Robust and Efficient Guardrails with Latent Reasoning

Siddharth Sai **Xiaofei Wen** **Muhao Chen**
University of California, Davis
{sai,xfwe,muhchen}@ucdavis.edu

Abstract

Maintaining the safety of large language models (LLMs) is crucial as they are increasingly deployed in real-world applications. Existing safety guardrails typically rely on single-pass classification or, more recently, distilled reasoning. Reasoning-based guardrails significantly outperform classification-only baselines, but they incur substantial query latency and token overhead that make them impractical for high-throughput deployment. To address this challenge, we propose CoLaGuard, a guardrail model that transfers multi-step safety reasoning into a continuous latent space through a stage-wise training curriculum, enabling direct hidden-state propagation at inference. Evaluated on ten prompt- and response-moderation settings spanning eight safety benchmarks, CoLaGuard improves macro-F1 by 8.24 points over Llama Guard 3 and matches our explicit reasoning baseline, GuardReasoner, in macro-F1 while delivering a 12.9× speedup and 22.4× reduction in token usage. Our results suggest that latent reasoning offers a practical alternative to explicit rationale generation for deployable guardrails, jointly improving safety robustness and inference efficiency rather than treating them as competing objectives. Project page: github.com/SiddSai/CoLaGuard

1 Introduction

As Large Language Models (LLMs) become integral to daily and industrial applications, ensuring their alignment with human values is critical. Although alignment training methods such as RLHF (Ouyang et al., 2022; Rafailov et al., 2023) can improve model behavior, they require modifying the target model and are costly to update after deployment. External safety guardrails (Inan et al., 2023; Han et al., 2024) therefore provide a practical alternative by offloading input and output moderation to smaller, third-party models. Early guardrails typically formulate moderation as single-pass classification, which is efficient but often becomes brittle under ambiguous, adversarial, or context-dependent safety decisions.

Recent explicit reasoning guardrails (Wen et al., 2025; Liu et al., 2025) improve robustness by learning from distilled chain-of-thought (CoT) supervision (Hsieh et al., 2023; Kim et al., 2023) and generating intermediate rationales before predicting a safety label. MrGuard (Yang et al., 2025) further extends reasoning-based guardrails to multilingual safety moderation by combining synthetic multilingual supervision with curriculum-guided Group Relative Policy Optimization (GRPO) (Shao et al., 2024). However, this robustness comes at a steep computational cost. Because these models verbalize their intermediate rationales, moderation becomes a long autoregressive generation process. The additional CoT tokens substantially inflate inference time and completion-token cost, making explicit reasoning guardrails difficult to deploy in high-traffic, real-time settings (Liu et al., 2025; Sreedhar et al., 2025). Existing efficiency-oriented variants, such as shorter supervised traces or reasoning on/off switches (NVIDIA, 2025; Sreedhar et al., 2025), reduce the amount or frequency of rationale generation but still rely on explicit decoding and may sacrifice robustness.

This motivates a natural question: *can guardrails retain the benefits of reasoning supervision without generating reasoning tokens at inference time?* We study this question through CoLaGuard, a latent-reasoning safety guardrail that internalizes explicit safety rationales into continuous recurrent states as shown in Figure 1. Inspired by Coconut (Hao et al., 2025) and ICoT-SI (Deng et al., 2025), CoLaGuard performs a fixed number of latent recurrent steps in place of explicit rationale generation. It first learns from CoT supervision and then progressively replaces rationale tokens with latent states, allowing the model to directly predict the safety label without autoregressive rationale generation. A practical challenge is that pretrained

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 57256–57268
October 24-29, 2026 ©2026 Association for Computational Linguistics

