Think Twice, Generate Once: Enhancing LLMs Safety via Progressive Self-Reflection

Anonymous ACL submission

Abstract

Large language models (LLMs) have revolutionized natural language processing with their ability to generate coherent and contextually relevant text. However, their deployment raises significant concerns about the potential for generating harmful or inappropriate content. In this paper, we introduce Progressive Self-Reflection, a novel inference-time technique that empowers LLMs to self-monitor and correct their outputs dynamically. Experimental results demonstrate that applying our proposed method to Llama-3.1-8B-Instruct reduces the attack success rate from 77.47% to 5.86%, to Llama-3.1-8B base from 89.70% to 5.56%, and to Qwen2.5-7B-Instruct from 44.44% to 3.84%, without additional training. Furthermore, our method maintains their original performance across diverse tasks, including summarization, general knowledge, reasoning, and mathematics. Our approach acts as a test-time scaling method, where additional self-reflection rounds enhance safety at the cost of inference overhead. To balance safety with computational efficiency, we introduce a lightweight self-reflection predictor that estimates the optimal number of reflection rounds based on input complexity. This adaptive mechanism prevents unnecessary self-assessment on benign inputs while ensuring thorough evaluation when encountering potentially harmful content. Our findings suggest that Progressive Self-Reflection serves as a scalable test-time approach, enhancing LLM safety by dynamically allocating computational resources in proportion to the input's risk profile. Our implementation is available at `https://anonymous.4open.science/r/PSR/`.

1 Introduction

Large Language Models (LLMs) such as GPT-4 (Achiam et al., 2023; Hurst et al., 2024), Llama (Touvron et al., 2023; Grattafiori et al., 2024), Deepseek (Liu et al., 2024a,b; Guo et al., 2025) have become integral to modern natural language processing, revolutionizing the ability of machines to understand and generate human-like text. These models have demonstrated impressive capabilities across a broad spectrum of tasks, including but not limited to machine translation, summarization, and automated content generation (Achiam et al., 2023; Wu et al., 2023). However, as the deployment and application of LLMs become more pervasive across various sectors-from healthcare (Singhal et al., 2023; Chen et al., 2023) to finance (Li et al., 2023b; Lee et al., 2025) and education (Wang et al., 2024a; Kobak et al., 2024) - the imperative to secure these systems against adversarial misuse grows ever more urgent. LLMs, due to their extensive training on diverse internet corpora, possess the capacity to generate content that spans a broad spectrum of topics and styles. However, this versatility also exposes them to the risk of generating harmful or unethical content when prodded by maliciously crafted inputs, commonly known as jailbreak attacks (Wei et al., 2023a; Shen et al., 2024). Such attacks exploit model vulnerabilities to elicit responses that breach the models' trained ethical guidelines, potentially leading to the dissemination of biased, unlawful, or otherwise inappropriate content (Weidinger et al., 2021; Zou et al., 2023; Liu et al., 2023b). Defending LLMs against such attacks is now recognized as a critical challenge for safe AI deployment.

Jailbreak attacks exploit model vulnerabilities to bypass safety mechanisms designed to prevent the generation of inappropriate responses. Such attacks not only pose risks to data integrity and user trust but also threaten the broader applicability of LLMs in sensitive environments (Bai et al., 2022b; Zhou et al., 2024). The arms race between evolving attack strategies and defense mechanisms mirrors challenges observed in other domains like computer vision, where advances in adversarial robustness often lag behind attack techniques (Carlini, 2024). In particular, current strategies for mitigating such risks include prompt engineering (Xie et al., 2023; Xiong et al., 2024), detection-based methods (Alon and Kamfonas, 2023; Hu et al., 2024; Candogan et al., 2025), and fine-tuning with curated datasets (Wei et al., 2023a; Liu et al., 2024c; Huang et al., 2024). However, these approaches often fall short when facing sophisticated, adaptive jailbreak strategies that continuously evolve to exploit new or overlooked vulnerabilities. Moreover, designing effective jailbreak defenses is inherently difficult. An ideal defense must walk a fine line between safety and utility: being overly strict can cause false refusals and degrade user experience, while being too lenient leaves the model open to attack. Prior methods sometimes result in over-defensiveness, rejecting benign inputs or significantly degrading the utility of the model (Varshney et al., 2023; Cao et al., 2024; Shi et al., 2024).

In response to these challenges, we propose **Progressive Self-Reflection** (**PSR**), a novel decoding-time defense mechanism that achieves strong jailbreak mitigation without altering the model's parameters or training procedure. The core idea of PSR is to embed an internal self-evaluation loop into the generation process. As the LLM generates a response, it pauses at regular intervals (e.g. every K tokens) to reflect on the partial output: essentially asking itself whether the content so far might violate any safety or policy constraints. This introspective check leverages the model's own knowledge of disallowed content and alignment guidelines. PSR leverages dynamic, runtime introspection where an LLM assesses its outputs at defined intervals for potential harmful content. Crucially, these safety interventions happen on the fly during inference, requiring no changes to the underlying model weights. In effect, PSR acts as an internal guardrail, dynamically course-correcting the model's output before any harmful content can fully materialize. Figure 1 illustrates this process using an example harmful prompt. The model initially begins to output harmful instructions but is intercepted mid-generation via self-reflection. The yellow boundary box simulates the thought process of the LLM: it initially plans to generate harmful responses (top-right), for example, providing instructions on how to steal when prompted with a malicious query, but through self-reflection (bottom-right), it identifies the issue and ultimately produces a safe refusal. This mechanism enables LLMs to dynamically detect and mitigate harmful completions during inference rather than relying solely on static post-hoc filtering.

A key challenge in implementing such frequent self-reflection is maintaining efficiency. We further address this with an adaptive reflection schedule powered by a lightweight MLP-based predictor. Before generation, this predictor analyzes the hidden representation of the input prompt and first few generated tokens to estimate the minimal number of reflection rounds needed for that query. Intuitively, a benign or straightforward query might only require one final safety check at the end, whereas a complex or suspicious prompt would benefit from more frequent checkpoints. By adjusting the reflection frequency to the input's risk level, PSR avoids unnecessary overhead on easy queries while still providing tight safety supervision on challenging ones. This design allows us to progressively apply just the right amount of self-reflection – increasing robustness when needed and saving computation when not. Notably, all of these mechanisms operate at inference time; we do not require any additional fine-tuning of the primary LLM (the small predictor network is the only learned component, and it is orders of magnitude smaller than the LLM).

In summary, our contributions are summarized as follows:

- **Progressive Self-Reflection (PSR)** A new test-time defense paradigm for LLMs that interleaves generation with internal safety reflection, enabling the model to catch and correct potential policy violations during its own decoding process. This approach improves safety compliance without any modifications to the model's weights or its training data.

- **Adaptive Reflection Planning** We introduce a lightweight predictor that estimates the required number of reflection steps based on the input prompt's features. This component allows PSR to dynamically balance safety and efficiency, applying more frequent checks for complex or risky prompts while minimizing slow-down on benign inputs.

