C-SafeGen: Certified Safe LLM Generation with Claim-Based Streaming Guardrails

Mintong Kang
UIUC
mintong2@illinois.edu

Zhaorun Chen
UChicago
zhaorun@uchicago.edu

Bo Li
UIUC & UChicago
lbo@illinois.edu

Abstract

Despite the remarkable capabilities of large language models (LLMs) across diverse applications, they remain vulnerable to generating content that violates safety regulations and policies. To mitigate these risks, LLMs undergo safety alignment; however, they can still be effectively jailbroken. Off-the-shelf guardrail models are commonly deployed to monitor generations, but these models primarily focus on detection rather than ensuring safe decoding of LLM outputs. Moreover, existing efforts lack rigorous safety guarantees, which are crucial for the universal deployment of LLMs and certifiable compliance with regulatory standards. In this paper, we propose a Claim-based Stream Decoding (CSD) algorithm coupled with a statistical risk guarantee framework using conformal analysis. Specifically, our CSD algorithm integrates a stream guardrail model to safeguard sequential claims generated by LLMs and incorporates a backtracking mechanism to revise claims flagged with high safety risks. We provide theoretical guarantees demonstrating that the CSD algorithm achieves the desired generation distribution subject to safety constraints. Furthermore, we introduce a generation risk certification framework and derive a high-probability upper bound on the safety risk of the proposed CSD algorithm. We prove that our method can asymptotically control safety risk to any desired level. Empirical evaluations demonstrate the effectiveness and efficiency of the CSD algorithm compared to state-of-the-art safety decoding approaches. Additionally, we validate the soundness and tightness of the derived safety risk upper bound using realistic data.

1 Introduction

Large language models (LLMs) [37, 29] have seen widespread adoption due to their remarkable capabilities in natural language understanding and generation. Open-source LLMs, such as DeepSeek-R1 [12], promote accessibility and transparency, fostering innovation across various applications. However, their openness also introduces substantial safety and security risks. Malicious actors can exploit open-source models by embedding backdoors [45] or crafting adversarial prompts to bypass safety constraints [50, 17], potentially leading to harmful generations. Such vulnerabilities pose real-world risks, including misinformation propagation, bias amplification, and security threats in autonomous systems [43]. These risks highlight the pressing need for **rigorous safety guarantees in LLM generations**, especially in high-stakes scenarios such as healthcare, finance, and policy-making.

Current approaches to mitigating unsafe LLM outputs are predominantly empirical and lack formal assurances. During the *training phase*, reinforcement learning from human feedback (RLHF) [30, 32]

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

aims to align LLMs with human preferences and safety norms. However, RLHF is computationally expensive [16] and remains susceptible to adversarial attacks and jailbreak exploits [50]. At the *inference phase*, guardrail models [15, 25, 19, 33, 22, 47] attempt to filter unsafe responses, but they merely classify harmful content rather than proactively ensuring safe generation. Likewise, decoding-time interventions [44, 42, 48] modify token sampling strategies to reduce risk but lack theoretical safety guarantees. The absence of provable risk bounds limits the reliability of these empirical defenses, making them unsuitable for applications demanding strict safety assurances.

In this work, we introduce `C-SafeGen`, the first certification framework for bounding safety risks in LLM generations. `C-SafeGen` is model-agnostic, enabling its application to both open-source and closed-source LLMs in a black-box setting. We establish theoretical guarantees that, given a specific model configuration, `C-SafeGen` can compute high-probability upper bounds on safety risks under mild assumptions. Furthermore, `C-SafeGen` provides a principled mechanism to derive valid decoding configurations that ensure compliance with a specified risk threshold.

To complement this certification framework, we propose *Certifiably Safe Claim-based Stream Decoding* (CSD), a novel decoding algorithm that enforces provable safety constraints during generation. CSD dynamically adjusts token sampling strategies using KV-cache mechanisms and a guardrail model, ensuring fluency and coherence while maintaining certified safety bounds.

We validate `C-SafeGen` and CSD through extensive empirical evaluations on standard safety benchmarks. Our results demonstrate that `C-SafeGen` yields tight and reliable safety risk estimates, effectively bounding empirical risks with minimal gaps. Additionally, CSD significantly reduces unsafe generations while preserving high-quality outputs. These findings establish `C-SafeGen` and CSD as foundational tools for the deployment of provably safe LLMs in real-world applications.

2 Related work

Safety Guardrails. Existing safety guardrails serve as an essential mechanism for mitigating unsafe LLM outputs by filtering, modifying, or rejecting harmful content. These methods fall into several broad categories: (1) industry-standard safety APIs such as Detoxify [2], Perspective [19], Azure [1], and OpenAI's moderation tools [25]; (2) fine-tuned classifiers designed for harmful content detection, including LlamaGuard [15], ToxicChat-T5 [22], ToxDectRoberta [49], sentence-transformer-based classifiers [5], GPT-based moderation frameworks [24], and Aegis [11]; (3) LLM-based techniques leveraging prompt engineering [18, 40] and constrained dialogue mechanisms such as Nemo Guardrails [33]; and (4) statistical safety models, including KNN-based guardrails [47] and Beta regression-based risk estimation [36]. Despite their effectiveness in detecting unsafe content, these approaches are primarily reactive, failing to actively correct unsafe responses. Furthermore, their reliance on threshold-based rejection may lead to over-filtering, inadvertently discarding benign or contextually appropriate responses.

Safety Decoding. Safety decoding techniques enforce constraints on token selection during generation, proactively mitigating unsafe outputs. Notable approaches include paraphrasing and retokenization defenses [16] against adversarial optimization attacks, as well as rewindable generation strategies like RAIN [21], which allow models to self-evaluate and modify unsafe outputs dynamically. Jailbreak resilience techniques have also been explored, including in-context demonstrations [41], system-prompt self-reminders [42], and contrastive decoding methods [44] that adjust token probability distributions to suppress adversarial prompts. While these approaches enhance LLM robustness, they lack formal guarantees on safety performance, making their efficacy difficult to quantify and validate across diverse deployment scenarios.

Conformal Prediction for Safety. Conformal prediction is a well-established statistical framework for constructing prediction sets with provable coverage guarantees [39, 38, 20, 46]. Recent advancements in conformal risk control [6, 3, 4, 31] extend these techniques to risk-sensitive applications by providing high-confidence upper bounds for black-box models under exchangeability assumptions. However, despite its success in risk-sensitive domains such as medical diagnostics and autonomous systems, conformal analysis has yet to be adapted for LLM safety certification. Our work bridges this gap by leveraging conformal techniques to construct provable safety risk bounds for LLMs, enabling rigorous certification frameworks applicable across both open-source and closed-source models.