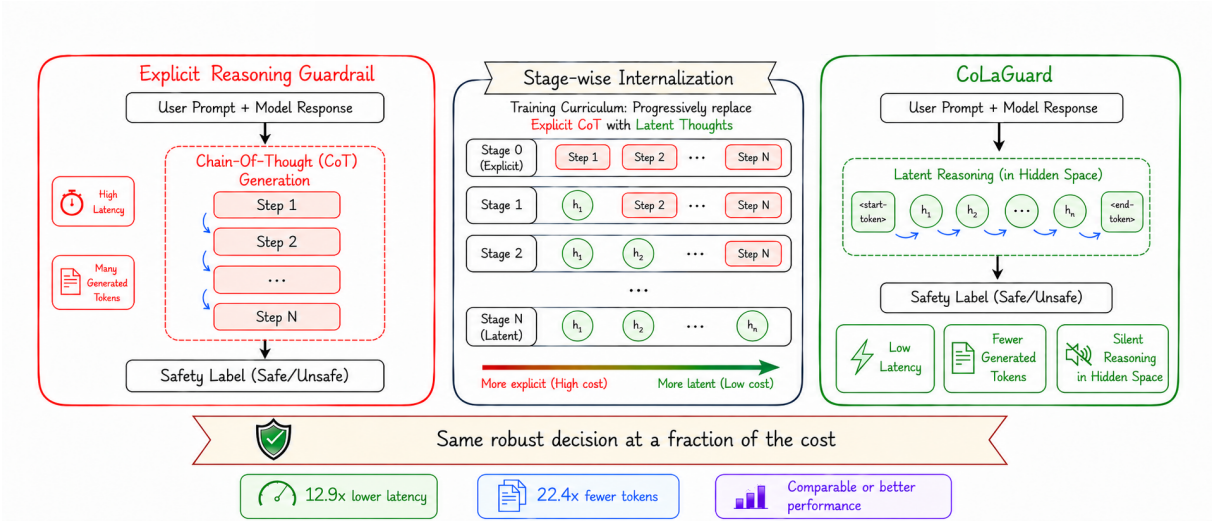

Figure 1: Overview of CoLaGuard. Unlike explicit reasoning guardrails (left) that generate chain-of-thought tokens before assigning labels, CoLaGuard (right) reasons through recurrent latent states, preserving moderation performance while avoiding token generation overhead and enabling 12.9× faster inference and 22.4× fewer tokens. CoLaGuard's stage-wise internalization curriculum (center) begins with explicit CoT supervision and progressively replaces reasoning tokens with latent states, shifting reasoning into hidden activations.

LLMs are optimized to consume token embeddings rather than recirculated contextual hidden states, which can create a distribution mismatch during latent recurrence. To reduce this mismatch, we adopt Context-Prediction Fusion (Liu et al., 2026), which combines contextual hidden-state information with predictive semantic guidance from the vocabulary embedding space. This stabilizes latent recurrence while preserving the latency and token-efficiency benefits of avoiding explicit CoT generation.

In summary, this work makes three main contributions. (1) We introduce CoLaGuard, a latent-reasoning safety guardrail that internalizes explicit safety rationales through a stage-wise curriculum, enabling moderation without autoregressive rationale generation at inference time. (2) We show that CoLaGuard preserves the robustness of explicit reasoning guardrails while substantially reducing inference cost, suggesting that reasoning-based moderation can be made practical without verbalized rationales. (3) We analyze the latent recurrence process and find that CoLaGuard improves over vanilla Coconut, consistent with progressive safety-relevant representation shifts across latent steps that are largely absent in vanilla Coconut recurrence.

2 Related Work

LLM Guardrails External guardrails provide a lightweight mechanism for safety moderation without modifying the base LLM. Early architectures such as Llama Guard (Inan et al., 2023) and WildGuard (Han et al., 2024) treated moderation as classification, followed by models like ShieldGemma (Zeng et al., 2024), Aegis (Ghosh et al., 2024), and Qwen3Guard (Zhao et al., 2025), which improved performance through broader taxonomies. The broader guardrail literature has expanded robustness through adversarially resilient moderation (Yuan et al., 2024) and structured safety knowledge (Kang and Li, 2025). Recent work further improves performance with reasoning: GuardReasoner (Liu et al., 2025) and ThinkGuard (Wen et al., 2025) use chain-of-thought rationales (Wei et al., 2023) from expert models to improve generalization, while MrGuard (Yang et al., 2025) extends reasoning-based guardrails to multilingual moderation through synthetic multilingual supervision and curriculum-guided GRPO. Others explore efficiency trade-offs through shorter rationale traces and on/off switches (NVIDIA, 2025; Sreedhar et al., 2025; Rebedea et al., 2023). However, because these models verbalize reasoning in natural language, they incur steep autoregressive decoding costs that limit their practicality for high-traffic, real-world deployment.

Latent Reasoning A growing literature suggests that effective reasoning can occur within a model's hidden states rather than through explicit tokens (Chen et al., 2025; Zhu et al., 2025; Biran et al.,

2024). This space includes augmenting models with "thinking" tokens (Goyal et al., 2024; Zelikman et al., 2024; Pfau et al., 2024), internalizing CoT through staged curricula (Deng et al., 2023, 2025), and feeding hidden states back as continuous input embeddings (Hao et al., 2025; Cheng and Durme, 2024; Zhu et al., 2025). However, these methods have largely been studied on mathematical and logical reasoning tasks, and directly recycling raw hidden states can become unstable at larger scales due to distribution mismatch with the token embedding manifold. Latent Thoughts Tuning (Liu et al., 2026) addresses this with a context-prediction fusion mechanism that aligns contextual hidden states with predictive signals from the vocabulary embedding space. COLAGUARD adapts these techniques, showing that latent reasoning can drastically reduce latency costs and preserve the robustness of explicit baselines in safety moderation.

3 COLAGUARD

We now present COLAGUARD, a latent-reasoning guardrail framework for efficient prompt and response moderation. COLAGUARD uses explicit safety rationales generated by expert models as training-time supervision, then progressively internalizes this step-by-step reasoning into recurrent latent states so that inference incurs only a fixed latent computation budget before decoding the safety labels. We formulate the guardrail task in §3.1, describe reasoning-augmented supervision and explicit warm-up in §3.2–§3.3, and present latent recurrence, stage-wise internalization and efficient inference in §3.4–§3.6.

3.1 Guardrail Task

Given a user prompt x and a model response s, a guardrail model G_θ predicts the safety of both the input request and the generated response $(\hat{y}^p, \hat{y}^r) = G_\theta(x, s)$, where $\hat{y}^p \in \mathcal{Y}$ denotes the predicted prompt harmfulness label, $\hat{y}^r \in \mathcal{Y}$ denotes the predicted response harmfulness label, and $\mathcal{Y}$ denotes the set of safety categories in the guardrail's policy.

