Table 3, SEED maintains high DSR under both individually adaptive attacks and their joint variant. In particular, under the joint attack that simultaneously targets the entropy-based triggering mechanism and the attention-based convergence mechanism, SEED achieves reasoning/final DSR of $0.95/0.98$. These results indicate that explicitly suppressing either the triggering signal or the convergence signal, or both jointly, does not substantially degrade the effectiveness of SEED. This robustness is consistent with the exploratory design of SEED, where repeated exploration and soft guidance reduce its dependence on any single triggering or convergence decision.

7.3. Utility Analysis

To answer RQ2, we evaluated the utility performance of different defense methods on three general reasoning benchmarks, as summarized in Table 2. Overall, SEED preserves the original reasoning capability across all evaluated models. For example, on DeepSeek-R1-Distill-Qwen-7B, SEED achieves an average utility of 0.72, which is comparable to the undefended model and higher than other defense baselines. This indicates that the robustness improvement of SEED does not come with substantial degradation on the evaluated benign tasks.

In contrast, several baseline defenses introduce noticeable utility degradation. Retokenization significantly reduces utility, especially on DeepSeek-R1-Distill-Llama-8B, where the average score drops from 0.65 to 0.36. This degradation is likely because retokenization alters the original tokenization structure of the input, which may disrupt the model's learned reasoning patterns and make it harder to maintain coherent multi-step problem solving. Self-Exam and Self-Reminder also show non-negligible performance loss on some models and benchmarks, suggesting that additional safety prompts or post-hoc checking can interfere with benign reasoning when applied uniformly. CoT-intervention defenses, such as ReasoningGuard and ThinkingI, can preserve utility in certain cases, but their performance is less stable across models. For example, ThinkingI performs well on Qwen-3-8B but substantially degrades MinervaMath on DeepSeek-R1-Distill-Llama-8B. This is because the trigger words used by ThinkingI cause the model to prematurely terminate or shorten its chain-of-thought process. This leads to a severe drop in reasoning performance on tasks that require long and careful derivations. For example, on MATH, ThinkingI produces reasoning traces averaging only 11 tokens on Qwen3, compared with 889 tokens for SEED, indicating a much more aggressive shortening of the reasoning process.

These results demonstrate that SEED effectively defends against jailbreak attacks while preserving the model's reasoning performance. Unlike defenses that globally transform inputs, inject fixed safety reminders, or impose strong constraints throughout generation, SEED preserves the original trajectory as a candidate and performs additional exploration towards safe reasoning. This design allows the model to retain its native reasoning behavior on benign tasks.

Besides, we further analyze the false-refusal behavior of SEED. Specifically, we evaluate SEED on Qwen3-8B using XSTest [45], which includes benign requests that may superficially resemble harmful ones. This evaluation provides an additional view of SEED's behavior in such potentially confusing cases. As shown in Table 4, SEED increases the false-refusal rate from 0.02 to 0.08, with an increase of only 0.06. Overall, considering the malicious

requests in our evaluation, SEED improves the F1 score from 0.46 to 0.93, reflecting a better balance between true refusal and false refusal.

TABLE 4: False-refusal and true-refusal performance of SEED on Qwen3-8B.

Method	False-Refusal	True-Refusal	F1
No Defense	0.02	0.30	0.46
SEED	0.08	0.87	0.93

7.4. Ablation Study and Sensitivity Analysis

To answer RQ3, we conducted ablation studies to examine the effectiveness of SEED's two core components. We also evaluated SEED's sensitivity to decoding-time parameters and model scales. We report the average DSR against four representative attacks spanning the considered categories, namely PAIR, DrAttack, GCG, and H-CoT, which cover LLM-based prompt optimization, prompt rewriting and reconstruction, gradient-based suffix optimization, and reasoning-oriented attacks. The utility evaluation is on MATH, since it directly measures multi-step problem solving and is sensitive to disruptions during the reasoning.

TABLE 5: Ablation study.

Method	Avg. DSR↑		Utility↑
	Rsng.	Final.	
SEED	0.90	0.92	0.88
w/o Attention-based Convergence	0.29	0.37	0.90
w/o Safety Guidance	0.18	0.23	0.89

Ablation Study. Table 5 reports the ablation results on DeepSeek-R1-Distill-Qwen-7B. We evaluated the contribution of SEED's two key designs through stepwise ablation experiments, progressively removing the early decoding convergence mechanism and the safety-eliciting guidance.