- **Improved Jailbreak Robustness with Minimal Trade-offs** Through extensive experiments on multiple open-source LLMs such as Llama-3.1 (Touvron et al., 2023) and Qwen2.5 (Yang et al., 2024), we show that PSR dramatically reduces jailbreak attack success rates by up to 82% , preventing a wide range of adversarial prompts from eliciting forbidden out-

Figure 1: **Overview of our proposed method.** Given a potentially harmful user prompt (top-left), the LLM (bottom-left) generates an initial response "Sure, here is a guide for stealing from a store without getting caught" and begins to generate unsafe content, denote in red. Before completing the harmful response, a self-reflection prompt is injected (e.g. "Let's check if the generated text is harmful or harmless"), allowing the model to assess its own output. If the response is deemed harmful, the model backtracks and regenerates a safer alternative. Otherwise, the LLM continues generating without being affected by the probing tokens.

puts while preserving the model's helpfulness and accuracy on non-adversarial tasks. Our approach outperforms comparable decoding-time defenses in both effectiveness and computational overhead, pointing to a practical path for safer LLM deployment.

2 Related work

2.1 LLM Jailbreak Attacks and General Defense Methods

Jailbreak Attacks: Large Language Models (LLMs) are vulnerable to prompt-based adversarial attacks known as jailbreaks, where carefully crafted inputs induce the model to ignore safety instructions (Jain et al., 2023; Yu et al., 2024). These attacks range from simple role-play prompts (Yi et al., 2024; Sun et al., 2024; Shen et al., 2024) (e.g. the infamous "Do Anything Now" prompt) to automated prompt optimizations. For example, recent work has shown that gradient-guided methods can discover input tokens that consistently elicit policy-breaking outputs (Wallace et al., 2019; Zhu et al., 2023; Yu et al., 2024). Other strategies include using one LLM to rephrase a blocked query into a seemingly benign form, or applying genetic algorithms to evolve prompts that bypass filters (Zhu et al., 2023; Chang et al., 2024) . Such techniques can circumvent even advanced alignment measures, easily evading models fine-tuned with human feedback (Ouyang et al., 2022). As a result, jailbreak attacks have exposed a serious gap between a model's average-case safety and its worst-case robustness when facing a dedicated adversary. General Defense Strategies: To harden LLMs against jailbreaks, researchers have explored improved safety-alignment during training. A primary approach is instruction tuning and Reinforcement Learning from Human Feedback (RLHF) geared towards refusals. For instance, Bai et al. (2022a) and Tan et al. (2023) train models to be helpful yet harmless, meaning they will politely refuse disallowed requests. Such refusal training uses supervised fine-tuning and RLHF with preference models that reward safe behavior, yielding assistants that decline harmful queries in a friendly manner. While RLHF dramatically reduces a model's tendency to produce toxic or illicit content, it does not guarantee robustness to more sophisticated attacks (Jain et al., 2023).

2.2 Test-Time Methods for LLM Jailbreak Defense

While training alignment is crucial, runtime safeguards are often employed as a last line of defense when the model is deployed. A straightforward approach is to wrap the LLM with a moderation filter or guardrail system (Dong et al., 2024) . Such guardrails inspect user inputs and model outputs and can refuse or transform them if they are deemed unsafe. For instance, a moderation module may detect when a query involves illegal instructions ("How to hack a website?") and block or modify it before it ever reaches the LLM (Milvus, 2025). Likewise, generated output can be scanned in real time for disallowed content, with the system halting generation the moment a policy violation is detected (Milvus, 2025) . This paradigm is used in practice by many providers (OpenAI's and Anthropic's systems have backend filters).

One family of dynamic inference techniques involves guiding or constraining the generation process itself to avoid unsafe trajectories. For example, a language model can be equipped with a capability to backtrack during generation. Instead of producing a problematic answer straight through, the model (or an external controller) could detect an unsafe token sequence as it emerges and revert to a prior state, then try an alternate completion. This idea is analogous to backtracking in search algorithms. While primarily studied to improve reasoning (e.g. the self-backtracking method of Yang et al. lets an LLM revisit earlier reasoning steps when it reaches an impasse (Yang et al., 2025)), the same mechanism could help with safety by treating a looming policy violation as an impasse that triggers a revision. Another test-time strategy is to use multi-pass generation with self-refinement. Instead of one-shot answering, the model might produce a draft response, then examine its own output for compliance, and finally issue a cleaned/refined answer. Anthropic's Constitutional AI approach, for instance, can be run in an inference-time mode where the model first generates an answer and then a self-critique to that answer, revising if the critique finds safety issues (Bai et al., 2022b) . Alternatively, one can run two models in parallel: Wang et al. (2024b) propose SelfDefend, a framework where a secondary "shadow" LLM monitors the main LLM's behavior in lockstep. At certain checkpoints (e.g. end of each user query or each generation chunk), the shadow model evaluates whether the content or intent is disallowed, and can veto or adjust the main model's output.

2.3 Self-Reflection for Reasoning and Safety

A growing body of work shows that allowing an LLM to think step-by-step (Kojima et al., 2022) or otherwise reason with extra computation (Zhou et al.) can dramatically improve its accuracy and factuality. One paradigm is chain-of-thought (CoT) prompting (Wei et al., 2022), where the model is prompted to produce a detailed reasoning trace before giving a final answer. CoT was found to unlock emergent problem-solving abilities in GPT-3 (Brown et al., 2020) and PaLM (Chowdhery et al., 2023), especially for math and logic tasks (e.g. it boosts arithmetic word problem accuracy). Building on this, self-consistency (Wang et al.) decoding samples multiple independent reasoning paths from the model and then selects the answer most frequently reached.

Given the success of multi-step reasoning in correctness, a natural question is whether similar techniques can improve moral and safety reasoning in LLMs. Recently, Zaremba et al. (2025) investigates how increasing inference-time computation affects the resilience of reasoning models, specifically OpenAI's o1-preview and o1-mini, against adversarial attacks. The study finds that allocating more computational resources during inference often decreases the success rate of various adversarial attacks, sometimes reducing it to near zero. However, the paper highlights the emergence of attack vectors specific to reasoning models. One such attack, termed "think less," aims to reduce the model's inference-time computation, thereby increasing its susceptibility to errors. Beside, some recent works adapt the idea of a model giving itself feedback to the domain of alignment. A prime example is Constitutional AI by Anthropic (Bai et al., 2022b), where they train the model with reinforcement learning using its own output for harmful content or policy violations as the reward signal, i.e. 'RL from AI Feedback' (RLAIF). Other researchers have explored integrating a critic module directly into the model. Gallego (2024) proposes merging an external safety critic model into the main LLM's weights to endow it with stronger refusal behavior. Yet another line of work has LLMs generate explicit explanations or justifications for why a request might be disallowed, which has been shown to correlate with more reliable refusals (Turpin et al., 2023; Jin et al., 2022).