3.2 Reasoning-Augmented Supervision

The central challenge is maintaining the robustness of reasoning-based guardrails without requiring that the guardrail verbalize its reasoning process at inference time. COLAGUARD addresses this by using explicit rationales for the initial training scaffolding. This follows prior work on chain-of-thought reasoning, step-by-step distillation, and reasoning-based safety guardrails, where intermediate rationales provide richer supervision than final labels alone (Wei et al., 2023; Hsieh et al., 2023; Kim et al., 2023; Liu et al., 2025; Wen et al., 2025).

We assume access to a reasoning-augmented guardrail corpus $\mathcal{D} = \{(x_i, s_i, r_i, y_i)\}_{i=1}^N$, where x_i is a user prompt, s_i is the corresponding model response, $y_i = (y_i^p, y_i^r)$ contains the final prompt and response safety labels, and

$$r_i = (r_i^1, r_i^2, \ldots, r_i^{m_i})$$

is a step-separated safety rationale.

Unlike standard label-only guardrail training, this supervision exposes the model to the reasoning underlying the final moderation decision. However, COLAGUARD does not aim to generate these rationales at inference time. Instead, the rationales serve as targets during the initial stages of training so that the model can later compress the deliberation process into latent steps.

3.3 Explicit Reasoning Warm-Up

The first stage (Stage 0) trains the model as an explicit reasoning guardrail. Given an instruction I, prompt x, response s, rationale r, and final label tuple y, the model is optimized to generate structured safety-relevant rationale followed by the final safety labels:

$$\mathcal{L}_{\text{warm}} = -\mathbb{E}_{(x,s,r,y)\sim\mathcal{D}} \log p_\theta(r, y \mid I, x, s).$$

This warm-up follows the explicit reasoning guardrail paradigm, where models learn to verbalize intermediate safety reasoning before predicting final moderation labels (Liu et al., 2025; Wen et al., 2025). We denote the resulting model as G_θ^0, from which subsequent stages progressively replace explicit rationale steps with latent recurrent steps.

3.4 Dual-Mode Latent Recurrence

To internalize reasoning, COLAGUARD switches between two modes. In language mode, the model consumes standard token embeddings and predicts the next token autoregressively. In latent mode, the model does not consume a standard token embedding; instead, the previous hidden state is fed back as the next input representation.

Let $e(\cdot)$ denote the token embedding function and let $h_t \in \mathbb{R}^d$ be the last-layer hidden state at

position t. For a sequence with a latent span beginning at position a and ending at position b, vanilla latent recurrence replaces the input embedding at each latent position with the previous hidden state:

$$E_t = \begin{cases} e(w_t), & t < a \text{ or } t > b, \\ h_{t-1}, & a \leq t \leq b, \end{cases}$$

where w_t is the discrete token at position t outside the latent span. This follows the chain-of-continuous-thought formulation introduced by Hao et al. (2025), allowing the model to perform recurrent computation in continuous latent space rather than generating intermediate rationale tokens.

While this latent recurrence lays the foundation of COLAGUARD, directly feeding contextual hidden states back into a pretrained transformer creates a distribution mismatch since the base model is trained to consume token embeddings, while h_{t-1} is a hidden representation. To reduce the hidden-state/token-embedding mismatch observed in latent recurrence, we adopt context-prediction fusion from Latent Thoughts Tuning (Liu et al., 2026). At each latent position, the model first computes a predictive embedding from the next-token distribution induced by the previous hidden state:

$$e_{\text{pred}}(h_{t-1}) = \sum_{v \in \mathcal{V}_p} \tilde{p}_\theta(v \mid h_{t-1}) e(v),$$

where $\mathcal{V}_p$ is the nucleus-filtered vocabulary set, and $\tilde{p}_\theta(v \mid h_{t-1})$ is the renormalized probability distribution over this set. Structural latent-control tokens are excluded from this distribution.

The recurrent input is then constructed by fusing the contextual hidden state with the predictive embedding:

$$\tilde{e}_t = \alpha h_{t-1} + (1 - \alpha) e_{\text{pred}}(h_{t-1}),$$

where $\alpha \in [0, 1]$ controls the balance between contextual continuity and semantic anchoring. Finally, a lightweight projection module maps the fused representation back into the model input space:

$$\mathbf{e}_t^{\text{in}} = \begin{cases} h_{t-1}, & \alpha = 1, \\ g_\phi(\tilde{e}_t), & \alpha < 1 \text{ and adapter is used}, \\ \tilde{e}_t, & \alpha < 1 \text{ and no adapter is used}. \end{cases}$$

where g_ϕ is a trainable adapter. When $\alpha = 1$, this reduces to vanilla hidden-state recurrence; when $\alpha < 1$, the recurrent state is anchored by predictive information from the vocabulary embedding.

3.5 Stage-Wise Internalization

The core of COLAGUARD is a stage-wise curriculum that progressively replaces natural-language rationale steps with recurrent latent steps. The staged replacement schedule follows prior internalization curricula showing that gradually replacing explicit reasoning tokens is more stable than removing rationales all at once (Deng et al., 2023, 2025; Hao et al., 2025).