First, to examine the role of the attention-based convergence mechanism, we replaced it with a common likelihood-based selection strategy used in beam search [31], where candidate trajectories were selected according to their average log probability. This variant substantially reduces DSR, where the reasoning-level DSR drops from 0.90 to 0.29, and the final-answer DSR drops from 0.92 to 0.37. This is because average log probability only reflects the model's preference for a continuation, while unsafe continuations induced by jailbreak attacks can also gain high probability scores. As a result, without an attention-based signal, decoding may still proceed along harmful trajectories even when SEED explores a safe trajectory.

Second, we further replaced the safety guidance on the additional trajectory with top-probability selection. In this setting, the decoding strategy degenerates into a native two-beam search. The third line of Table 5 depicts a sharp drop in both reasoning-level and final-answer DSR. The reasoning-level DSR decreases to 0.18, while the final-answer DSR decreases to 0.23. In detail, the parallel branch is still generated after the trigger point, but the safety-eliciting branch no longer receives the early-stage logit bonus. As a result, the model is no longer elicited to search for a safety-oriented trajectory. Instead, it only explores an unguided alternative path. The large performance drop indicates that simply increasing decoding-time exploration is insufficient for jailbreak defense.

Overall, these results suggest that guided exploration and attention-based trajectory convergence are both important to the effectiveness of SEED. Safety guidance elicits the model's safety capabilities, while attention-based convergence identifies trajectories for safe reasoning. Together, these two components form a cohesive decoding mechanism that helps the model identify a safety-eliciting trajectory when handling malicious requests.

Parameter Analysis. To evaluate SEED's hyperparameter choices, we conducted a comprehensive analysis of four key settings. The results are presented in Figures 4 and 5.

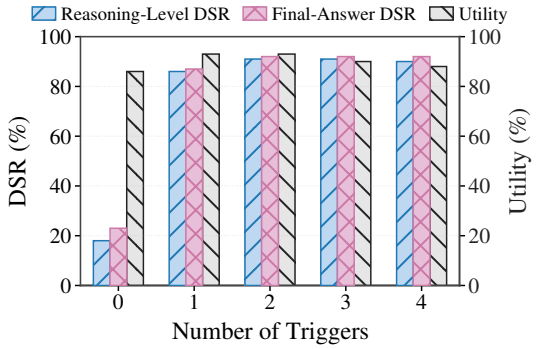
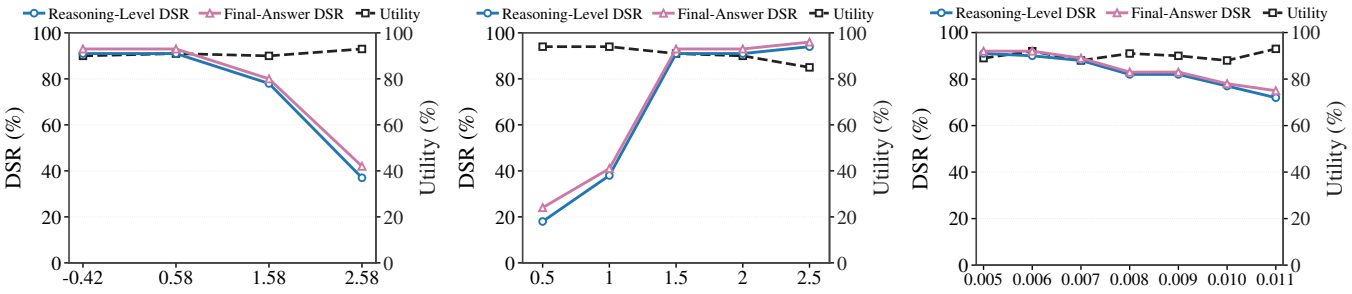


Figure 4: Effect of the max number of triggers K during generation on DeepSeek-R1-Distill-Qwen-7B.

Trigger Number: Figure 4 studies the effect of the maximum number of triggers during generation. When no trigger is allowed, SEED cannot perform safety-eliciting trajectory exploration, resulting in low DSR. As the trigger budget increases to two, both reasoning-level DSR and final-answer DSR improve substantially, showing that a small number of trigger positions is sufficient to expose safe trajectories in most cases. When the trigger budget further increases beyond two, the defense performance changes only marginally. The DSR remains stable, suggesting that additional triggers provide limited extra benefit once the main unsafe trigger positions have been explored. Meanwhile, the utility score shows no obvious degradation across different trigger budgets, indicating that sparse exploration does not noticeably disrupt benign reasoning. These results indicate that two triggers are sufficient to capture most of the defense benefit, while larger budgets introduce little additional defense gain but increase overhead.