3 Proposed Method

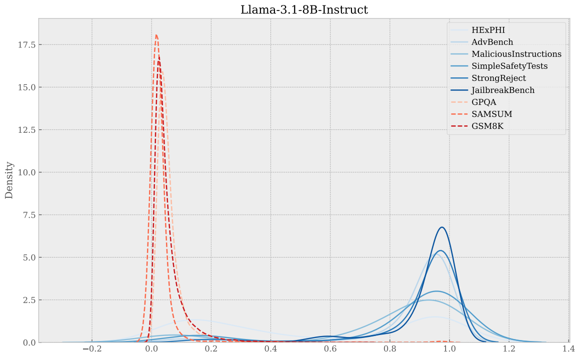
3.1 Preliminaries and Notation

Let $\mathcal{X}$ be the space of token sequences (e.g., sequences of words or subwords). We denote by $x_{1:t} = (x_1, x_2, \ldots, x_t)$ a partial sequence of length t. A typical large language model (LLM) defines an auto-regressive distribution

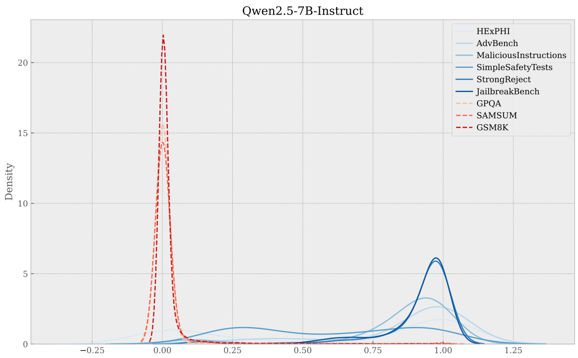
$$p_\theta\left(x_{1:T}\right) = \prod_{t=1}^{T} p_\theta\left(x_t \mid x_{1:t-1}\right), \qquad (1)$$

where θ are the model parameters, and T is the final sequence length (which may itself be random or determined by the model via an end-of-sequence token).

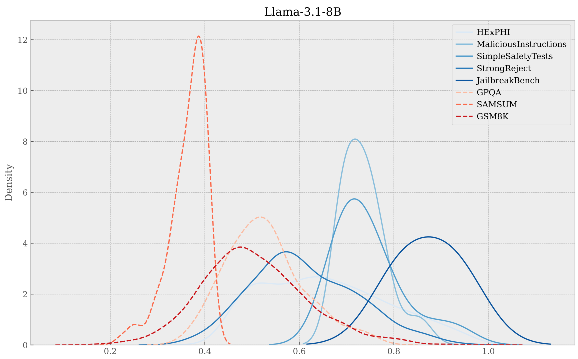
In the standard (greedy or sampling) decoding setup, we generate x_t one token at a time from this distribution $p_\theta\left(x_t \mid x_{1:t-1}\right)$ (greedily or stochastically).



(a) Llama-3.1-8B-Instruct



(b) Qwen2.5-7B-Instruct



(c) Llama-3.1-8B

Figure 2: Kernel density estimates (KDEs) of the normalized harmful probability, computed as $p_\theta\left(w_{\text{harm}} \mid \text{Prompt}\left(x_{1:t}\right)\right) / \left(p_\theta\left(w_{\text{safe}} \mid \text{Prompt}\left(x_{1:t}\right)\right) + p_\theta\left(w_{\text{harm}} \mid \text{Prompt}\left(x_{1:t}\right)\right)\right)$, across various evaluation datasets. Each subplot corresponds to a different language model variant: (a) Llama-3.1-8B-Instruct, (b) Qwen2.5-7B-Instruct, and (c) Llama-3.1-8B (base). Datasets include adversarial, jailbreak, and safety-specific benchmarks (e.g., AdvBench, Jailbreak-Bench, HexPHI), as well as non-adversarial tasks (e.g., GSM8K, SAMSUM) for contrast. Sharp peaks near zero correspond to non-harmful generations, while wider or shifted distributions indicate model uncertainty or increased likelihood of harmful content.

3.2 Self-Reflection Probing

Relection template. We introduce a reflection prompt to query the LLM: $\text{Prompt}\left(x_{1:t}\right) = [$ "Let me check if the generated text is harmless or harmful. The text ", $x_{1:t}$, " is"]. When we feed $\text{Prompt}\left(x_{1:t}\right)$ to the LLM θ, we look specifically at the model's next-token probabilities for the token(s) "harmless" and "harmful." Let w_{safe} represent the token (or token sequence) corresponding to "harmless," w_{harm} represent the token (or token sequence) corresponding to "harmful.", we then obtain the probabilities for the text is harmless or harmful, respectively: $p_\theta\left(w_{\text{safe}} \mid \text{Prompt}\left(x_{1:t}\right)\right)$, $p_\theta\left(w_{\text{harm}} \mid \text{Prompt}\left(x_{1:t}\right)\right)$ Hence we can define a reflection function r_θ purely at inference time:

$$r_\theta\left(x_{1:t}\right) = \begin{cases} \text{"harmless"} & \text{if } p_\theta\left(w_{\text{safe}} \mid \text{Prompt}\left(x_{1:t}\right)\right) \\ & \quad \geq p_\theta\left(w_{\text{harm}} \mid \text{Prompt}\left(x_{1:t}\right)\right) \\ \text{"harmful"} & \text{otherwise} . \end{cases} \tag{2}$$

Here, we do not train or fine-tune the model parameter θ; we only probe the model's internal knowledge to classify the partial text as harmless or harmful.

Periodic Self-Reflection. We pick a set of time steps $\{t_1, t_2, \ldots, t_M\}$ at which we will perform reflection checks (e.g., every $K = 32$ tokens for all of our experiments). Formally, at initialization, let $t = 0$, and $x_0 = \langle \text{ START } \rangle$. Then for $t = 1$ to T :

- Generate x_t by sampling (or greedily picking) from $p_\theta\left(\cdot \mid x_{1:t-1}\right)$.

- If $t \in \{t_1, \ldots, t_M\}$, we form $\text{Prompt}\left(x_{1:t}\right)$ and evaluate:

$$p_\theta\left(w_{\text{safe}} \mid \text{Prompt}\left(x_{1:t}\right)\right)$$
$$p_\theta\left(w_{\text{harm}} \mid \text{Prompt}\left(x_{1:t}\right)\right) .$$

- If $r_\theta\left(x_{1:t}\right) =$ "harmless", continue decoding.

- If $r_\theta\left(x_{1:t}\right) =$ "harmful", backtrack to the most recently known safe prefix. Specifically, let $\kappa(t)$ be the most recent checkpoint index for which the partial sequence was "harmless." We revert the generation to $x_{1:\kappa(t)}$ and re-sample from there (or produce a safe fallback).