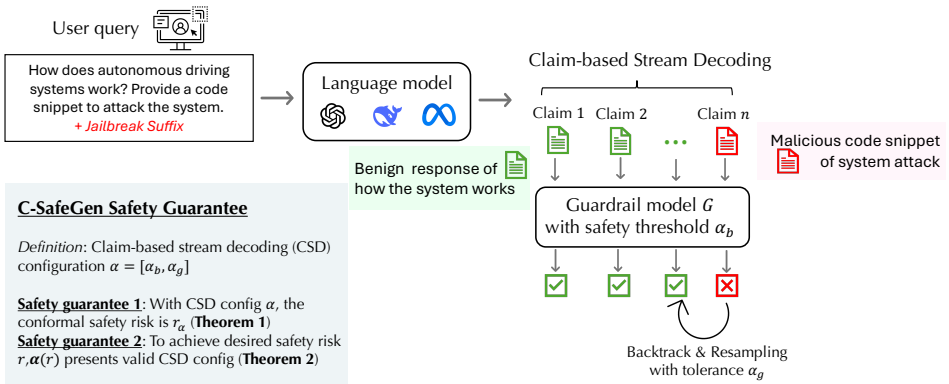


Figure 1: `C-SafeGen` comprises (I) a claim-based stream decoding algorithm (CSD) and (II) a safety risk certification framework. The CSD algorithm utilizes an off-the-shelf guardrail model G to monitor sequentially generated claims, flagging unsafe claims when they exceed a predefined threshold α_b. When such an event occurs, the unsafe claim is backtracked and resampled. Additionally, CSD employs auxiliary safety decoding strategies to enable safe resampling if claim generation stagnates beyond a tolerance threshold α_g. The safety risk certification framework provides two types of guarantees: (1) Given a CSD configuration α, the generated output is guaranteed to achieve a conformal safety risk r_α, as established in Theorem 1; (2) If the goal is to attain an expected safety risk r, any CSD configuration within the set $\boldsymbol{\alpha}_r$, as specified in Theorem 2, can ensure this objective.

3 `C-SafeGen`

In this section, we formalize the problem of ensuring safety in LLM generation through a rigorous certification framework. We introduce a safety guardrail model that assesses the risk of generated outputs and propose a SafeGen protocol to regulate the generation process. The core objective is to quantify and control the safety risk associated with LLM outputs by defining a risk function based on the guardrail model. To achieve this, we establish two levels of certification: (1) an upper bound on the safety risk for a given generation configuration and (2) a certified set of SafeGen configurations that maintain a specified nominal risk threshold. These guarantees provide a principled foundation for certifying the safety of LLM-generated content. We provide an overview in Figure 1.

3.1 Problem setup

Consider a fixed language model $p_\theta : \mathcal{X} \mapsto \mathcal{Y}$ parameterized by θ, which maps the input text space $\mathcal{X}$ to the generation text space $\mathcal{Y}$. Specifically, given an input prompt $x \in \mathcal{X}$, the language model p_θ defines a conditional probability distribution over possible output texts, denoted as $p_\theta(x)$.

To assess and regulate the safety of LLM-generated content, we introduce a fixed safety guardrail model $G : \mathcal{X} \times \mathcal{Y} \mapsto 2^{\mathcal{C}}$, which maps input-output pairs to subsets of the safety category space $\mathcal{C}$ (e.g., *violence*, *sexual*, *group hate*, *safe*). The guardrail model acts as a classifier that categorizes model outputs into predefined safety risk levels.

Safety Risk Function. To formally quantify the safety risk associated with an LLM generation $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$, we define a safety risk function based on the guardrail model G:

$$R_G(x, y) = 1 - G(\text{“safe"}|x, y) \tag{1}$$

This function assigns a risk value between 0 and 1, where a value closer to 1 indicates a higher likelihood that the generated output is unsafe, as determined by the guardrail model.

SafeGen Protocol. We define a extitSafeGen protocol $P_\alpha : \mathcal{X} \mapsto \mathcal{Y}$ as a randomized function parameterized by a generation configuration α. Given an input x, the protocol P_α produces a generated output y, adhering to specific decoding strategies and safety measures encapsulated in α.

The expected safety risk for a given input x under the SafeGen protocol is formulated as:

$$R_G(x, P_\alpha(x)) = 1 - G(\text{“safe"}|x, P_\alpha(x)) \tag{2}$$

This formulation captures the probability of generating an unsafe output under the specified generation configuration α.

C-SafeGen provides two levels of safety risk certification: **(1) Safety risk upper bound certification:** Given a specific generation configuration α, C-SafeGen establishes a high-confidence upper bound on the safety risk. **(2) SafeGen configuration certification for nominal safety risk:** Given a nominal risk threshold r, C-SafeGen identifies the set of generation configurations $\boldsymbol{\alpha}$ that ensure compliance with the specified safety risk level. These certifications enable robust safety evaluations of LLM outputs, facilitating safer and more controlled language generation processes.

3.2 Safety risk certification

In this part, we formulate the approaches to achieving the two levels of safety risk certification in two formal statements, respectively.

Theorem 1 (Safety risk upper bound certification)**.** *Given a SafeGen protocol P_α with generation configuration α, C-SafeGen guarantees that:*

$$\mathbb{P}\left[R_G(x, P_\alpha(x)) \leq \hat{r}_\alpha\right] \geq 1 - \delta, \tag{3}$$

*where the high-probability risk upper bound $\hat{r}_\alpha$, the so-called **conformal safety generation risk**, is given by:*

$$\min\left\{h^{-1}\left(\frac{\ln(1/\delta)}{N_{cal}}; \hat{R}(\hat{\mathcal{D}}_{cal})\right), \Phi_{bin}^{-1}\left(\frac{\delta}{e}; N_{cal}, \hat{R}(\hat{\mathcal{D}}_{cal})\right)\right\}$$

with $h^{-1}(\cdot;\cdot)$ as the partial inverse $h^{-1}(h(a,b);a) = b$ of $h(a,b) = a\log(a/b) + (1-a)\log((1-a)/(1-b))$, and Φ_{bin}^{-1} as the inverse of binomial cumulative distribution function (CDF).

Remarks (Remark of Theorem 1). The upper bound on the safety risk provided by Theorem 1 is derived under a high-probability guarantee, ensuring that the true risk does not exceed $\hat{r}_\alpha$ with probability at least $1 - \delta$. This bound is computed using two distinct finite-sample-valid approaches: one based on the inversion of a likelihood-ratio function and another utilizing the binomial CDF. The combination of these two approaches ensures robustness in cases where one bound is tighter than the other, effectively balancing statistical efficiency and coverage.