First Trigger Threshold: We further evaluated the effect of the first trigger threshold as shown in Figure 5a. A smaller threshold makes SEED more sensitive to uncertain



(a) First Trigger Threshold δ_0 on Log Scale (b) Logit Bonus Strength α (c) Attention Threshold η_{sel}

Figure 5: Parameter analysis results on DeepSeek-R1-Distill-Qwen-7B.

decoding states and therefore more likely to activate safety-eliciting decoding. In this range, both reasoning-level DSR and final-answer DSR remain high, indicating that potential risk points can be explored before unsafe continuations dominate the reasoning. As the threshold increases, triggering becomes more conservative. Moderate threshold values still maintain strong DSR and stable utility, suggesting that SEED does not rely on a single narrowly tuned threshold. However, when the threshold becomes too large, the defense may rarely activate the safety-eliciting branch, or even fail to trigger it at all. In this case, SEED degenerates toward ordinary generation, and the model is more likely to continue along the default unsafe trajectory. This explains the sharp drop in both reasoning-level and final-answer DSR under overly large thresholds.

Logit Bonus Strength: Figure 5b studies the effect of the logit bonus strength used in the safety-eliciting trajectory. The bonus strength controls how strongly the exploratory trajectory deviates from the original decoding path toward safety-eliciting tokens. When the bonus strength is small, e.g., 0.5 or 1.0, both reasoning-level and final-answer DSR remain relatively low, indicating that weak guidance is insufficient to shift the trajectory and elicit the model's internal safety capability. As the bonus strength increases to 1.5, both DSR metrics improve sharply, showing that a moderate bonus can induce safety-aware reflection and expose latent safe continuations while preserving benign reasoning. Further increasing the bonus strength to 2.5 brings only marginal additional improvement in DSR, while the utility score slightly decreases. This is because overly large bonuses may behave similarly to fixed-phrase interventions that coarsely interrupt reasoning, thereby harming the model's reasoning utility. Overall, a moderate bonus strength like 1.5 provides a better balance between safety elicitation and reasoning preservation.

Attention Threshold: Figure 5c studies the sensitivity of SEED to the attention threshold used in trajectory convergence. The threshold determines how much internal attention evidence is required for selecting the safety-eliciting trajectory. When the threshold is small, both reasoning-level and final-answer DSR remain high, suggesting that SEED can effectively recover safe continuations once they attend sufficiently to the safety-eliciting tokens. As the threshold increases, the convergence criterion becomes more conservative, making SEED less likely to adopt the safety-eliciting trajectory and causing both DSR metrics to decrease gradually. Meanwhile, the utility score changes slightly across thresholds, indicating that the attention threshold has a limited impact on benign reasoning. Overall, these results show that SEED is robust within a moderate threshold range, while an overly strict threshold may miss valid safe continuations and reduce defense effectiveness.

TABLE 6: Evaluation of SEED on different model scales. Results on attacks are reported as reasoning/final DSR.

	DeepSeek-R1-Distill-Qwen-14B		*DeepSeek-R1-Distill-Qwen-32B*	
Attack	**Original**	**SEED**	**Original**	**SEED**
PAIR	0.18/0.19	0.98/1.00	0.19/0.19	0.95/0.98
DrAttack	0.38/0.49	0.99/1.00	0.66/0.67	0.99/0.98
GCG	0.70/0.85	0.99/1.00	0.97/0.97	0.99/0.99
H-CoT	0.00/0.00	0.90/0.92	0.00/0.00	0.90/0.92
Avg. DSR	0.32/0.38	0.97/0.98	0.46/0.46	0.96/0.97
Utility	0.94	0.92	0.93	0.92

Model Scale Analysis. To examine whether the effectiveness of SEED generalizes to larger model scales, we further evaluate SEED on DeepSeek-R1-Distill-Qwen-14B and DeepSeek-R1-Distill-Qwen-32B. As shown in Table 6, SEED remains effective on both the 14B and 32B models, while largely preserving reasoning utility. These results extend the effectiveness of SEED beyond the smaller models evaluated in the main experiments. Because SEED operates at decoding time and relies on signals naturally available during generation, its exploration and selection mechanisms do not depend on a specific model size. The consistent improvements across the 14B and 32B models therefore provide further empirical evidence that SEED can be applied to LRMs at different scales.

7.5. Overhead and System-Level Deployment

To answer RQ4, we evaluated the runtime overhead and deployment feasibility of SEED by integrating it into SGLang [40], a realistic LLM serving framework. As shown

in Table 7, SEED can be efficiently integrated into SGLang with limited serving overhead.