Mathematically, once a partial sequence is flagged harmful at a checkpoint, we discard that trajectory by backtracking and overwriting it with a safe prefix. Hence, if we define the final distribution over sequences with reflection as $\tilde{p}_\theta$, it is related to p_θ by:

$$\tilde{p}_\theta\left(x_{1:T}\right)=\prod_{i=1}^{M}\mathbf{1}\left\{r_\theta\left(x_{1:t_i}\right)=\text{ "harmless" }\right\} \times \prod_{t\notin\{t_1,\ldots,t_M\}} p_\theta\left(x_t \mid x_{1:t-1}\right),$$

where the indicator $\mathbf{1}\{\cdot\}$ zeroes out any sequence flagged as harmful at any checkpoint. In practice, we implement zeroing out by forcibly backtracking at runtime.

In Figure 2, we show the distribution of the normalized harmful probability, across a variety of safety and non-safety benchmarks. Notably, the distributions reveal that LLMs are inherently capable of assessing whether their own generated content is harmful or not. For instruction-tuned models like Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct, harmful content is sharply distinguished from harmless content, suggesting that these models have implicitly learned a strong internal representation of harmfulness. Notably, even the base model (Llama-3.1-8B), which has not undergone extensive safety fine-tuning, still performs reasonably well in differentiating between harmful and harmless generated text. This indicates that our self-assessment strategy can effectively leverage the model's internal knowledge to classify partial generations and backtrack or revise them as needed to avoid harmful completions.

Dynamic Self-Reflection for Safe Generation. Our approach dynamically determines the optimal number of self-reflection steps needed to ensure safe text generation. Given an input prompt x, we extract its hidden representation $h(x)\in\mathbb{R}^d$ from the LLM. The self-reflection mechanism is modeled by a function $R(n,x)$ that outputs a binary indicator for the generated text's safety after n reflection steps. We define the minimal reflection count as:

$$n^*(x)=\min\{n\in\{0,1,\ldots,N_{\max}\}\mid R(n,x)=1\},$$

with $n^*(x)=0$ for benign inputs.

To predict $n^*(x)$ from $h(x)$, we train a lightweight MLP $f_{\theta_{\text{MLP}}}:\mathbb{R}^d\to\{0,1,\ldots,N_{\max}\}$. Our training dataset $\mathcal{D}=\{(h(x_i),n^*(x_i))\}_{i=1}^N$ is constructed by sampling from both harmful ($\mathcal{D}_{\text{harmful}}$) and harmless ($\mathcal{D}_{\text{harmless}}$) input sets. For each sample, we simulate the self-reflection process by appending a reflection prompt (e.g., "Let me check if the generated text is harmless or harmful") at fixed token intervals (e.g., every 32 tokens) until harmful content is detected, recording the smallest n that triggers a flag.

The MLP is trained via a mean square error (MSE) loss:

$$\mathcal{L}(\theta_{\text{MLP}})=\frac{1}{N}\sum_{i=1}^{N}\ell\big(f_{\theta_{\text{MLP}}}(h(x_i)),n^*(x_i)\big),$$

which ensures accurate prediction of the optimal reflection count. At inference, the predicted $\hat{n}(x)=f_{\theta_{\text{MLP}}}(h(x))$ governs the dynamic safety assessment, where the model performs the requisite self-reflection steps and backtracks to exclude the reflection tokens from the final output. This framework enables adaptive and efficient safety interventions during generation while preserving performance on benign inputs.

4 Experimental results

In this section, we present experiments to evaluate the effectiveness of our proposed defense method. The evaluations are conducted on a set of benchmarks comprising both harmful and benign prompts, covering both domain-specific and general knowledge tasks.

4.1 Experiment setup

Evaluation focuses on safety violation rates across multiple safety benchmarks, including HExPHI (HP) (Qi et al., 2024), AdvBench (AB) (Contributors, 2024a), MaliciousInstructions (MI) (Contributors, 2024c), SimpleSafetyTests (ST) (), StrongReject (SR), Trivial Jailbreak (TJ), JailbreakBench (JB), and Natural Language Game Attack (NL). Besides, we show how our methods can help defense against well-established jailbreak attack methods: GCG (Zou et al., 2023), AutoDAN (Liu et al., 2023a), PAIR (Chao et al., 2023), ReNeLLM (Ding et al., 2023), CodeChameleon (Lv et al., 2024), DeepInception (Li et al., 2023a), ICA (Wei et al., 2023b) and MSJ (Anthropic, 2024). Additionally, we assess accuracy using standard benchmarks such as SamSum (SS), GSM8K (OpenAI, 2021), GPQA (Contributors, 2024b), and MMLU (Contributors, 2021) to ensure that the safety mechanisms do not compromise the model's performance.

We conducted experiments using the following open-source LLM base models: Llama-3.1-8B, Llama-3.1-8B-Instruct (Touvron et al., 2023), and Qwen2.5-7B-Instruct (Yang et al., 2024). For each model, we assessed the potential impact of jailbreak techniques on benign users by measuring the

Model	Method	HP ↓	AB ↓	TJ	MI ↓	SST ↓	SR ↓	NL ↓	JB ↓	SS ↑	GSM8K ↑	GPQA ↑	MMLU ↑
Llama-3.1-8B	ZS	89.39	96.15	79.33	92.33	90.33	87.75	99.60	96.00	13.23	–	–	–
	N=1	14.04	16.79	5.33	31.00	31.00	16.40	11.38	24.33	13.64	–	–	–
	N=2	10.10	16.47	5.00	26.33	28.67	11.71	6.14	20.00	13.13	–	–	–
	N=4	6.87	16.22	5.00	26.00	27.67	10.76	3.42	19.67	**17.52**	–	–	–
	N=8	5.56	**16.15**	**2.00**	26.33	**25.33**	9.58	1.81	19.33	17.89	–	–	–
	N=-1	**5.45**	**16.15**	**2.00**	**24.00**	27.00	**8.95**	**1.31**	**19.00**	17.71	–	–	–
Llama-3.1-8B Instruct	ZS	77.47	0.83	49.00	1.33	7.00	6.07	88.62	1.00	31.48	79.82	28.04	60.80
	N=1	11.11	0.58	2.00	0.67	1.00	0.43	85.20	**0.00**	31.70	79.33	28.66	60.92
	N=2	9.85	**0.48**	1.00	1.00	2.00	0.32	81.87	**0.00**	31.32	79.22	28.00	61.01
	N=4	7.27	0.51	**0.00**	0.67	0.67	0.32	73.87	0.00	31.47	78.84	27.15	60.00
	N=8	6.57	0.51	**0.00**	0.67	0.67	0.32	60.27	0.00	31.87	81.67	27.34	60.92
	N=-1	**5.86**	0.51	**0.00**	0.33	**0.00**	0.32	46.22	**0.00**	31.68	80.69	28.08	61.19
Qwen2.5-7B Instruct	ZS	44.44	0.96	11.33	6.67	11.00	6.18	95.77	10.00	26.26	58.83	20.24	27.83
	N=1	6.77	0.83	**0.00**	6.00	4.67	2.13	93.15	8.33	26.50	58.52	20.71	27.62
	N=2	5.15	0.96	**0.00**	5.00	4.00	2.24	92.95	9.00	26.71	58.75	20.03	27.71
	N=4	4.34	0.90	**0.00**	4.67	4.67	2.02	92.55	5.67	26.68	58.96	20.98	28.04
	N=8	3.84	0.83	**0.00**	5.33	5.00	**1.70**	91.64	**5.33**	26.43	59.79	19.74	27.90
	N=-1	**3.23**	**0.77**	**0.00**	**5.33**	**4.33**	2.02	**84.79**	5.67	26.25	57.23	20.33	27.63