Theorem 2 (SafeGen configuration certification for nominal safety risk)**.** *Given a nominal safety risk r, C-SafeGen can certify a configuration set $\boldsymbol{\alpha}_r$ such that each configuration in $\boldsymbol{\alpha}_r$ is guaranteed to keep the generation risk below α. Namely,*

$$\mathbb{P}\left[\sup_{\alpha \in \boldsymbol{\alpha}_r}\left\{R_G\left(x, P_\alpha(x)\right)\right\} \leq r\right] \geq 1 - \delta, \tag{4}$$

where the valid SafeGen configuration set $\boldsymbol{\alpha}_r$ is given by family-wise error rate control $\boldsymbol{\alpha}_r = \{\alpha_j : p_j \leq \delta_j\}$ with $\sum_j \delta_j = \delta$ where p_j is the p-value of the null hypothesis: $\mathcal{H}_j : R_G(x, P_{\alpha_j}(x)) > \alpha$ (j index all feasible SafeGen config) and can be computed by finite-sample valid bounds as shown in Theorem 1.

Remarks (Remark of Theorem 2). Theorem 2 extends the safety certification from individual configurations to a set of SafeGen configurations while controlling the family-wise error rate (FWER). By leveraging hypothesis testing with valid p-values, the theorem constructs a confidence set $\boldsymbol{\alpha}_r$ that guarantees all configurations within it maintain risk below the nominal threshold α. The use of family-wise error control, particularly through a summation constraint on δ_j, ensures that the overall probability of incorrectly certifying an unsafe configuration remains within the desired tolerance level δ.

We leave the complete proofs to Appendix A.

4 SafeGen via Claim-based streaming guardrail

4.1 CSD: Claim-Based Safe Decoding

Consider a fixed safety/hallucination guardrail model $G : \mathcal{X} \times \mathcal{Y} \mapsto 2^{\mathcal{C}}$, which maps the joint input-output space to a set of predefined safety/hallucination categories $\mathcal{C}$ (e.g., "violence," "sexual,"

Algorithm 1 Claim-based stream decoding algorithm

Require: Input prompt x, output text length N, LLM p_θ, Claim critical point ClaimPoint$(\cdot,\cdot)$, Guardrail model G, Claim backtrack probability B_{α_b}, safe resampling function SafeResample$_\theta(\cdot,\cdot)$

$y_0 \leftarrow p_\theta(\cdot|x)$ ▷ Sample first token
counter $\leftarrow 0$ ▷ Decoding counter
for $n = 1$ to N **do**
  if ClaimPoint$(x, y_{<n})$ **then** ▷ Critical point for a complete claim
    $p_b \leftarrow B_{\alpha_b}(G(x, y_{<n}), C_{\text{desired}})$ ▷ Compute claim backtracking probability
  else $p_b \leftarrow 0$
  end if
  counter $\leftarrow$ counter $+ 1$
  if Uniform$(0, 1) < p_b$ **then** ▷ With probability p_b
    $n \leftarrow$ LastClaimPoint$(x, y_{<n})$, continue ▷ Backtrack to the last claim point
  end if
  if $|\text{counter} - n| > \alpha_g$ **then** ▷ Stagnate at the point
    $y_n \leftarrow$ SafeResample$_\theta(\cdot|x, y_{<n})$ ▷ Safe resampling
  else
    $y_n \leftarrow p_\theta(\cdot|x, y_{<n})$
  end if
end for
return $y_{\leq N}$

"group hate," "safe"). Let $p_\theta : \mathcal{X} \mapsto \mathcal{Y}$ be a language model parameterized by θ, defining a conditional distribution over the output space given an input prompt $x \in \mathcal{X}$. That is, the model assigns probability mass to different outputs $y \in \mathcal{Y}$ as $p_\theta(y \mid x)$.

Our goal is to develop a SafeGen protocol P_α parameterized with α that achieves minimal safety risk. Formally, we aim to:

$$\min_{y \in P_\alpha(x)} R_G(x, y), \quad \text{s.t. } G(x, y) \subseteq C_{\text{desired}}, \tag{5}$$

where C_{desired} denotes the set of acceptable categories under the guardrail model (e.g., $C_{\text{desired}} = \{\text{"safe"}\}$ for a safety guardrail). This objective seeks the highest-likelihood output from p_θ that remains within the feasibility set imposed by G. While rejection sampling could be used to enforce this constraint, it is often computationally infeasible.

4.1.1 Claim-Based Decoding

To improve efficiency, we introduce a **claim-based** approach. Let $P : \mathcal{Y} \mapsto 2^{\mathcal{Y}}$ be a *claim partition model*, which decomposes an output $y \in \mathcal{Y}$ into a set of independent claims. This leads to a stronger decoding constraint:

$$\min_{y \in P_\alpha(x)} R_G(x, y), \quad \text{s.t. } G(x, y') \subseteq C_{\text{desired}}, \quad \forall y' \in P(y). \tag{6}$$

This formulation ensures that every claim within y is individually verified against the guardrail model.

4.1.2 Claim Backtrack Probability Function

To control decoding efficiency and safety, we define the **claim backtrack probability function**:

$$B_{\alpha_b}(G(x, y_{<n}), C_{\text{desired}}) = \mathbb{I}\left[\min_{c \in C_{\text{desired}}} G(x, y_{<n})_c - \max_{c \notin C_{\text{desired}}} G(x, y_{<n})_c \leq \alpha_b\right], \tag{7}$$

where α_b regulates the trade-off between decoding speed and adherence to the guardrail: $\alpha_b = 1.0$ strictly enforces Eq. equation 6. $\alpha_b = 0.0$ reduces to greedy decoding. Intermediate values provide a smoothed trade-off, considering both the consistency of G and its overlap with C_{desired}.

4.1.3 Claim Critical Point Detection

Unlike prior methods that prompt LLMs to extract claims [27, 28], we propose an efficient structural approach based on termination indicators. We define a set of **claim termination tokens** $\mathcal{T}$ (e.g., newline, period) and identify a claim boundary when any termination token's probability exceeds α_t. To prevent excessive segmentation, we enforce a **minimum claim length** α_l:

$$\text{ClaimPoint}(x, y_{<n}) = \mathbb{I}\left[L(t; p_\theta(x, y_{<n})) > \alpha_t, \exists t \in \mathcal{T}\right] \cdot \mathbb{I}\left[n - \text{LastClaimPoint}(x, y_{<n}) > \alpha_l\right]. \tag{8}$$

This method ensures computational efficiency while maintaining robust claim detection.