TABLE 7: Serving efficiency of the SGLang implementation on DeepSeek-R1-Distill-Qwen-7B. ↑ means higher is better, and ↓ means lower is better.

Metric	Original	Ours	Relative
ITL (ms) ↓	16.39	16.68	+1.8%
Throughput (tok/s) ↑	60.96	55.21	-9.4%
E2E latency (s) ↓	24.12	25.31	+4.9%
Cache Reuse (%) ↑	5.67	95.09	+89.4 pp

We evaluated four metrics of serving performance from different aspects. In detail, SEED introduces only an acceptable token-level overhead. The inter-token latency (ITL) increases slightly from 16.39 ms to 16.68 ms, corresponding to only a 1.8% increase. ITL is calculated on all generated tokens, including those on abandoned paths. As for the overall decode throughput, it decreases from 60.96 to 55.21 tokens/s, corresponding to a 9.4% reduction. This overhead mainly comes from the additional safety-eliciting trajectory and trajectory-level signal collection. Importantly, the end-to-end latency, measured as the per-sample inference time, changes from 24.12 s to 25.31 s with a slight increase of only 4.9%. This suggests that the trajectory search in SEED does not slow down individual inference too much. Although it introduces additional exploration at selected trigger points, the explored trajectories share the same prefix and can effectively reuse KV cache, as reflected by the substantial increase in prefix cache reuse from 5.67% to 95.09%.

These results suggest that by exploiting prefix sharing in SGLang, additional trajectory exploration can be implemented with moderate serving overhead and slight end-to-end latency, demonstrating the practicality of deploying SEED in real LLM serving frameworks.

8. Discussion

Classifiers for Jailbreak Attacks. Many jailbreak defenses rely on classifiers to directly detect malicious semantics or to conditionally process suspicious inputs [17], [46]. Their effectiveness therefore depends heavily on classifier precision. However, building a reliable boundary between benign and malicious generations is inherently difficult, especially under unseen jailbreaks with obfuscated malicious intents [37]. Thus, though SEED also adopts conditional mechanisms for malicious requests, such as using a trigger to activate parallel exploration and attention signals for trajectory convergence, it is designed to be robust to imperfect trigger and convergence decisions. This robustness stems from treating safety defense as exploratory decoding. SEED can repeatedly explore trajectories to elicit the model's native safety capability, tolerating imperfect decisions. Moreover, soft logit modulation gently guides decoding, allowing LRMs to autonomously explore their reasoning space rather than imposing a hard intervention.

Reasoning Search Paradigms. Since LRMs can reason and reflect over long generation trajectories, prior work has explored ways to expand the reasoning space by explicitly structuring intermediate thoughts. Representative paradigms, such as Tree-of-Thoughts (ToT) [47] and Graph-of-Thoughts (GoT) [48], organize thoughts into tree-structured or graph-structured states and search over alternative reasoning branches. Although these methods and SEED both explore alternatives beyond a single default reasoning path, they operate on different representations of reasoning. ToT and GoT search over explicit thought states through external generation, evaluation, and selection, which can introduce considerable inference overhead. In contrast, SEED operates as part of the LLM generation pipeline, exploring alternative trajectories within the model's native decoding process, under the guidance of model internal signals. Thus, SEED is orthogonal to ToT and GoT paradigms. They search over structured thought states, whereas SEED searches over latent decoding trajectories.

Mechanism of Soft Logit Modulation. The safety-eliciting sequence R provides soft, context-dependent guidance rather than enforcing a fixed safety template. For example, among 89 triggered trajectories on MATH, the model produces 64 distinct realizations of R, illustrating substantial contextual variation. Importantly, SEED does not treat individual safety-related tokens as standalone indicators of safety. Instead, safety elicitation is realized through a context-dependent sequence composed of multiple tokens, whose joint effect depends on the surrounding reasoning context rather than on any isolated token. The decaying guidance is therefore intended to encourage broader reflective reasoning patterns. This lightweight design also provides a practical trade-off for deployment, as it avoids costly online search for safety guidance or auxiliary models.

Adaptation to Normal LLMs. This paper focuses on exploring LRMs' reasoning trajectories to elicit the model's internal safety capability. However, SEED is a general decoding strategy, and has the potential to be applied to a broader range of autoregressive LLMs. Although normal LLMs may not expose long reasoning trajectories as explicitly as LRMs, they also generate alternative continuations during decoding, and parallel decoding has been explored for improving generation quality. Similarly, SEED can also be incorporated as a safety-oriented branch in parallel decoding and identify safer responses accordingly, enabling its extension from LRMs to general-purpose LLMs. We will explore this direction in future work.