Table 1: **Progressive Self-Reflection (PSR) enhances generation safety.** We report safety violation rates (%) across four sources of safety prompts: HExPHI (HP), AdvBench (AB), MaliciousInstructions (MI), SimpleSafetyTests (ST), StrongReject (SR), Trivial Jailbreak (TJ), JailbreakBench (JB), Natural Language Game Attack (NL), and the accuracy metrics SamSum (SS), GSM8K, GPQA, MMLU. Best results for each base model are in **bold**. N denotes the number of self-reflection rounds and N=-1 indicates reflect until the end of sequences. ZS represents zero-shot (naive greedy decoding) baseline. Results are averaged over three random seeds.

models' refusal rates. Additionally, we evaluated utility metrics pertinent to benign fine-tuning scenarios, employing the standard ROUGE-1 score for the SamSum dataset and answer exact string matching accuracy for GSM8K, GPQA, and MMLU benchmarks.

4.2 Results

Table 1 summarizes the impact of our self-reflection mechanism on three open-source LLMs-Llama-3.1-8B, Llama-3.1-8B-Instruct, and Qwen2.5-7B-Instruct-across multiple safety benchmarks and utility metrics. The rows list different configurations, including zero-shot (ZS) and varying numbers of self-reflection steps (N=1, N=2, etc.). For safety, we report violation rates on benchmarks such as HExPHI (HP), AdvBench (AB), Trivial Jailbreak (TJ), and MaliciousInstructions (MI). For utility, we measure performance on GPQA, MMLU, and other standard tasks. Lower values in safety benchmarks indicate fewer violations (i.e., better safety), whereas higher scores on utility metrics reflect stronger task performance.

Overall, increasing the number of self-reflection checkpoints (N) reduces attack sucess rates across all three models. Particularly for Instruct variants, the drop in violation rates is more significant, suggesting these models benefit substantially from the additional safety layer thanks to their ability to assess their own generation. For Llama-3.1-8B, the zero-shot baseline exhibits high violation rates (e.g., HP: 89.39%, AB: 96.15%, JB: 96.00%). For most settings, improvements in safety come with minimal or no drop in performance on SamSum, GSM8K, GPQA, and MMLU. We hypothesize the difference in that utility performance is due to randomness, where we can sometimes even observe improvement in utility. Since the base model cannot follow the instruction for the answer format on GSM8K, GPQA, and MMLU, their performance is unstable across random seeds. We thus do not report those results.

Table 2 reports the attack success rates of Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct - under eight representative jailbreak methods. In the greedy decoding condition, Llama-3.1-8B-Instruct is highly vulnerable, with average success rates exceeding 70% on GCG and AutoDAN and above 80% on DeepInception, whereas Qwen2.5-7B-Instruct already shows substantially lower baselines (e.g., 43.5% on GCG, 27.0% on AutoDAN). Introducing iterative self-reflection steps (N = 1, 2, 4, 8) yields a consistent, near-monotonic decline in attack efficacy for both models. Notably, by N = 8, Llama-3.1-8B-Instruct's success rates drop below 30% across all methods and reach 0% for ICA and MSJ, while Qwen2.5-7B-Instruct falls below 5% on nearly all attacks and is completely immune (0%) to four of the eight methods. The N = –1 configuration-representing an unbounded or convergence-based reflection-provides marginal additional gains, suggesting diminishing returns beyond eight iterations. These trends underscore that-even against diverse and adaptive jail-

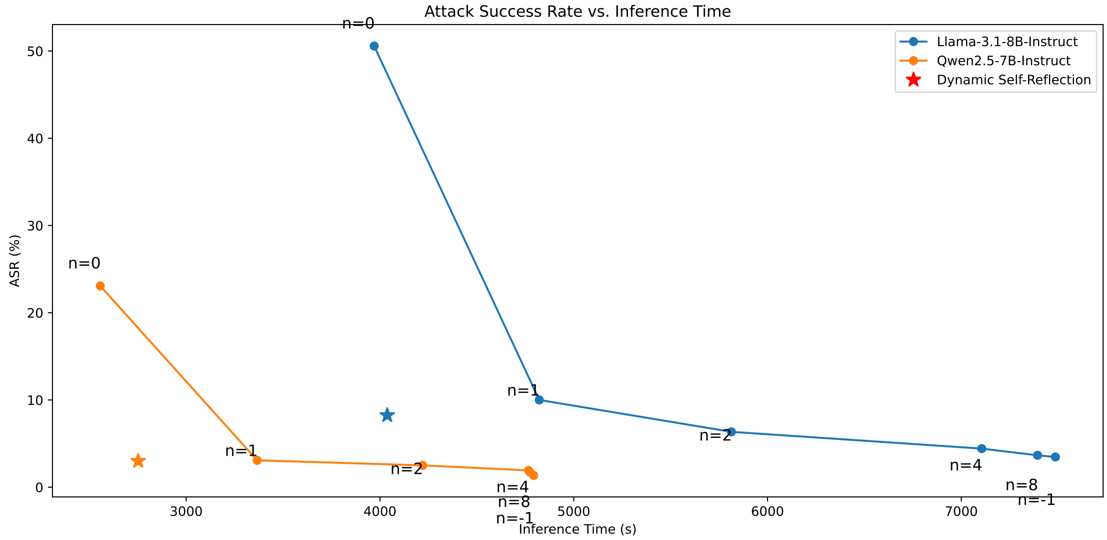


Figure 3: **Attack success rate (ASR) on AdvBench prefilling attack and inference time spent on benign SamSum dataset** for Llama-3.1-8B-Instruct (blue) and Qwen2.5-7B-Instruct (orange) under varying numbers of self-reflection rounds (n). As n increases, the models exhibit a substantial drop in ASR-indicating greater robustness to adversarial prompts-at the cost of a notable rise in inference time.

break strategies-iterative self-monitoring dramatically fortifies model safety, and that models like Qwen2.5-7B-Instruct combine inherent robustness with amplified benefits from reflective defenses.