For an example with m rationale steps, we write the step-separated rationale as

$$r = (r^1, r^2, \ldots, r^m),$$

let K denote the maximum number of reasoning steps represented by the latent budget, and define $\ell_k = \min(k, K)$. At stage k, the first k rationale steps are removed and replaced with $\ell_k c$ latent positions:

$$(r^1, \ldots, r^k) \rightarrow (z_1, \ldots, z_{\ell_k c}),$$

where c is the number of latent positions allocated per replaced reasoning step within the latent budget. We denote the resulting training sequence as $q^{(k)}$, which contains the instruction, prompt, response, latent span, any remaining rationale steps $r^{k+1}, \ldots, r^m$, and the final labels y.

If $k \geq m$, the rationale is fully replaced and the final label tuple follows the latent span directly; however, because the latent budget is fixed, examples with more than K rationale steps may still contain explicit rationale tokens after the maximum latent stage is reached. We therefore include a final compression stage that keeps the latent span fixed at Kc positions while removing all remaining rationale steps:

$$r^{1:m} \rightarrow (z_1, \ldots, z_{Kc}).$$

This extra stage enables the absorption of residual explicit reasoning signal into the fixed latent recurrence rather than remaining decoded as text.

Training optimizes only the remaining language tokens and final labels. The prompt, response, latent-control tokens, and latent positions are masked from the language-modeling loss. Let $\mathcal{M}^{(k)}$ be the set of supervised token positions in $q^{(k)}$. The internalization objective is

$$\mathcal{L}_{\text{int}}^{(k)} = -\mathbb{E}_{(x,s,r,y)\sim\mathcal{D}} \sum_{t \in \mathcal{M}^{(k)}} \log p_\theta(q_t^{(k)} \mid q_{<t}^{(k)}).$$

Benchmark	Samples
Prompt Harmfulness Detection	
ToxicChat (Lin et al., 2023)	2,853
OpenAI Moderation (Markov et al., 2023)	1,680
Aegis Safety Test (Ghosh et al., 2024)	359
HarmBench (Mazeika et al., 2024)	239
WildGuardTest (Han et al., 2024)	1,756
Response Harmfulness Detection	
HarmBench (Mazeika et al., 2024)	602
SafeRLHF (Ji et al., 2024)	2,000
BeaverTails (Ji et al., 2023)	3,021
XSTest (Röttger et al., 2024)	446
WildGuardTest (Han et al., 2024)	1,768

Table 1: Evaluation benchmarks for prompt and response harmfulness detection.

The same masked language-modeling objective is used for the final compression stage, with supervised positions restricted to the final safety-label tokens.

As k increases, less of the original rationale remains in language space, forcing more of the safety decision process to be represented by latent recurrence. This objective gives the latent positions no direct textual target so that the latent states are optimized through their downstream ability to predict the remaining rationale steps and the final safety labels.

3.6 Efficient Inference

At inference time, CoLaGuard receives only the instruction, prompt, and response. It appends a fixed latent span and performs recurrent latent computation using the fused update in §3.4. Let $Z_L = [\langle\text{start-latent}\rangle, z_1, \ldots, z_L, \langle\text{end-latent}\rangle]$ denote the fixed latent span, where L is the latent budget used at deployment. After the latent span, the model returns to language mode and autoregressively predicts the final prompt and response safety labels:

$$(\hat{y}^p, \hat{y}^r) = \arg \max_{(y^p, y^r) \in Y^2} p_\theta(y^p, y^r \mid I, x, s, Z_L).$$

Because CoLaGuard does not generate natural-language rationales, its inference cost scales with the number of latent positions rather than the length of an explicit chain-of-thought.

4 Experiments

To evaluate CoLaGuard, we conduct experiments on multiple safety benchmarks, comparing (1) safety classification performance across various baselines and (2) inference efficiency against explicit reasoning guardrails.

4.1 Experimental Setup

Reasoning Augmented Dataset. We use the GuardReasonerTrain dataset (Liu et al., 2025) as the primary training source for our guardrail model. GuardReasonerTrain is a 127,000-example reasoning-augmented compilation of the following safety-focused datasets: WildGuard (Han et al., 2024), AegisSafety (Ghosh et al., 2024), BeaverTails (Ji et al., 2023), and ToxicChat (Lin et al., 2023). Each example comes with a prompt, composed of the guardrail instructions, a user input, and a user output; a multi-step reasoning trace separated into the three tasks of request moderation, refusal detection, and response moderation; and ground-truth answers for the three tasks. Using the same reasoning-augmented supervision source as GuardReasoner allows us to directly compare explicit rationale generation against latent internalization under a matched training signal.

Both iCoT (Deng et al., 2025) and Coconut (Hao et al., 2025) show that replacing too many language tokens per stage can destabilize training. We therefore split reasoning traces into smaller step-level replacements, but this increases the number of training stages and overall computational cost. To reduce cost and focus on request and response safety moderation, we remove the refusal task from CoLaGuard training supervision.

Training Details. We use separate training configurations for the explicit CoT warm-up stage and the latent internalization stages. In Stage 0, we fully fine-tune Llama 3.1 8B on GuardReasonerTrain to obtain an explicit reasoning baseline. Training is performed on 8×A100 (80GB) GPUs for 3 epochs, with per-device batch size 1, gradient accumulation 32, AdamW optimization (Loshchilov and Hutter, 2019), a cosine learning-rate schedule, and an initial learning rate of 5×10^{-5}. The fusion coefficient is set to $\alpha = 1.0$ in this stage, so the fusion module is inactive.