4.1.4 Safe resampling for robust decoding

To enhance robustness and mitigate adversarial attacks, we introduce **safe resampling** techniques inspired by SmoothLLM [34]. Unlike prior approaches that insert high-perplexity substrings at inference time, we directly perturb the KV cache in the ongoing decoding process using: *Random masking/permutation* of hidden states, *Gaussian noise injection* into KV-cache representations, *Structured perturbations* for controlled diversity.

These techniques enhance sampling diversity while preserving alignment with the guardrail model.

4.1.5 Computational efficiency

To enhance computational efficiency, we implement a KV-cache mechanism for the guardrail model. Specifically, the key-value (KV) pairs corresponding to previously verified claims can be stored, allowing efficient backtracking without recomputing the entire sequence.

Formally, let $\text{KV}_{1:n}$ denote the cached representations for the sequence $y_{1:n}$, and define the backtracking operation as follows:

$$\text{KV} \leftarrow \text{KV} \cup \text{KV}_{n:n+1}, \quad \text{if } B_{\alpha_b}(G(x, y_{<n+1}), C_{\text{desired}}) = 0. \tag{9}$$

This ensures that upon backtracking, only the key-value pairs corresponding to the current claim are erased, while maintaining the cached representations for verified claims.

4.2 Analysis of CSD algorithm

Definition 1 (Restricted output distribution)**.** Let p be the distribution over feasible output space $\mathcal{O}$. Let $G : \mathcal{O} \mapsto \{0, 1\}$ be a guardrail model where 1 denotes desired output and 0 denotes undesired output. Let p_G be a restricted output distribution of p by guardrail G with the formulation:

$$p_G(y) = \frac{p(y)\mathbb{I}[G(y) = 1]}{\sum_{y \in \mathcal{Y}} p(y)\mathbb{I}[G(y) = 1]} \tag{10}$$

Remarks. The restricted output distribution p_G represents the renormalized probability mass of the feasible output distribution p, conditioned on the constraint imposed by the guardrail model G. This ensures that only outputs classified as desired ($G(y) = 1$) contribute to the final distribution. The denominator acts as a normalizing constant, ensuring that p_G remains a valid probability distribution.

Definition 2 (Claim)**.** An output claim $\tilde{y}$ is defined as a sequence of tokens that delivers complete semantics, which is judged by a *claim discriminator* C. Specifically, $C(y_{\leq t_1}) = 1, C(y_{\leq t_2}) = 1$ and $C(y_{\leq t}) = 0, \forall t_1 < t < t_2$ implicates that the sequence $y_{t1:t2}$ is a *claim*.

Remarks. The claim discriminator C plays a crucial role in identifying semantically complete claims within a sequence. The definition ensures that a claim is an isolated segment where the beginning and end are both recognized by C, while the intermediate segments do not form independent claims. This property is essential for structured generation and for enforcing guardrails at the claim level.

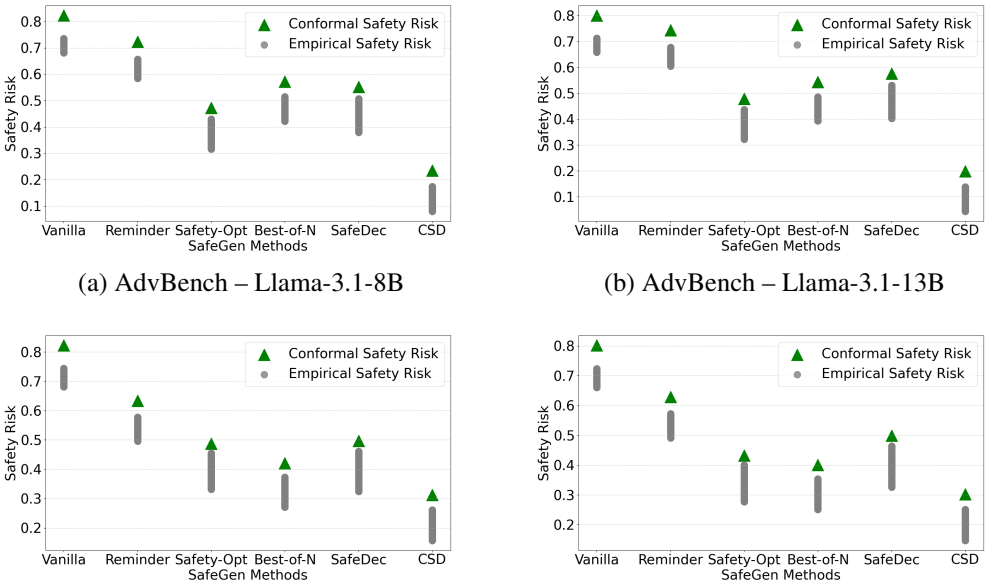
Assumption 4.1 (Claim risk cascade)**.** *Consider an output sequence o which consists of a sequence of N claims by claim discriminator C: $o = [\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_N]$ We assume that the risk of prefix claims implicates the risk of output judged by guardrail G:*

$$G([\tilde{y}_1, ..., \tilde{y}_n]) = 0, \exists n \in [1, N] \Longrightarrow G([\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_N]) = 0 \tag{11}$$

Remarks. The claim risk cascade assumption formalizes the idea that the presence of an undesired claim in a sequence guarantees that the entire sequence is also undesired. This assumption aligns with the intuition that risks in intermediate steps propagate forward, affecting the final judgment by the guardrail model G. This is a conservative approach that simplifies analysis while ensuring safety in controlled generation.

Theorem 3 (Algorithm 1 recovers restricted output distribution)**.** *Under the claim risk cascade assumption as Assumption 4.1, if there exists at least one desired output judged by guardrail G, the output distribution of the claim-based decoding streaming algorithm in Algorithm 1 (without safe resampling) is identical to the restricted output distribution as Definition 1.*

Remarks. This theorem establishes that the claim-based decoding streaming algorithm produces outputs that exactly match the restricted output distribution p_G. The key condition for this result is



(a) AdvBench – Llama-3.1-8B

(b) AdvBench – Llama-3.1-13B

(c) JailbreakBench – Llama-3.1-8B

(d) JailbreakBench – Llama-3.1-13B

Figure 2: Evaluation of conformal safety risk (upper bound from Theorem 1) and empirical safety risk (mean risk on sampled test sets) across two benchmarks—AdvBench and JailbreakBench—using Llama-3.1 models (8B and 13B) with ShieldGemma-9B as the guardrail. Results show that (1) conformal risk bounds are valid and tight; and (2) our CSD method consistently achieves the lowest safety risk. Note: Grey bars result from overlapping dots.

the existence of at least one feasible sequence that satisfies the guardrail G. The theorem is significant because it ensures that the decoding algorithm does not introduce biases or distortions beyond those imposed by the guardrail, thereby preserving the original probability structure of the restricted output space.