9. Related Work

Jailbreak attacks have been a serious concern for LLMs, and a growing line of work has explored how to defend against them. Existing defenses can be broadly categorized into training-based and training-free methods.

Training-Based Defenses. Training-based defenses improve model robustness against jailbreak attacks through additional training or alignment [27], [49], [50], [51], [52], [53].

These methods typically incorporate safety-oriented data, preference signals, or reasoning supervision to strengthen the model's intrinsic refusal capability. Although such methods can improve model-level safety, they typically require additional training data and extra optimization overhead. Moreover, because the model is trained on known attack patterns, it may remain brittle against unseen jailbreaks.

Training-Free Defenses. Training-free defenses improve model safety without updating model parameters, making them practical for deployed LRMs. The first line of work focuses on input preprocessing. For example, perplexity-based filtering identifies abnormal prompts that contain adversarial jailbreak patterns, while retokenization perturbs the input representation to weaken attack effectiveness [43], [44]. Although lightweight and easy to deploy, these methods mainly intervene before generation and may fail to address unsafe content in the reasoning trajectory.

Additionally, many methods attach guard mechanisms around the target model. These guards detect model outputs and block unsafe responses [16], [17], [54]. Although they leave the target model unchanged, they often incur substantial inference overhead due to additional model calls, especially for LRMs. Moreover, defining a reliable classification boundary between benign and malicious content remains challenging, particularly under unseen jailbreaks and obfuscated harmful intents [37]. Furthermore, these guard mechanisms mainly focus on filtering generated content, leaving the intermediate reasoning process largely untouched.

Some training-free defenses further perform inference-time intervention on the reasoning process itself [18], [19]. These methods usually intervene in LRM's intermediate reasoning steps by inserting a sentence like "I am a responsible assistant...". Those fixed sentences will redirect the model's reasoning to a limited scope, from which LRM may fail to restore the normal reasoning for benign requests. Beyond fixed-form intervention, other inference-time defenses rely on additional guidance signals or trajectory-level search. For example, SafeDecoding [55] introduces expert models, RDS [56] leverages trained hidden-state classifiers, and SEA [57] utilizes reward models to provide additional guidance. Other methods, such as RAIN [58] and SafeChain [27], perform trajectory-level search through iterative generation, evaluation, and selection. While these methods improve jailbreak robustness, their reliance on auxiliary models or iterative search introduces additional deployment complexity and inference overhead, which can be challenging for LRMs with long reasoning trajectories. In contrast, SEED elicits safety-aligned reasoning directly during native decoding, which supports direct integration into SGLang, a real-world LLM serving framework.

10. Conclusion

In this paper, we presented SEED, a lightweight inference-time defense for LRMs. Instead of treating jailbreak defense as post-hoc detection or direct intervention during reasoning, SEED improves safety by exploring alternative decoding trajectories. This design allows the model to elicit its internal safety capability and reach safe reasoning trajectories without broadly disrupting benign reasoning. Across six jailbreak attacks and three LRMs, SEED consistently improves both reasoning-level and final-answer safety while preserving utility on benign tasks. We also implemented SEED into SGLang, demonstrating that SEED can be integrated into production-level serving frameworks without inducing substantial overhead.

ACKNOWLEDGEMENTS

We would like to express our appreciation to the anonymous reviewers for their valuable and constructive feedback, as well as to the shepherd for the guidance in improving the paper. This research is supported in part by the NSFC Programs (No. 62602199, No. 62672273, No. 62602196) and Beijing Natural Science Foundation (No. L2607019).

References

[1] Z.-Z. Li, D. Zhang, M.-L. Zhang, J. Zhang, Z. Liu, Y. Yao, H. Xu, J. Zheng, P.-J. Wang, X. Chen *et al.*, "From system 1 to system 2: A survey of reasoning large language models," *arXiv preprint arXiv:2502.17419*, 2025.

[2] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[3] OpenAI, "Learning to reason with llms," https://openai.com/index/learning-to-reason-with-llms/, 2024.

[4] ——, "Introducing openai o3 and o4-mini," https://openai.com/index/introducing-o3-and-o4-mini/, 2025.

[5] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the math dataset," *arXiv preprint arXiv:2103.03874*, 2021.

[6] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

[7] C. He, R. Luo, Y. Bai, S. Hu, Z. Thai, J. Shen, J. Hu, X. Han, Y. Huang, Y. Zhang *et al.*, "Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 3828–3850.

[8] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan, "Swe-bench: Can language models resolve real-world github issues?" in *International Conference on Learning Representations*, vol. 2024, 2024, pp. 54 107–54 157.