Starting from the Stage-0 checkpoint, we then train the stage-wise internalization curriculum. Since roughly 80% of GuardReasonerTrain examples contain at most six reasoning steps, we use six latent recurrent steps as the fixed inference budget. Each internalization stage replaces one additional reasoning step with latent states, and a final

Table 2: F1 Score (%) of Models on 5 Benchmarks of Prompt Harmfulness Detection. **Bold** and underlined values denote the best and runner-up. "–" denotes the result is unavailable.

Method	Model Size	ToxicChat	HarmBench	OpenAI Mod.	Aegis SafetyTest	WildGuard Test	Macro Avg	Micro Avg
Closed-Source Guard API								
GPT-4o	Unknown	64.46	82.27	62.26	81.07	80.87	74.19	69.59
GPT-4o+CoT	Unknown	73.43	81.98	76.78	88.24	82.75	80.64	77.69
o1-preview	Unknown	57.69	89.61	74.60	83.15	76.31	76.27	69.00
Open-Source Guard Model								
LLaMA Guard	7B	61.60	67.20	75.80	74.10	56.00	66.94	64.48
LLaMA Guard 2	8B	47.10	94.00	76.10	71.80	70.90	71.98	63.16
LLaMA Guard 3	8B	53.12	**98.94**	79.69	**99.50**	68.47	79.94	67.52
Aegis Guard Defensive	7B	70.00	77.70	67.50	84.80	78.50	75.70	72.60
Aegis Guard Permissive	7B	73.00	70.50	74.70	82.90	71.50	74.52	73.46
Aegis Guard 2.0	8B	–	–	81.00	–	81.60	–	–
ShieldGemma	2B	6.91	11.81	13.89	7.47	9.36	9.89	9.44
ShieldGemma	9B	67.92	67.96	78.58	77.63	57.74	69.97	68.43
WildGuard	7B	70.80	98.90	72.10	89.40	88.90	84.02	77.68
QwQ-preview	32B	34.81	86.73	61.58	80.23	66.02	65.87	53.47
GuardReasoner	1B	72.43	96.31	70.06	89.34	87.37	83.10	77.37
GuardReasoner	3B	78.20	89.10	71.87	91.39	89.01	83.91	80.48
GuardReasoner	8B	**78.79**	91.86	72.00	90.18	89.17	**84.40**	**80.83**
Latent Reasoning Guardrail (Ours)								
CoLaGuard (Ours)	3B	75.27	94.25	73.15	90.58	88.15	84.28	79.49
CoLaGuard (Ours)	8B	75.26	93.54	73.45	89.45	**89.44**	84.23	79.77

compression stage removes any remaining explicit reasoning for longer traces while preserving the same six-step latent budget. Each stage is trained for one epoch with a reset AdamW optimizer and a constant learning rate of 1×10^{-5}.

During internalization, we linearly anneal the fusion coefficient from $\alpha = 1.0$ to $\alpha = 0.6$ over the first 200 warm-up steps. We set the fusion temperature to 1.0, top-p to 0.9, and use a fusion adapter with hidden dimension 1024. All training is conducted in bf16 precision, and checkpoints are saved after each stage.

For the 3B model, we use Llama 3.2 3B as the backbone and set the internalization learning rate to 2×10^{-5}, which we found more stable for this scale. Following the implementation choice in Liu et al. (2026), we disable the fusion adapter for the 3B model because Llama 3.2 3B uses tied input-output embeddings. All other training settings are kept identical to the 8B configuration.

Safety Evaluation. To assess the performance and efficiency of our guardrail model while isolating the effect of latent reasoning against explicit rationale generation under matched supervision, we evaluate on benchmarks used by Liu et al. (2025) (Table 1) and use GuardReasoner (SFT-only, without hard-sample DPO) as our primary explicit reasoning baseline. More details on these benchmarks can be found in Appendix A.

We compare CoLaGuard against 20 baselines spanning closed-source APIs, open-source guard models, and our primary explicit reasoning baseline. Baseline names and model sizes are reported in Tables 2 and 3; corresponding references are provided in Appendix A.2.

4.2 Results

Overall Classification Performance. Tables 2 and 3 report F1 scores on prompt and response harmfulness detection. CoLaGuard 8B is comparable to GuardReasoner 8B, with prompt macro-F1 of 84.23 vs. 84.40 and response macro-F1 of 83.33 vs. 83.13. Compared with Llama Guard 3, it improves the average macro-F1 across both tasks by 8.24 points while avoiding explicit rationale generation.

At the benchmark level, CoLaGuard 8B achieves the best F1 on WildGuardTest for both prompt and response detection (89.44 and 81.23), and ranks second on HarmBench response and SafeRLHF (86.38 and 70.49). Its lower prompt micro-F1 relative to GuardReasoner 8B (79.77 vs. 80.83) is mainly due to ToxicChat, which accounts for 41.4% of the prompt evaluation set and therefore has a large effect on the micro average.

Table 3: F1 Score (%) of Models on 5 Benchmarks of Response Harmfulness Detection. **Bold** and underlined values denote the best and runner-up. "–" denotes the result is unavailable.