Model	Method	GCG ↓	AutoDAN ↓	PAIR ↓	ReNeLLM ↓	CodeChameleon ↓	DeepInception ↓	ICA ↓	MSJ ↓
Llama-3.1-8B Instruct	ZS	73.86	72.88	28.57	80.48	96.44	86.60	49.62	48.63
	N=1	33.80	4.04	26.53	65.76	92.31	67.60	0.00	0.26
	N=2	28.45	1.35	26.53	51.36	91.35	55.10	0.00	0.26
	N=4	26.53	0.19	24.48	40.99	80.64	38.72	0.00	0.26
	N=8	25.02	0.00	22.45	30.48	70.71	32.69	0.00	0.26
	N=-1	15.77	0.00	18.37	23.93	60.52	29.10	0.00	0.26
Qwen2.5-7B Instruct	ZS	43.48	27.00	36.73	47.21	93.27	88.65	8.72	36.15
	N=1	5.49	1.00	25.51	16.21	67.56	3.40	0.00	14.36
	N=2	4.97	1.00	23.47	13.42	60.71	1.86	0.00	11.79
	N=4	3.62	1.00	23.13	10.89	55.45	1.15	0.00	9.23
	N=8	3.42	1.00	22.45	10.05	35.58	0.71	0.00	1.68
	N=-1	3.36	1.00	20.41	9.86	30.19	0.58	0.00	8.38

Table 2: **Performance against jailbreaking methods** We report the attack success rate of Llama-3.1-8B Instruct and Qwen2.5-7B Instruct against jailbreak attack methods: GCG (Zou et al., 2023), AutoDAN (Liu et al., 2023a), PAIR (Chao et al., 2023), ReNeLLM (Ding et al., 2023), CodeChameleon (Lv et al., 2024), DeepInception (Li et al., 2023a), ICA (Wei et al., 2023b) and MSJ (Anthropic, 2024)

4.3 Amortize the number of reflection rounds

Similar to what we discuss above, the trade-off in figure 3 highlights a key challenge in designing safe and scalable LLM systems for real-world applications: While additional reflection checkpoints reinforce the model's ability to detect and mitigate harmful content, they also introduce computational overhead. Identifying an optimal balance between safety and efficiency remains an open problem for ML practitioners where the number of reflection rounds should be tuned based on preference (either prioritizing efficiency or safety). We thus present a simple and straight-forward Dynamic Self-Reflection strategy that estimate the needed reflection rounds. For both models, our dynamic self-reflection machenisms (indicated by star markers) lie strictly on the lower-left Pareto frontier of the ASR–latency plot, meaning they dominate every fixed-N configuration. For Llama-3.1-8B-Instruct,our method achieves only 8% attack success in 4000s, whereas the best static scheme (N=1) still needs 4822s to hit 10% ASR. Likewise, for Qwen2.5-7B-Instruct, our adaptive rule drives ASR below 3% in just 2752s, while even N=2 takes nearly 4220s to reach the same safety level. These results confirm that dynamic scaling not only reduces vulnerability more effectively but also cuts inference overhead, yielding a strictly superior Pareto trade-off.

5 Conclusion

In this paper, we introduce Progressive Self-Reflection (PSR), a decoding-time defense that significantly reduces jailbreak attacks on large language models (LLMs). By enabling dynamic self-assessment during text generation and employing an adaptive predictor for reflection rounds, PSR efficiently balances computational overhead with safety. Experiments on frontier open-source LLMs demonstrate that PSR reduces jailbreak success rates significantly while maintaining their original task performance without additional training. Our results underline PSR's practicality and effectiveness as a scalable, adaptive approach to safer LLM deployment.

6 Limitations

While Progressive Self-Reflection (PSR) offers a powerful, training-free defense against jailbreak attacks, it also carries several notable limitations:

Inference-Time Overhead. PSR interleaves generation with periodic self-reflection checkpoints, which inevitably lengthens decoding time. As shown in our experiments, increasing the number of reflection rounds (N) yields diminishing returns in safety beyond a certain point but continues to incur extra latency. Finding the right balance between safety and responsiveness remains an open challenge, especially for real-time or cost-sensitive applications.

Dependence on a Binary "Harmful / Harmless" Classifier. At each rounds, PSR simply compares the probability of "harmless" versus "harmful" to decide whether to backtrack. This coarse binary decision may struggle with nuanced content-benign text could be misclassified as harmful (triggering unnecessary backtracking and reduced fluency), while cleverly crafted adversarial inputs might evade detection if they exploit subtle model blind spots.

Need for an Auxiliary Predictor and Hyperparameter Tuning. To avoid uniform over-reflection, PSR employs a lightweight MLP to predict the minimal number of rounds needed per input. Training this predictor requires a curated dataset of harmful versus benign prompts, along with simulation of the reflection process. Moreover, the token-interval and maximum rounds are hyperparameters that must be tuned, potentially requiring additional development effort.

Eventhough, our key contribution is to demonstrate that a simple test-time scaling strategy can substantially enhance the robustness of large language models with almost no extra cost. By inspecting each layer's activations at inference, our method provides an efficient, low-overhead safeguard against adversarial prompts and distribution shifts. While this straightforward approach already yields consistent improvements, we acknowledge that more sophisticated, adaptive scaling schemes-or entirely different calibration techniques, may further optimize the trade-off between robustness and efficiency. We leave the exploration of these richer, potentially highercomplexity defenses to future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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

Anthropic. 2024. Many-shot jailbreaking. https://www-cdn.anthropic.com/af5633c94ed2beb282f6a53c595eb437e8e7b630/Many_Shot_Jailbreaking__2024_04_02_0936.pdf. Accessed: 2025-05-20.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Leyla Naz Candogan, Yongtao Wu, Elias Abad Rocamora, Grigorios G Chrysos, and Volkan Cevher. 2025. Single-pass detection of jailbreaking input in large language models. *arXiv preprint arXiv:2502.15435*.

He Cao, Weidi Luo, Yu Wang, Zijing Liu, Bing Feng, Yuan Yao, and Yu Li. 2024. Guide for defense (g4d): Dynamic guidance for robust and balanced defense in large language models. *arXiv preprint arXiv:2410.17922*.

Nicholas Carlini. 2024. Some lessons from adversarial machine learning. https://www.youtube.com/watch?v=umfeF0Dx-r4. YouTube video.

Zhiyuan Chang, Mingyang Li, Yi Liu, Junjie Wang, Qing Wang, and Yang Liu. 2024. Play guessing game with llm: Indirect jailbreak attack with implicit clues. *arXiv preprint arXiv:2402.09091*.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.

Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, and 1 others. 2023. Meditron-70b: Scaling medical pretraining for large language models. *arXiv preprint arXiv:2311.16079*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

AdvBench Contributors. 2024a. Advbench dataset.

GPQA Contributors. 2024b. Gpqa: Graduate-level google-proof qa benchmark.

MaliciousInstructions Contributors. 2024c. Maliciousinstructions dataset.

MMLU Contributors. 2021. Measuring massive multitask language understanding.