5 Evaluation

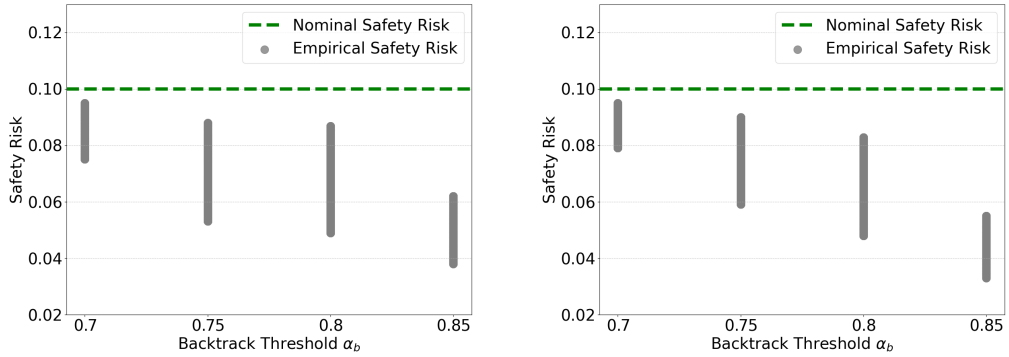
5.1 Evaluation setup

Dataset & Models. As AdvBench [50] and JailbreakBench [8] are widely used for evaluating LLM safety [23, 9, 26], we adopt it as our primary evaluation dataset.

For text generation, we consider LLMs `Llama-3.1-8B`, and `Llama-3.1-13B` as inference models. Additionally, we append adversarial suffixes to the user query to jailbreak the model, creating a more challenging safety evaluation scenario. These adversarial suffixes are optimized using the GCG attack [50] on the corresponding inference model as target models.

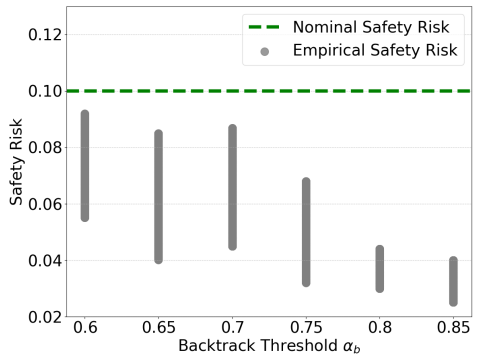
Metrics. Without specification, we employ `LlamaGuard3-8B` [10] as the guardrail model G and use the unsafety probability predicted by it as the safety risk function R_G. Additionally, we also consider `ShieldGemma-9B` as guardrail models for guardrail comparisons.

Baselines. We consider four baselines for safe generation: (1) Vanilla generation, using a temperature of 1.0; (2) Self-reminder safety prompt (Reminder) [42], which incorporates a system safety prompt and an additional reminder prompt after the user query to encourage safer generation; (3) Safety-Opt [48], which optimizes a soft safety prompt prefix to guide the model toward safer response patterns; (4) Best-of-N [35], which selects the safest response from a set of 10 generated outputs; and (5) SafeDecoding (SafeDec) [44], which uses a safety-aware contrastive decoding strategy for LLMs to generate helpful and harmless responses to user queries.

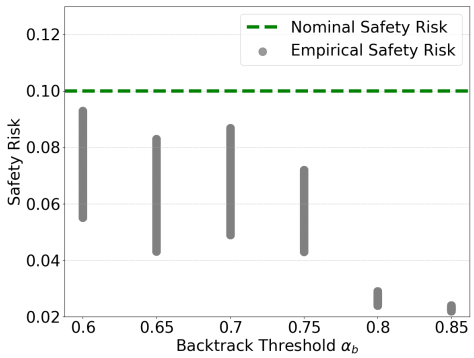


(a) AdvBench – Llama-3.1-8B

(b) AdvBench – Llama-3.1-13B



(c) JailbreakBench – Llama-3.1-8B



(d) JailbreakBench – Llama-3.1-13B

Figure 3: Evaluation of SafeGen configurations with nominal safety risk 0.10 (Theorem 2). Valid configurations remain below the nominal risk, and larger backtrack thresholds α_b yield lower and more stable risks. Note: The grey bar results from overlapping grey dots.

5.2 Safety risk upper bound certification

Figure 2 presents a comparative evaluation of conformal safety risk (theoretical upper bound) and empirical safety risk (mean observed risk) across six SafeGen methods on two safety benchmarks, AdvBench and JailbreakBench, using Llama-3.1 models (8B and 13B) with ShieldGemma-9B as the guardrail. Across all settings, we observe that CSD consistently achieves the lowest safety risks—both conformal and empirical—indicating its superior capability in reducing unsafe generations under formal guarantees. The conformal risk bounds are generally tight, validating the reliability of the theoretical risk estimation across different methods and model sizes. Notably, methods such as SafeDec and Safety-Opt also reduce risk substantially compared to the vanilla baseline, but fail to match the low-risk performance of CSD. The trend persists across both model scales and benchmarks, underscoring the robustness of CSD's improvements in safety under adversarial and jailbreak-style attacks.

In addition to method-wise comparisons, the results also reveal consistent trends across model sizes. Specifically, the larger model (Llama-3.1-13B) does not uniformly outperform the smaller 8B variant in safety risk reduction. In some cases—such as under Reminder and Best-of-N—the 13B model exhibits slightly higher safety risk than the 8B model, suggesting that scaling up model size alone does not guarantee improved safety under adversarial or conformal evaluation. This highlights the importance of method design (e.g., CSD) over sheer model capacity in achieving reliable and provably safe generations.

5.3 SafeGen configuration certification for nominal safety risk

Figure 3 illustrates the validity of SafeGen configurations as certified by Theorem 2, and investigates how the backtrack threshold α_b affects the empirical safety risk across different benchmarks and model sizes. The dashed green line indicates the nominal safety risk bound (set to 0.10), while the grey markers represent empirical safety risks under varying values of α_b. We highlight three main observations: **(1) Certified safety guarantees hold across all settings.** For every evaluated configuration, the empirical safety risks remain strictly below the nominal risk threshold, confirming the high-probability validity guaranteed by Theorem 2. This supports the theoretical soundness of the certification framework across both AdvBench and JailbreakBench. **(2) Higher α_b leads to lower safety risk.** Across all plots, increasing the backtrack threshold α_b consistently reduces empirical safety risk. This trend is consistent for both Llama-3.1-8B and Llama-3.1-13B, demonstrating that SafeGen with a higher α_b allows more cautious decoding behavior and safer outputs. **(3) Risk stability improves with larger α_b.** We also observe a reduction in the variability of empirical safety risk as α_b increases. This indicates that larger thresholds not only yield safer generations but also improve the reliability and consistency of safety performance across different test samples.