Method	Model Size	HarmBench	SafeRLHF	BeaverTails	XSTest Response	WildGuard Test	Macro Avg	Micro Avg
Closed-Source Guardrail API								
GPT-4o	Unknown	56.34	64.05	78.63	65.12	65.24	65.88	69.41
GPT-4o+CoT	Unknown	65.99	65.10	82.26	86.90	71.43	74.34	74.45
o1-preview	Unknown	76.40	66.60	79.96	74.75	50.00	69.54	69.22
Open-Source Guardrail								
LLaMA Guard	7B	52.00	48.40	67.10	82.00	50.50	60.00	58.27
LLaMA Guard 2	8B	77.80	51.60	71.80	90.80	66.50	71.70	66.99
LLaMA Guard 3	8B	85.07	44.36	67.84	87.67	70.80	71.15	64.97
Aegis Guard Defensive	7B	62.20	59.30	74.70	52.80	49.10	59.62	62.79
Aegis Guard Permissive	7B	60.80	55.90	73.80	60.40	56.40	61.46	63.55
Aegis Guard 2.0	8B	–	–	–	86.20	77.50	–	–
ShieldGemma	2B	35.36	16.92	30.97	65.55	20.13	33.79	27.24
ShieldGemma	9B	56.44	47.07	63.61	73.86	47.00	57.60	55.67
HarmBench LLaMA	13B	84.30	60.00	77.10	64.50	45.70	66.32	65.49
HarmBench Mistral	7B	**87.00**	52.40	75.20	72.00	60.10	69.34	66.70
MD-Judge	7B	81.60	64.70	86.70	90.40	76.80	80.04	78.67
BeaverDam	7B	58.40	**72.10**	**89.90**	83.60	63.40	73.48	76.60
WildGuard	7B	86.30	64.20	84.40	**94.70**	75.40	81.00	77.95
QwQ-preview	32B	69.65	62.76	77.26	45.95	17.56	54.64	57.73
GuardReasoner	1B	84.75	68.39	85.84	90.12	74.81	80.78	79.06
GuardReasoner	3B	85.66	69.02	86.72	91.36	79.70	82.49	80.80
GuardReasoner	8B	85.47	70.04	87.60	94.34	78.20	83.13	81.22
Latent Reasoning Guardrail (Ours)								
CoLaGuard	3B	86.36	68.72	86.29	94.19	77.23	82.56	80.22
CoLaGuard	8B	86.38	70.49	86.55	92.02	**81.23**	**83.33**	**81.55**

Table 4: Inference Efficiency and Performance Comparison. We report inference time, completion token cost, and combined macro-F1 (averaged across prompt and response moderation). Inference is conducted on 1×H100 (80GB) GPU. LlamaGuard-3-8B is a single-pass, non-reasoning classifier included as a lower bound on latency and token cost.

Metric	3B		8B		
	GuardReasoner	CoLaGuard	GuardReasoner	CoLaGuard	LlamaGuard-3
Time Cost (ms/query)	3801.03	318.9	4407.8	342.0	110.56
Token Cost (token/query)	281.96	13.0	289.4	12.9	3.67
Macro-F1 (Combined)	83.20	83.42	83.77	83.78	75.55

Model Size Comparison. CoLaGuard 3B is already competitive with GuardReasoner 3B, slightly improving both prompt macro-F1 (84.28 vs. 83.91) and response macro-F1 (82.56 vs. 82.49). Scaling to 8B mainly benefits response detection and yields better combined averages (83.78 vs. 83.42 macro; 80.66 vs. 79.86 micro), suggesting a modest but more consistent gain from the larger backbone.

Inference Efficiency. Table 4 shows that CoLaGuard substantially reduces inference cost compared with GuardReasoner. At 8B, latency drops from 4,407.8 to 342.0 ms/query, a 12.9× speedup, while token usage decreases from 289.4 to 12.9 tokens/query, a 22.4× reduction, with combined macro-F1 held essentially constant (83.77 vs. 83.78). These gains come from replacing long autoregressive CoT generation with a fixed six-step latent recurrence. We additionally report LlamaGuard-3-8B, a single-pass non-reasoning classifier, as a lower bound on latency and token cost. It is faster than CoLaGuard (110.56 ms/query, 3.67 tokens/query) but trails its combined macro-F1 by 8.24 points (75.55 vs. 83.78), confirming that CoLaGuard's efficiency gains reflect a genuine accuracy trade-off against reasoning-based guardrails rather than a shortcoming relative to non-reasoning classifiers.

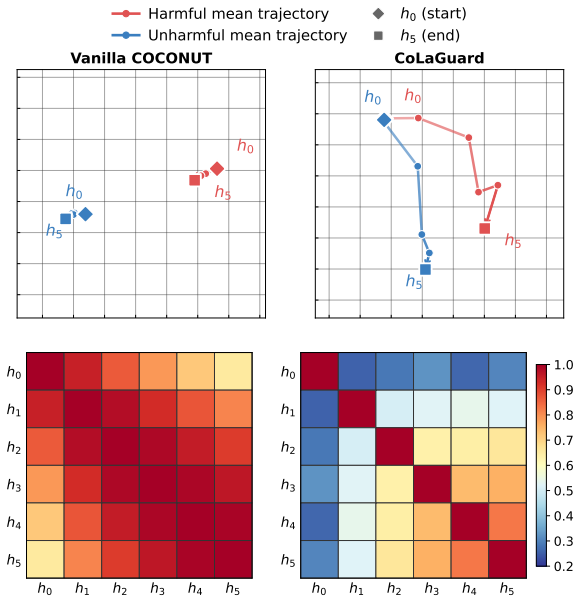

Figure 2: Geometric Analysis of Latent Representations. *(Top)* UMAP of mean harmful/unharmful trajectories across recurrence steps h_0–h_5. *(Bottom)* Intra-sample cosine similarity heatmap between latent steps. Vanilla Coconut shows highly similar latent states and early label separation, while CoLaGuard exhibits progressive class differentiation across recurrence steps.