[9] N. Jain, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica, "Livecodebench: Holistic and contamination free evaluation of large language models for code," in *International Conference on Learning Representations*, vol. 2025, 2025, pp. 58 791–58 831.

[10] EvalScope, "Minerva-Math," https://evalscope.readthedocs.io/en/latest/benchmarks/minerva_math.html.

[11] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang *et al.*, "Mmlu-pro: A more robust and challenging multi-task language understanding benchmark," *Advances in Neural Information Processing Systems*, vol. 37, pp. 95 266–95 290, 2024.

[12] K. Zhou, C. Liu, X. Zhao, S. Jangam, J. Srinivasa, G. Liu, D. Song, and X. E. Wang, "The hidden risks of large reasoning models: A safety assessment of r1," in *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, 2025, pp. 3250–3265.

[13] Y. Huang, Y. Sun, S. Ruan, Y. Zhang, Y. Dong, and X. Wei, "Breaking the ceiling: Exploring the potential of jailbreak attacks through expanding strategy space," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 7870–7888.

[14] C. Wang, Y. Liu, B. Li, D. Zhang, Z. Li, and J. Fang, "Safety in large reasoning models: A survey," *arXiv preprint arXiv:2504.17704*, 2025.

[15] M. Kuo, J. Zhang, A. Ding, Q. Wang, L. DiValentin, Y. Bao, W. Wei, H. Li, and Y. 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.

[16] M. Phute, A. Helbling, M. Hull, S. Peng, S. Szyller, C. Cornelius, and D. H. Chau, "Llm self defense: By self examination, llms know they are being tricked," *arXiv preprint arXiv:2308.07308*, 2023.

[17] H. Inan, K. Upasani, J. Chi, R. Rungta, K. Iyer, Y. Mao, M. Tontchev, Q. Hu, B. Fuller, D. Testuggine *et al.*, "Llama guard: Llm-based input-output safeguard for human-ai conversations," *arXiv preprint arXiv:2312.06674*, 2023.

[18] T. Wu, C. Xiang, J. T. Wang, G. E. Suh, and P. Mittal, "Effectively controlling reasoning models through thinking intervention," *arXiv preprint arXiv:2503.24370*, 2025.

[19] Y. Wang, M. Zhang, Y. Wang, G. Hong, M. Wen, X. You, and M. Yang, "Reasoningguard: Safeguarding large reasoning models with inference-time safety aha moments," *arXiv preprint arXiv:2508.04204*, 2025.

[20] Y. Xie, J. Yi, J. Shao, J. Curl, L. Lyu, Q. Chen, X. Xie, and F. Wu, "Defending chatgpt against jailbreak attack via self-reminders," *Nature Machine Intelligence*, vol. 5, no. 12, pp. 1486–1496, 2023.

[21] P. Chao, E. Debenedetti, A. Robey, M. Andriushchenko, F. Croce, V. Sehwag, E. Dobriban, N. Flammarion, G. J. Pappas, F. Tramer *et al.*, "Jailbreakbench: An open robustness benchmark for jailbreaking large language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 55 005–55 029, 2024.

[22] P. Chao, A. Robey, E. Dobriban, H. Hassani, G. J. Pappas, and E. Wong, "Jailbreaking black box large language models in twenty queries," in *IEEE Conference on Secure and Trustworthy Machine Learning, SaTML 2025, Copenhagen, Denmark, April 9-11, 2025*. IEEE, 2025, pp. 23–42. [Online]. Available: https://doi.org/10.1109/SaTML64287.2025.00010

[23] "Aim prompt jailbreak attack," https://deepleaps.com/prompt/chatgpt-aim-jailbreak-prompt/.

[24] P. Ding, J. Kuang, D. Ma, X. Cao, Y. Xian, J. Chen, and S. Huang, "A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 2136–2153.

[25] X. Li, R. Wang, M. Cheng, T. Zhou, and C.-J. Hsieh, "Drattack: Prompt decomposition and reconstruction makes powerful llms jailbreakers," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 13 891–13 913.

[26] A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," *arXiv preprint arXiv:2307.15043*, 2023.

[27] F. Jiang, Z. Xu, Y. Li, L. Niu, Z. Xiang, B. Li, B. Y. Lin, and R. Poovendran, "Safechain: Safety of language models with long chain-of-thought reasoning capabilities," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 23 303–23 320.

[28] C. Shi, H. Yang, D. Cai, Z. Zhang, Y. Wang, Y. Yang, and W. Lam, "A thorough examination of decoding methods in the era of LLMs," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 8601–8629.