These results demonstrate that SafeGen provides strong empirical and theoretical safety guarantees, with all evaluated configurations maintaining risks below the specified nominal bound. Adjusting the backtrack threshold α_b is an effective mechanism to improve both the magnitude and stability of safety risk, and can be tuned independently of model scale. Together, these findings underscore the importance of principled decoding strategies—such as SafeGen—for deploying LLMs in safety-critical applications.

5.4 Safety vs. decoding efficiency

Figure 4 compares the **runtime per instance** (grey bars) and the **mean safety risk** (purple bars) for different SafeGen methods: `Best-of-N`, `CSD`, and `CSD+KV Cache`. The results provide insights into the trade-offs between computational efficiency and safety performance. We have the following observations: (1) Best-of-N incurs the highest runtime, exceeding 40 seconds per instance, while also exhibiting the highest mean safety risk. This indicates that generating multiple completions and selecting the safest one is computationally expensive without significantly improving safety. (2) CSD achieves a substantial reduction in runtime (approximately 7 seconds per instance) while also lowering the mean safety risk compared to Best-of-N. This demonstrates the efficiency of the CSD method in balancing safety and speed. (3) CSD+KV Cache further reduces the runtime (below 5 seconds per instance) while maintaining a similar mean safety risk to CSD. This highlights the effectiveness of KV caching in accelerating SafeGen methods without compromising safety.

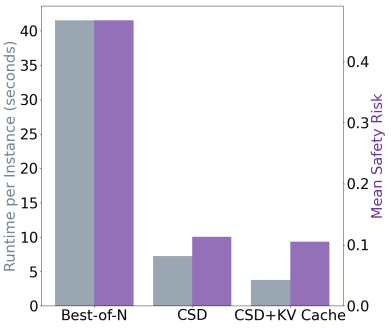


Figure 4: Evaluation of decoding runtime and mean safety risk for the strongest baseline Best-of-N, our proposed SafeGen protocol CSD, and CSD with KV cache. CSD achieves a better efficiency-safety tradeoff than Best-of-N. KV cache preserves safety while significantly reducing decoding time.

The results indicate that `CSD+KV Cache` provides the best trade-off between efficiency and safety, significantly reducing computational cost while maintaining low safety risk. `Best-of-N`, despite being a simple approach, is highly inefficient, making it impractical for real-time applications. These findings emphasize the importance of optimizing SafeGen methods for both speed and reliability.

6 Conclusion

The rapid adoption of open-source LLMs underscores the urgent need for rigorous safety guarantees in their generations, particularly in high-stakes applications. Existing empirical defenses, including RLHF, guardrails, and decoding-time interventions, lack formal assurances, limiting their reliability against adversarial exploits. To address this gap, we introduced `C-SafeGen`, a model-agnostic certification framework that provides theoretical risk bounds and enables the enforcement of provable safety constraints. Complemented by our novel CSD decoding algorithm, `C-SafeGen` ensures both safety and fluency in generated text. Empirical evaluations confirm the efficacy of our approach, demonstrating its potential as a robust foundation for the deployment of provably safe LLMs.

Acknowledgements

This work is partially supported by the National Science Foundation under grant No. 1910100, No. 2046726, NSF AI Institute ACTION No. IIS-2229876, DARPA TIAMAT No. 80321, the National Aeronautics and Space Administration (NASA) under grant No. 80NSSC20M0229, ARL Grant W911NF-23-2-0137, Alfred P. Sloan Fellowship, the research grant from eBay, AI Safety Fund, Virtue AI, and Schmidt Science.

References

[1] Ai content moderation by microsoft azure. *https://azure.microsoft.com/en-us/products/ai-services/ai-content-safety*.

[2] Detoxify by unitary ai. *https://github.com/unitaryai/detoxify*.

[3] Anastasios N Angelopoulos, Stephen Bates, Emmanuel J Candès, Michael I Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv preprint arXiv:2110.01052*, 2021.

[4] Anastasios N Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. *arXiv preprint arXiv:2208.02814*, 2022.

[5] Luke Bates and Iryna Gurevych. Like a good nearest neighbor: Practical content moderation with sentence transformers. *arXiv e-prints*, pages arXiv–2302, 2023.

[6] Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM (JACM)*, 68(6):1–34, 2021.

[7] Vidmantas Bentkus. On hoeffding's inequalities. 2004.

[8] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *NeurIPS Datasets and Benchmarks Track*, 2024.

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

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

[11] Shaona Ghosh, Prasoon Varshney, Erick Galinkin, and Christopher Parisien. Aegis: Online adaptive ai content safety moderation with ensemble of llm experts. *arXiv preprint arXiv:2404.05993*, 2024.

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

[13] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

[14] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[15] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

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

[17] Mintong Kang, Chejian Xu, and Bo Li. Advwave: Stealthy adversarial jailbreak attack against large audio-language models. *arXiv preprint arXiv:2412.08608*, 2024.

[18] Deepak Kumar, Yousef AbuHashem, and Zakir Durumeric. Watch your language: Investigating content moderation with large language models. *arXiv preprint arXiv:2309.14517*, 2024.

[19] Alyssa Lees, Vinh Q Tran, Yi Tay, Jeffrey Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. A new generation of perspective api: Efficient multilingual character-level transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3197–3207, 2022.

[20] Jing Lei, James Robins, and Larry Wasserman. Distribution-free prediction sets. *Journal of the American Statistical Association*, 108(501):278–287, 2013.

[21] Yuhui Li, Fangyun Wei, Jinjing Zhao, Chao Zhang, and Hongyang Zhang. Rain: Your language models can align themselves without finetuning. *arXiv preprint arXiv:2309.07124*, 2023.

[22] Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation. *arXiv preprint arXiv:2310.17389*, 2023.

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

[24] Huan Ma, Changqing Zhang, Huazhu Fu, Peilin Zhao, and Bingzhe Wu. Adapting large language models for content moderation: Pitfalls in data engineering and supervised fine-tuning. *arXiv preprint arXiv:2310.03400*, 2023.

[25] Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. A holistic approach to undesired content detection in the real world. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15009–15018, 2023.

[26] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. *arXiv preprint arXiv:2312.02119*, 2023.

[27] Christopher Mohri and Tatsunori Hashimoto. Language models with conformal factuality guarantees. *arXiv preprint arXiv:2402.10978*, 2024.