4.3 Ablation Studies

Analyzing Latent Recurrence Dynamics. Recent work questions whether latent tokens in Coconut-style recurrence perform meaningful computation beyond acting as learned placeholders. Zhang et al. (2025) find that vanilla Coconut tokens form clustered embeddings with limited input sensitivity, suggesting placeholder behavior from learned shortcuts. Liu et al. (2026) show that Context-Prediction Fusion mitigates inter-sample representational collapse, suggesting more expressive latent states. We extend this analysis to CoLaGuard through WildGuardTest latent trajectories and a full-suite Context-Prediction-Fusion ablation against vanilla Coconut.

Figure 2 shows the average pairwise cosine similarity between latent steps (h_i, h_j) across samples and mean harmful/unharmful trajectories via UMAP (McInnes et al., 2020). Vanilla Coconut exhibits uniformly high cross-step similarity, consistent with early commitment to a fixed latent state that is simply propagated forward; its harmful and unharmful trajectories are already separated at h_0, with limited additional separation in later steps. In contrast, CoLaGuard shows noticeably lower cross-step similarity, indicating that its latent states

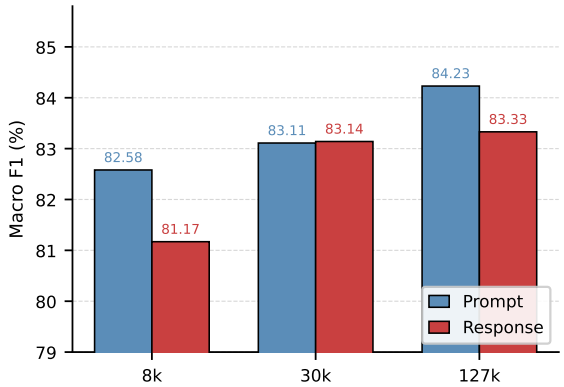

Figure 3: Training Data Scaling. CoLaGuard 8B prompt and response macro-F1 across training data sizes.

continue to evolve throughout the recurrence rather than collapsing after the initial step. Its trajectories begin closer together and diverge progressively, suggesting that recurrence contributes to the refinement of safety-relevant representations rather than simply preserving an early decision.

As an ablation of Context-Prediction Fusion, a vanilla Coconut guardrail with the same six-step latent budget reaches 81.82 combined macro-F1 and 79.78 combined micro-F1, compared with 83.78 and 80.72 for CoLaGuard. Context-Prediction Fusion yields clear gains that bring it to parity with the explicit reasoning baseline (+1.96 macro-F1, +0.94 micro-F1), suggesting that the more progressive latent shifts in Figure 2 may be relevant to downstream moderation performance.

Furthermore, the complementary interventional check in Appendix A.3 reports a latent-budget sweep on the same checkpoint, showing that CoLaGuard's combined macro-F1 increases monotonically with the number of latent steps used at inference, from 80.92 at zero steps to 83.78 at the full six-step budget.

Scaling Training Data. Figure 3 shows that CoLaGuard 8B improves consistently with more reasoning-augmented training data. Response macro-F1 improves sharply from 8k to 30k examples (+1.97 points) but shows limited additional gain at 127k (+0.19 points). Prompt macro-F1 increases more gradually, gaining 0.53 points from 8k to 30k and 1.12 points from 30k to 127k.

These trends suggest that response moderation benefits earlier from diverse supervision, while prompt moderation continues to improve at larger scale. Overall, the results show that CoLaGuard scales reliably with training data and achieves its

best performance with the full GuardReasonerTrain corpus.

5 Conclusion

We introduced CoLaGuard, a latent reasoning guardrail that internalizes explicit safety reasoning through a stage-wise curriculum. Across prompt and response harmfulness detection benchmarks, CoLaGuard matches the average macro-F1 of an explicit reasoning guardrail while substantially reducing inference cost. CoLaGuard 8B matches GuardReasoner 8B in macro-F1 while achieving 12.9× lower latency and 22.4× fewer tokens. These results show that latent reasoning is a practical path toward safety guardrails that are both robust and efficient for deployment.

Limitations

While CoLaGuard demonstrates strong efficiency and competitive safety performance, several limitations remain. First, our evaluation focuses on text-based prompt and response harmfulness detection, leaving broader policy taxonomies, multilingual inputs, multimodal content, adaptive adversaries, and long-horizon agent behavior for future work. Second, CoLaGuard is trained from distilled reasoning traces and may inherit biases or coverage gaps from the underlying supervision, which latent internalization may make harder to detect than in models that still verbalize their reasoning. Finally, although our latent representation analysis suggests progressive safety-relevant refinement, more causal interventions are needed to fully characterize how each latent step contributes to the final decision to improve interpretability of safety decisions.