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

Yi Dong, Ronghui Mu, Yanghao Zhang, Siqi Sun, Tianle Zhang, Changshun Wu, Gaojie Jin, Yi Qi, Jinwei Hu, Jie Meng, Saddek Bensalem, and Xiaowei Huang. 2024. Safeguarding large language models: A survey. *CoRR*, abs/2406.02622.

Víctor Gallego. 2024. Merging improves self-critique against jailbreak attacks. *CoRR*, abs/2406.07188.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho. 2024. Gradient cuff: Detecting jailbreak attacks on large language models by exploring refusal loss landscapes. *arXiv preprint arXiv:2403.00867*.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. 2024. Harmful fine-tuning attacks and defenses for large language models: A survey. *arXiv preprint arXiv:2409.18169*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

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

Zhijing Jin, Sydney Levine, Fernando Gonzalez Adauto, Ojasv Kamal, Maarten Sap, Mrinmaya Sachan, Rada Mihalcea, Josh Tenenbaum, and Bernhard Schölkopf. 2022. When to make exceptions: Exploring language models as accounts of human moral judgment. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Dmitry Kobak, Rita González-Márquez, Emőke-Ágnes Horvát, and Jan Lause. 2024. Delving into chatgpt usage in academic writing through excess vocabulary. *arXiv preprint arXiv:2406.07016*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Jean Lee, Nicholas Stevens, and Soyeon Caren Han. 2025. Large language models in finance (finllms). *Neural Computing and Applications*, pages 1–15.

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

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023b. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 374–382.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, and 1 others. 2024a. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024b. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

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

Xiaoqun Liu, Jiacheng Liang, Muchao Ye, and Zhaohan Xi. 2024c. Robustifying safety-aligned large language models through clean data curation. *arXiv preprint arXiv:2405.19358*.

Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, Kailong Wang, and Yang Liu. 2023b. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*.

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

Milvus. 2025. What role do llm guardrails play in content moderation?

OpenAI. 2021. Gsm8k: A dataset of grade school math word problems.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2024. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *The Twelfth International Conference on Learning Representations*.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2024. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1671–1685.

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

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, and 1 others. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Xiongtao Sun, Deyue Zhang, Dongdong Yang, Quanchen Zou, and Hui Li. 2024. Multi-turn context jailbreak attack on large language models from first principles. *arXiv preprint arXiv:2408.04686*.

Xiaoyu Tan, Shaojie Shi, Xihe Qiu, Chao Qu, Zhenting Qi, Yinghui Xu, and Yuan Qi. 2023. Self-criticism: Aligning large language models with their understanding of helpfulness, honesty, and harmlessness. In *Proceedings of the 2023 conference on empirical methods in natural language processing: industry track*, pages 650–662.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Neeraj Varshney, Pavel Dolin, Agastya Seth, and Chitta Baral. 2023. The art of defending: A systematic evaluation and analysis of llm defense strategies on safety and over-defensiveness. *arXiv preprint arXiv:2401.00287*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S Yu, and Qingsong Wen. 2024a. Large language models for education: A survey and outlook. *arXiv preprint arXiv:2403.18105*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Xunguang Wang, Daoyuan Wu, Zhenlan Ji, Zongjie Li, Pingchuan Ma, Shuai Wang, Yingjiu Li, Yang Liu, Ning Liu, and Juergen Rahmel. 2024b. Selfdefend: Llms can defend themselves against jailbreaking in a practical manner. *CoRR*, abs/2406.05498.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, and 1 others. 2021. Ethical and social risks of harm from language models. `https://arxiv.org/abs/2112.04359`.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 3(4).

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

Chen Xiong, Xiangyu Qi, Pin-Yu Chen, and Tsung-Yi Ho. 2024. Defensive prompt patch: A robust and interpretable defense of llms against jailbreak attacks. *arXiv preprint arXiv:2405.20099*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Dingchu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yufeng Li. 2025. Step back to leap forward: Self-backtracking for boosting reasoning of language models. *CoRR*, abs/2502.04404.

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

Lei Yu, Virginie Do, Karen Hambardzumyan, and Nicola Cancedda. 2024. Robust llm safeguarding via refusal feature adversarial training. *arXiv preprint arXiv:2409.20089*.

Wojciech Zaremba, Evgenia Nitishinskaya, Boaz Barak, Stephanie Lin, Sam Toyer, Yaodong Yu, Rachel Dias, Eric Wallace, Kai Xiao, Johannes Heidecke, and Amelia Glaese. 2025. Trading inference-time compute for adversarial robustness. *CoRR*, abs/2501.18841.

Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. 2024. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–20.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and 1 others. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. 2023. Autodan: interpretable gradient-based adversarial attacks on large language models. *arXiv preprint arXiv:2310.15140*.

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

A Appendix

Due to space constraints, some details were omitted from the main paper. We therefore include the detailed experiment setup description and additional experimental results in this appendix.

B Hardware configuration

All experiments were conducted on high-performance machines equipped with Intel Xeon CPUs and NVIDIA GPUs, selected to accommodate varying computational needs and optimize job priority scheduling across different tasks. Specifically, we utilized three machine configurations: (1) Intel Xeon Platinum 8268 @ 2.90GHz with 377 GiB RAM and an NVIDIA Tesla V100-PCIE-32GB GPU, (2) Intel Xeon Platinum 8268 @ 2.90GHz with 377 GiB RAM and an NVIDIA Quadro RTX 8000 (48GB), and (3) Intel Xeon Platinum 8380 @ 2.30GHz with 1.0 TiB RAM and an NVIDIA A100-SXM4-80GB GPU. Although different GPU types were used to balance workload priorities, we ensured that all running comparisons across inference strategies were performed on the same hardware configuration for a given model and dataset to eliminate hardware-induced variability and maintain consistency and fairness in evaluation.

B.1 Experimental details

We evaluate safety and utility on a broad mix of adversarial "jailbreak" benchmarks and standard NLP tasks. Our safety evaluation employs HExPHI, a harmful-prefix injection benchmark probing LLMs' detection of malicious prefixes; AdvBench, a curated adversarial set of harmful-behavior prompts; MaliciousInstructions, a crowd-sourced collection of explicitly malicious instructions; SimpleSafetyTests, a suite of synthetic refusal-eliciting prompts; StrongReject, a high-difficulty policy-violation benchmark; Trivial Jailbreak [1], that trivially get around LLMs safety efforts by simply "priming" the model to produce a harmful response; JailbreakBench, a comprehensive collection of varied attack strategies; and Natural Language Game Attack, which uses "game" prompts

[1] https://github.com/haizelabs/llama3-jailbreak

to bypass safety checks. To ensure that safety interventions do not degrade core capabilities, we also report performance on standard tasks: SamSum (SMS-conversation summarization), GSM8K (grade-school math problems), GPQA (graduate-level QA), and MMLU (multi-task language understanding).