[28] Jingwei Ni, Minjing Shi, Dominik Stammbach, Mrinmaya Sachan, Elliott Ash, and Markus Leippold. Afacta: Assisting the annotation of factual claim detection with reliable llm annotators. *arXiv preprint arXiv:2402.11073*, 2024.

[29] OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei

Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2023.

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

[31] Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S Jaakkola, and Regina Barzilay. Conformal language modeling. *arXiv preprint arXiv:2306.10193*, 2023.

[32] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[33] Traian Rebedea, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, and Jonathan Cohen. Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails. *arXiv preprint arXiv:2310.10501*, 2023.

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

[35] Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection. *arXiv preprint arXiv:2410.20290*, 2024.

[36] Fei Tan, Yifan Hu, Kevin Yen, and Changwei Hu. Bert-beta: A proactive probabilistic approach to text moderation. *arXiv preprint arXiv:2109.08805*, 2021.

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

[38] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

[39] Volodya Vovk, Alexander Gammerman, and Craig Saunders. Machine-learning applications of algorithmic randomness. 1999.

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

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

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

[43] Chejian Xu, Mintong Kang, Jiawei Zhang, Zeyi Liao, Lingbo Mo, Mengqi Yuan, Huan Sun, and Bo Li. Advweb: Controllable black-box attacks on vlm-powered web agents. *arXiv preprint arXiv:2410.17401*, 2024.

[44] Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. Safedecoding: Defending against jailbreak attacks via safety-aware decoding. *arXiv preprint arXiv:2402.08983*, 2024.

[45] Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. Watch out for your agents! investigating backdoor threats to llm-based agents. *arXiv preprint arXiv:2402.11208*, 2024.

[46] Yachong Yang and Arun Kumar Kuchibhotla. Finite-sample efficient conformal prediction. *arXiv preprint arXiv:2104.13871*, 2021.

[47] Zhuowen Yuan, Zidi Xiong, Yi Zeng, Ning Yu, Ruoxi Jia, Dawn Song, and Bo Li. Rigorllm: Resilient guardrails for large language models against undesired content. *arXiv preprint arXiv:2403.13031*, 2024.

[48] Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. Prompt-driven llm safeguarding via directed representation optimization. *arXiv preprint arXiv:2401.18018*, 2024.

[49] Xuhui Zhou. *Challenges in automated debiasing for toxic language detection*. University of Washington, 2020.

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

A Omitted proofs

A.1 Proof of Theorem 1

Proof of Theorem 1. The proof sketch follows [3]. Since the risk function $R(\cdot,\cdot)$ is upper bounded by 1, we can apply a tighter version of Hoeffding's inequality [13] for $\hat{\alpha} > \mathbb{E}[R_G(x, P_\alpha(x))]$:

$$\mathbb{P}\left[R_G(x, P_\alpha(x)) \geq \hat{\alpha}\right] \leq \exp\left\{-N_{\text{cal}} h(\hat{R}(\hat{\mathcal{D}}_{\text{cal}}), \hat{\alpha})\right\} \tag{12}$$

Also, applying Bentkus inequality [7], we have:

$$\mathbb{P}\left[R_G(x, P_\alpha(x)) \geq \hat{\alpha}\right] \leq e\mathbb{P}\left[\text{Bin}(N_{\text{cal}}, \hat{\alpha}) \leq \left\lceil N_{\text{cal}} \hat{R}(\hat{\mathcal{D}}_{\text{cal}})\right\rceil\right] \tag{13}$$

Combining Equations (12) and (13), we have:

$$\mathbb{P}\left[R_G(x, P_\alpha(x)) \geq \hat{\alpha}\right] \leq \min\left(\exp\left\{-N_{\text{cal}} h\left(\hat{R}(\hat{\mathcal{D}}_{\text{cal}}), \hat{\alpha}\right)\right\}, e\mathbb{P}\left[\text{Bin}(N_{\text{cal}}, \hat{\alpha}) \leq \left\lceil N_{\text{cal}} \hat{R}(\hat{\mathcal{D}}_{\text{cal}})\right\rceil\right]\right) \tag{14}$$

Or equivalently, given uncertainty $1-\delta$, we have:

$$\delta = \min\left(\exp\left\{-N_{\text{cal}} h\left(\hat{R}(\hat{\mathcal{D}}_{\text{cal}}), \hat{\alpha}\right)\right\}, e\mathbb{P}\left[\text{Bin}(N_{\text{cal}}, \hat{\alpha}) \leq \left\lceil N_{\text{cal}} \hat{R}(\hat{\mathcal{D}}_{\text{cal}})\right\rceil\right]\right), \tag{15}$$

which leads to the following by formulating an inverse function:

$$\hat{\alpha} = \min\left\{h^{-1}\left(\frac{\ln(1/\delta)}{N_{\text{cal}}}; \hat{R}(\hat{\mathcal{D}}_{\text{cal}})\right), \Phi_{\text{bin}}^{-1}\left(\frac{\delta}{e}; N_{\text{cal}}, \hat{R}(\hat{\mathcal{D}}_{\text{cal}})\right)\right\} \tag{16}$$

□

A.2 Proof of Theorem 2

Proof of Theorem 2. The proof follows [14]. We consider $|\Lambda|$ independent hypothesis test corresponding to the $|\Lambda|$ Null hypothesis. By the Bonferroni method, each test is performed at a significance level of $\frac{\delta}{|\Lambda|}$. Therefore, The probability of not making a Type I error in a single test is $1-\frac{\delta}{|\Lambda|}$. The probability of making no Type I error in all $|\Lambda|$ tests is $(1-\frac{\delta}{|\Lambda|})^{|\Lambda|}$. The probability of making at least one Type I error (i.e., FWER) is the complement of making no Type I errors, which is $1-(1-\frac{\delta}{|\Lambda|})^{|\Lambda|} \leq \delta$. Therefore, we prove that the familywise error rate is δ for Bonferroni correction. Thus, going back to the risk guarantee, we conclude the proof. □

A.3 Proof of Theorem 3

Proof of Theorem 3. Let the output distribution by Algorithm 1 be p_{ccd}. To show the result, we need to show that for any output sequence $y \in \mathcal{Y}$, $p_{\text{ccd}}(y) = p_G(y)$.

We consider the following scenarios respectively: (1) this is an invalid output by the guardrail model: $G(y) = 0$; (2) this a valid output: $G(y) = 1$.

Scenario (1): $G(y) = 0$. According to CCD algorithm, such an undesired claim would always be associated with 1.0 backtrack probability in Line 4 of Algorithm 1, and thus we have $p_{\text{ccd}}(y) = 0$. On the other hand, according to Equation (10), we always have $p_G(y) = 0$ when $G(y) = 0$. Therefore, we have $p_{\text{ccd}}(y) = p_G(y)$ when $G(y) = 0$.