[29] J. Gu, K. Cho, and V. O. Li, "Trainable greedy decoding for neural machine translation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1968–1978.

[30] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019.

[31] M. Freitag and Y. Al-Onaizan, "Beam search strategies for neural machine translation," in *Proceedings of the First Workshop on Neural Machine Translation*, 2017, pp. 56–60.

[32] L. Gui, C. Gârbacea, and V. Veitch, "Bonbon alignment for large language models and the sweetness of best-of-n sampling," *Advances in Neural Information Processing Systems*, vol. 37, pp. 2851–2885, 2024.

[33] X. Li, Z. Zhou, J. Zhu, J. Yao, T. Liu, and B. Han, "Deepinception: Hypnotize large language model to be jailbreaker," *arXiv preprint arXiv:2311.03191*, 2023.

[34] Z. Yang, M. Zhang, F. Chen, G. Ding, L. Hou, X. Tao, and Y.-C. Chen, "Less is more: Improving llm reasoning with minimal test-time intervention," *arXiv preprint arXiv:2510.13940*, 2025.

[35] S. Wang, L. Yu, C. Gao, C. Zheng, S. Liu, R. Lu, K. Dang, X. Chen, J. Yang, Z. Zhang, Y. Liu, A. Yang, A. Zhao, Y. Yue, S. Song, B. Yu, G. Huang, and J. Lin, "Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning," in *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

[36] G. Shen, D. Zhao, H. Tong, J. Li, F. Zhao, and Y. Zeng, "Safety instincts: Llms learn to trust their internal compass for self-defense," in *International Conference on Learning Representations*, vol. 2026, 2026, pp. 91 842–91 873.

[37] W. Hackett, L. Birch, S. Trawicki, N. Suri, and P. Garraghan, "Bypassing prompt injection and jailbreak detection in llm guardrails," *arXiv e-prints*, pp. arXiv–2504, 2025.

[38] W. Zhao, X. Ren, J. Hessel, C. Cardie, Y. Choi, and Y. Deng, "Wildchat: 1m chatgpt interaction logs in the wild," *arXiv preprint arXiv:2405.01470*, 2024.

[39] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45.

[40] L. Zheng, L. Yin, Z. Xie, C. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez *et al.*, "Sglang: Efficient execution of structured language model programs," *Advances in neural information processing systems*, vol. 37, pp. 62 557–62 583, 2024.

[41] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[42] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman, "Gpqa: A graduate-level google-proof q&a benchmark," *arXiv preprint arXiv:2311.12022*, 2023.

[43] G. Alon and M. Kamfonas, "Detecting language model attacks with perplexity," *arXiv preprint arXiv:2308.14132*, 2023.

[44] N. Jain, A. Schwarzschild, Y. Wen, G. Somepalli, J. Kirchenbauer, P.-y. Chiang, M. Goldblum, A. Saha, J. Geiping, and T. Goldstein, "Baseline defenses for adversarial attacks against aligned language models," *arXiv preprint arXiv:2309.00614*, 2023.

[45] P. Röttger, H. Kirk, B. Vidgen, G. Attanasio, F. Bianchi, and D. Hovy, "Xstest: A test suite for identifying exaggerated safety behaviours in large language models," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 5377–5400.

[46] Meta AI, "Llama prompt guard 2," https://www.llama.com/docs/model-cards-and-prompt-formats/prompt-guard/, 2025.

[47] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in neural information processing systems*, vol. 36, pp. 11 809–11 822, 2023.

[48] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk *et al.*, "Graph of thoughts: Solving elaborate problems with large language models," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 16, 2024, pp. 17 682–17 690.

[49] W. Jeung, S. Yoon, M. Kahng, and A. No, "SAFEPATH: preventing harmful reasoning in chain-of-thought via early alignment," in *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

[50] S. Lee, S. Park, Y. Choi, G. Kim, M. Kang, J. Yun, D. Park, J. Park, and S. J. Hwang, "Thinksafe: Self-generated safety alignment for reasoning models," *arXiv preprint arXiv:2601.23143*, 2026.

[51] Z. Wang, H. Tu, Y. Wang, J. Wu, Y. Liu, J. Mei, B. R. Bartoldson, B. Kailkhura, and C. Xie, "Star-1: Safer alignment of reasoning llms with 1k data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, no. 44, 2026, pp. 37 988–37 997.

[52] Y. Zhang, S. Zhang, Y. Huang, Z. Xia, Z. Fang, X. Yang, R. Duan, D. Yan, Y. Dong, and J. Zhu, "Stair: Improving safety alignment with introspective reasoning," *arXiv preprint arXiv:2502.02384*, 2025.