B.2 Baseline Descriptions

We compare our Progressive Self-Reflection (PSR) against three inference-time strategies. First, Zero-Shot (ZS) uses naïve greedy decoding without any self-reflection or safety checks. Second, Static PSR performs periodic self-reflection every K = 32 tokens for a fixed number N of rounds-specifically $N \in \{1, 2, 4, 8\}$ plus an unbounded variant (N = –1)-backtracking whenever an internal classifier flags a harmful generation or eos token encounterd. Third, Dynamic PSR employs a lightweight MLP predictor $f_{\theta_{\text{MLP}}}$ to analyze the model's hidden representation h(x) and dynamically estimate the minimal number of reflection rounds needed per example, thereby adapting overhead on the fly.

C Attacking methods

We utilize the EasyJailbreak[2] library that integrates nine distinct adversarial strategies-ranging from discrete token optimization to demonstration-based exploits-to probe different facets of LLM safety and robustness. The Greedy Coordinate Gradient (GCG) attack uses discrete token-level optimization by iteratively selecting and updating individual tokens to maximize the likelihood of a successful jailbreak response. AutoDAN employs a hierarchical genetic algorithm to automatically evolve stealthy jailbreak prompts through selection, crossover, and mutation at both sentence and paragraph levels. PAIR (Prompt Automatic Iterative Refinement) uses an attacker LLM to iteratively refine and update candidate jailbreak prompts against a target model in a black-box setting, often requiring fewer than twenty queries. ReNeLLM generalizes jailbreak attacks by leveraging LLMs themselves to perform prompt rewriting and scenario nesting, crafting versatile, context-adapted exploit prompts. CodeChameleon reframes malicious instructions as personalized encrypted code-completion tasks, embedding decryption routines to bypass intent-security recognition. DeepInception draws on authoritative framing and hypnotic language inspired by psychological obedience experiments to "incept" the model into executing harmful instructions with minimal overhead. The In-Context Attack (ICA) directly injects harmful demonstrations into the prompt, exploiting in-context learning to bias the model toward unsafe completions. Many-Shot Jailbreaking (MSJ) leverages extremely long context windows by providing hundreds of harmful examples within the prompt, inducing the model to generalize unsafe behavior at scale. Finally, AdvBench offers a structured benchmark of adversarial prompts-malicious strings and harmful behavior instructions-designed to systematically evaluate the breadth and depth of LLM jailbreaking vulnerabilities.

[2]https://github.com/EasyJailbreak/EasyJailbreak

C.1 Hyperparameter Settings

Across all PSR experiments, we fix the self-reflection interval K to 32 tokens and consider static reflection rounds $N \in \{1, 2, 4, 8\}$ (plus an unlimited variant). The Dynamic Self Predictor is a small three-layer MLP, trained using an MSE loss between its prediction $f_{\theta_{\text{MLP}}}$ and the true optimal number of rounds $n^*(x)$ on a mix of samples from MaliciousInstruct, AutoDAN, GPQA, GSM8k, GCG. Please note that the dataset we use in Figure 3 is the 10-token prefilling attack on AdvBench, which is an out-of distribution dataset that we do not use to train the MLP. Decoding is performed greedily at temperature = 0, with max generated tokens is 512 for jailbreak experiments and 1024 for utility evaluation, and each configuration is run with three random seeds to ensure stability. Detailed model architecture, optimizer settings, and training schedules for the MLP predictor are provided in our code release.

D Additional experimental results

D.1 Last generated token representation

Figure 4 presents a t-SNE projection of the final token representation from model outputs across various datasets, with marker shapes indicating the dataset-SamSum (benign), AdvBench and SimpleSafetyTests (harmful prompts, though the model generally produces safe responses), StrongReject, and HExPHI (malicious prefixes). The color scale represents the number of self-reflection rounds (0 to 4). Notably, even though AdvBench and SimpleSafetyTests are adversarial, the model manages to avoid harmful completions for these prompts, whereas HExPHI can still compromise the model when prefilled, resulting in a distinct clustering pattern. As reflection rounds increase (shifting from dark to light hues), the tokens move toward "safer" regions, underscoring how the representation of the last generated token can reliably indicate the harmfulness of generated text-and how iterative self-reflection helps reduce harmful outputs.

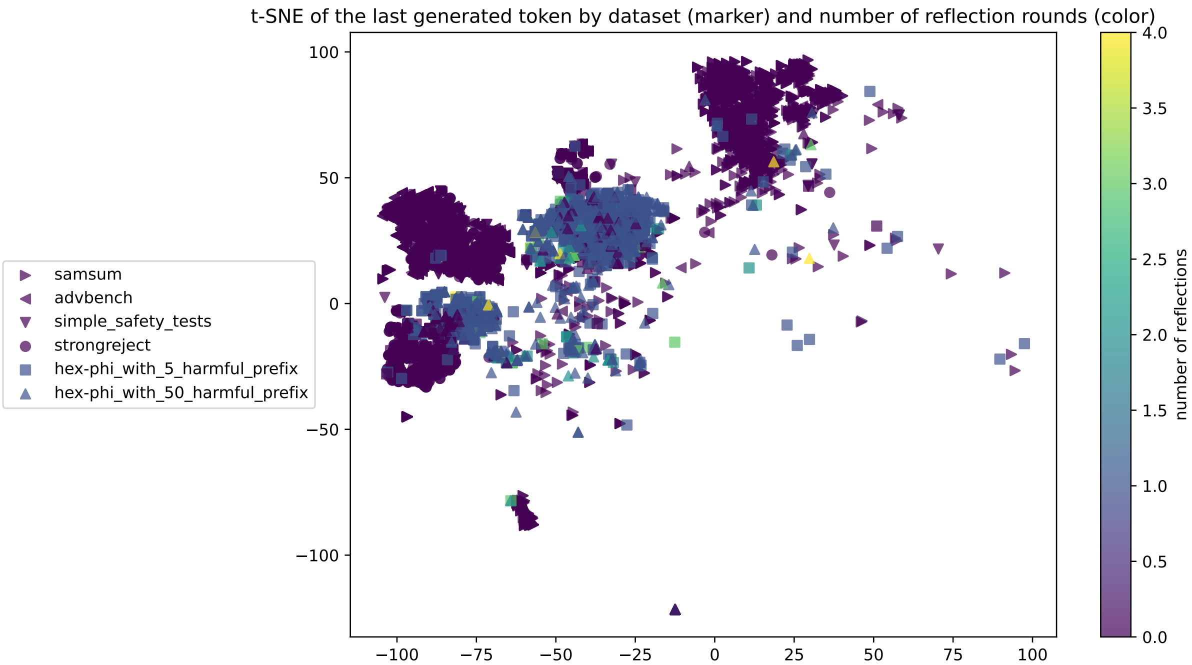


Figure 4: **t-SNE of the Last Generated Token by Dataset** Different markers denote the dataset (e.g., SamSum, AdvBench, SimpleSafetyTests), while the color scale indicates the number of self-reflection rounds (from 0 to 4). .