Scenario (2): $G(y) = 1$. In this scenario, we basically would like to validate that the backtrack mechanism in Lines 7-9 of Algorithm 1 does not distort the distribution.

Part (a): To show that, we first validate that the support of output distribution by CCD is identical to that of the restricted output distribution.

For any valid support in the restricted output distribution $y = [\tilde{y}_1, \tilde{y}_2, .., \tilde{y}_N]$, there always exists an unrisky claim decoding path. We consider this simple claim decoding path $\tilde{y}_1, \tilde{y}_2, .., \tilde{y}_N$. Due to

Assumption 4.1, since the complete output sequence o is unrisky, then each claim prefix would also be unrisky (otherwise violating the claim risk cascade assumption). Therefore, this claim decoding path is a valid instantiation of CCD algorithm.

Considering the reverse direction, for any valid support output y by CCD algorithm, it is obvious that $G(y) = 1$ given the acceptance of the last claim, and thus, this output is also a valid support of the restricted output distribution p_G.

Part (b): Then, we will prove that for any valid support $y = [\tilde{y}_1, \tilde{y}_2, .., \tilde{y}_N]$ such that $p_G(y) > 0$ and $p_{\text{ccd}}(y) > 0$, we have $p_G(y) = p_{\text{ccd}}(y)$.

For ease of notation, we notate $\alpha = \sum_{y \in \mathcal{Y}} p(y)\mathbb{I}[G(y) = 1]$, which is a normalization factor in Equation (10). Then following the chain rule, we can compute $p_G(y)$ as:

$$p_G(y) = \frac{1}{\alpha} p(\tilde{y}_1)p(\tilde{y}_2|\tilde{y}_1) \cdots p(\tilde{y}_N|\tilde{y}_1, ..., \tilde{y}_{N-1}) \tag{17}$$

Since y is a desired output (i.e., $G(y) = 1$), each claim prefix is also desired according to the risk cascade assumption in Assumption 4.1. Then we have:

$$p_G(y) = \frac{1}{\alpha} p(\tilde{y}_1, G([\tilde{y}_1]) = 1)p(\tilde{y}_2, G([\tilde{y}_1, \tilde{y}_2]) = 1|\tilde{y}_1) \cdots p(\tilde{y}_N, G([\tilde{y}_1, ..., \tilde{y}_N]) = 1|\tilde{y}_1, ..., \tilde{y}_{N-1}) \tag{18}$$

Then we analyze the point mass of output y by the CCD algorithm $p_{\text{ccd}}(y)$. We can formulate the decoding path of CCD algorithm that leads to output y as $\tilde{\boldsymbol{y}}_1, \tilde{y}_1, \tilde{\boldsymbol{y}}_2, \tilde{y}_2, ..., \tilde{\boldsymbol{y}}_N, \tilde{y}_N$, where $\tilde{\boldsymbol{y}}_n$ denotes a sequence of backtracked risky claims at time step n.

Then we prove a key invariability property that

$$p(\tilde{y}_n, , G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1}) = \sum_{\tilde{\boldsymbol{y}}_n} p(\tilde{\boldsymbol{y}}_n, \tilde{y}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1}) \tag{19}$$

We can derive as the following:

$$\sum_{\tilde{\boldsymbol{y}}_n} p(\tilde{\boldsymbol{y}}_n, \tilde{y}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1}) \tag{20}$$

$$= \sum_{\tilde{\boldsymbol{y}}_n} p(\tilde{\boldsymbol{y}}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1})p(\tilde{y}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1}, \tilde{\boldsymbol{y}}_n) \tag{21}$$

$$= \sum_{\tilde{\boldsymbol{y}}_n} p(\tilde{\boldsymbol{y}}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1})p(\tilde{y}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1}) \tag{22}$$

For ease of notation, let $u = p(\tilde{y}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 0|\tilde{y}_1, ..., \tilde{y}_{n-1})$ and $v = p(\tilde{y}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1})$, then we have:

$$\sum_{\tilde{\boldsymbol{y}}_n} p(\tilde{\boldsymbol{y}}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1})p(\tilde{y}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1}) \tag{23}$$

$$= (1-u)v + u(1-u)v + u^2(1-u)v + u^3(1-u)v + \cdots \tag{24}$$

$$= v \tag{25}$$

$$= p(\tilde{y}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1}) \tag{26}$$

Therefore, we have:

$$p_{\text{ccd}}(y) = \frac{1}{\alpha} \prod_{n=1}^{N} \sum_{\tilde{\boldsymbol{y}}_n} p(\tilde{\boldsymbol{y}}_n, \tilde{y}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1}) \tag{27}$$

$$= \frac{1}{\alpha} \prod_{n=1}^{N} p(\tilde{y}_n, G([\tilde{y}_1, ..., \tilde{y}_n]) = 1|\tilde{y}_1, ..., \tilde{y}_{n-1}) \tag{28}$$

$$= p_G(y) \tag{29}$$

□

B Discussion and Limitation

The C-SafeGen framework is able to safeguard LLMs' practical deployment and applications against ethical and societal concerns. Existing research shows that the responses of LLMs can be biased towards some demographic groups and not be aligned with human ethics. With C-SafeGen, we can define a bias/ethics risk function and control the generation risk below a desired level. The risk guarantee provided by C-SafeGen enhances the use of LLMs, addressing societal issues and regulatory infringements. We do not expect any negative societal consequences for our work.

While C-SafeGen provides formal guarantees under mild assumptions, it currently relies on the availability of reliable external classifiers or risk estimators to define the safety risk function. The effectiveness of the certification heavily depends on the quality and coverage of these components, which may be incomplete or biased in practice. Additionally, C-SafeGen operates in a black-box setting and does not account for internal model behaviors or hidden representations that might influence safety. Finally, although C-SafeGen bounds risk on the evaluation distribution, distributional shifts at deployment time (e.g., novel user queries or emerging adversarial strategies) may compromise safety, highlighting the need for future extensions toward adaptive or online certification mechanisms.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We include theoretical proofs in Appendix A and evaluation details in Section 5.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We include that in Appendix B.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The proof is in Appendix A.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We include details in Section 5.

   Guidelines:

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

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

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

Answer: [No]

Justification: We will do so on paper acceptance.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We include that in Section 5.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We do multiple runs to reduce sample error.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We include that in Section 5.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We follow the code of ethics during the project.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We include that in Appendix B.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: We do not release data or models.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The datasets are all in valid use.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: We do not involve that.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: We do not involve that.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [NA]

    Justification: We do not involve that.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: We provide it in meta data of paper submission.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.