[53] J. Zhu, L. Yan, S. Wang, D. Yin, and L. Sha, "Reasoning-to-defend: Safety-aware reasoning can defend large language models from jailbreaking," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 29 331–29 349.

[54] B. Upadhayay and V. Behzadan, "X-guard: Multilingual guard agent for content moderation," in *Proceedings of the The First Workshop on LLM Security (LLMSEC)*, 2025, pp. 54–86.

[55] Z. Xu, F. Jiang, L. Niu, J. Jia, B. Y. Lin, and R. Poovendran, "Safedecoding: Defending against jailbreak attacks via safety-aware decoding," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 5587–5605.

[56] X. Zeng, Y. Shang, J. Chen, J. Zhang, and Y. Tian, "Root defense strategies: Ensuring safety of llm at the decoding level," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 1974–1988.

[57] Y. Yuan, T. Xiao, Y. Li, B. Xu, S. Tao, Y. Qiu, H. Shen, and X. Cheng, "Inference-time alignment in continuous space," in *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

[58] Y. Li, F. Wei, J. Zhao, C. Zhang, and H. Zhang, "Rain: Your language models can align themselves without finetuning," in *International Conference on Learning Representations*, vol. 2024, 2024, pp. 20 580–20 595.

Appendix A. Ethics Considerations

This work studies jailbreak defense for large reasoning models and has potential dual-use implications. Although the goal is defensive, evaluating jailbreak attacks and analyzing unsafe model behavior could potentially inform misuse. To mitigate this risk, we focus on existing attack methods as evaluation benchmarks, do not propose a new jailbreak attack, report aggregate results, and avoid unnecessary reproduction of harmful model outputs.

This work does not involve human subjects research, personally identifiable information, or user data collection. It also does not involve interacting with, measuring, exploiting, or disrupting live systems that do not belong to the authors. All experiments are conducted in controlled offline settings using open-source models and research computing infrastructure.

The paper does not identify vulnerabilities in deployed third-party systems that require coordinated vulnerability disclosure. The main ethical consideration is the dual-use nature of jailbreak evaluation. We believe the defensive benefits of improving inference-time robustness outweigh the risks, and we take steps to limit misuse by presenting the work as a defense-focused evaluation rather than operational guidance for harmful behavior.

Appendix B. Meta-Review

The following meta-review was prepared by the program committee for the 2027 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

B.1. Summary

The paper proposes a lightweight exploratory decoding strategy that elicits LRMs' inherent safety capabilities without training. SEED operates in three steps: (1) trigger detection, (2) safety-guided parallel decoding, and (3) attention-based trajectory convergence. Extensive experiments on three open-source LRMs demonstrate that SEED improves defense success rate while preserving benign-task performance. The authors evaluate SEED on three open-source reasoning models against six jailbreak attacks and compare it with six training-free defenses. They report substantial improvements in defense success rate while maintaining performance on three benign reasoning benchmarks, and implement SEED in SGLang with limited serving overhead.

B.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field

B.3. Reasons for Acceptance

1) The paper provides a valuable step forward in an established field. Providing safety of the reasoning trace is an emerging and important problem, and the research question is practically significant for deployed LRMs. The motivating observation is clear and well-supported. The idea of exploring alternative decoding trajectories is interesting and the combination of selective branching, logit modulation, and early convergence is technically meaningful.
2) The method is efficient and practically deployable. Inspired by the costly brute-force best-of-N strategy, it offers a parallel decoding alternative with lower overhead: exploration is triggered selectively rather than continuously, only one additional trajectory is generated, and early convergence limits the added inference cost. The training-free design and the SGLang implementation with acceptable latency further strengthen the systems aspect of the work.
3) The evaluation is broad and the results are strong. It covers three models, six diverse attacks, six training-free baselines, and multiple utility benchmarks, with consistent gains across attacks and model families. The ablations show that both guided exploration and attention-based trajectory convergence contribute.

B.4. Noteworthy Concerns

1) Utility was primarily claimed on benchmarks that cannot detect false refusals. While the paper reported false refusal rates on XSTest on one model evaluation, these results do not sufficiently support utility claims on false refusal rates.
2) The attention-based convergence signal is not sufficiently validated. The experiments do not establish that attention to the safety-eliciting tokens reliably distinguishes safe and unsafe trajectories. Prior work shows that raw last-layer attention is not by itself a direct measure of token influence. It is therefore necessary to verify the correlation between the proposed attention score and trajectory safety.