Ethics Statement

The aim of this work is to improve the reliability and efficiency of LLM safety guardrails. While latent reasoning moderation may make strong safety filters more practical in high-traffic settings, these guardrails can still produce false positives and false negatives on ambiguous or context-dependent inputs. Therefore, CoLaGuard itself should not be considered a replacement for human oversight in real-world deployment, but rather should be used as part of a broader moderation system. The safety data used in the evaluation and training processes may contain harmful or sensitive content and should be handled with appropriate access controls and annotator-care practices.

References

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. 2024. Hopping too late: Exploring the limitations of large language models on multi-hop queries. *Preprint*, arXiv:2406.12775.

Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. 2025. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *Preprint*, arXiv:2505.16782.

Jeffrey Cheng and Benjamin Van Durme. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *Preprint*, arXiv:2412.13171.

Yuntian Deng, Yejin Choi, and Stuart Shieber. 2025. From explicit CoT to implicit CoT: Learning to internalize CoT step by step. In *The Thirteenth International Conference on Learning Representations*.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. 2023. Implicit chain of thought reasoning via knowledge distillation. *Preprint*, arXiv:2311.01460.

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

Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. 2025. AEGIS2.0: A diverse AI safety dataset and risks taxonomy for alignment of LLM guardrails. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5992–6026, Albuquerque, New Mexico. Association for Computational Linguistics.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*.

Aaron Grattafiori and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. 2024. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in neural information processing systems*, 37:8093–8131.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. 2025. Training large language model to reason in a continuous latent space.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *Preprint*, arXiv:2312.06674.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Juntao Dai, Boren Zheng, Tianyi Qiu, Jiayi Zhou, Kaile Wang, Boxuan Li, Sirui Han, Yike Guo, and Yaodong Yang. 2024. PKU-SafeRLHF: Towards multi-level safety alignment for LLMs with human preference. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of LLM via a human-preference dataset. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Mintong Kang and Bo Li. 2025. r^2-guard: Robust reasoning enabled LLM guardrail via knowledge-enhanced logical reasoning. In *The Thirteenth International Conference on Learning Representations*.

Seungone Kim, Se June Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. *Preprint*, arXiv:2305.14045.

Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. 2024. Salad-bench: A hierarchical and comprehensive safety benchmark for large language models. *Preprint*, arXiv:2402.05044.

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

Weihao Liu, Dehai Min, and Lu Cheng. 2026. Latent thoughts tuning: Bridging context and reasoning with fused information in latent tokens. *Preprint*, arXiv:2602.10229.

Yue Liu, Hongcheng Gao, Shengfang Zhai, Yufei He, Jun Xia, Zhengyu Hu, Yulin Chen, Xihong Yang, Jiaheng Zhang, Stan Z. Li, Hui Xiong, and Bryan Hooi. 2025. Guardreasoner: Towards reasoning-based llm safeguards. *Preprint*, arXiv:2501.18492.

Llama Team. 2024. Meta Llama guard 2. https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Todor Markov, Chong Zhang, Sandhini Agarwal, Tyna Eloundou, Teddy Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. *Preprint*, arXiv:2208.03274.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *Preprint*, arXiv:2402.04249.

Leland McInnes, John Healy, and James Melville. 2020. Umap: Uniform manifold approximation and projection for dimension reduction. *Preprint*, arXiv:1802.03426.

NVIDIA. 2025. Nemotron Content Safety Reasoning 4B. https://huggingface.co/nvidia/Nemotron-Content-Safety-Reasoning-4B.

OpenAI. 2024. OpenAI o1 system card. https://openai.com/index/openai-o1-system-card/.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Jacob Pfau, William Merrill, and Samuel R. Bowman. 2024. Let's think dot by dot: Hidden computation in transformer language models. *Preprint*, arXiv:2404.15758.

Qwen Team. 2024. QwQ: Reflect deeply on the boundaries of the unknown. https://qwenlm.github.io/blog/qwq-32b-preview/.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Makesh Narsimhan Sreedhar, Traian Rebedea, and Christopher Parisien. 2025. Safety through reasoning: An empirical study of reasoning guardrail models. *Preprint*, arXiv:2505.20087.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Xiaofei Wen, Wenxuan Zhou, Wenjie Jacky Mo, and Muhao Chen. 2025. Thinkguard: Deliberative slow thinking leads to cautious guardrails. *arXiv preprint arXiv:2502.13458*.

Yahan Yang, Soham Dan, Shuo Li, Dan Roth, and Insup Lee. 2025. MrGuard: A multilingual reasoning guardrail for universal LLM safety. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 27377–27396, Suzhou, China. Association for Computational Linguistics.

Zhuowen Yuan, Zidi Xiong, Yi Zeng, Ning Yu, Ruoxi Jia, Dawn Song, and Bo Li. 2024. RigorLLM: Resilient guardrails for large language models against undesired content. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman. 2024. Quiet-STar: Language models can teach themselves to think before speaking. In *First Conference on Language Modeling*.

Wenjun Zeng, Yuchi Liu, Ryan Mullins, Ludovic Peran, Joe Fernandez, Hamza Harkous, Karthik Narasimhan, Drew Proud, Piyush Kumar, Bhaktipriya Radharapu, Olivia Sturman, and Oscar Wahltinez. 2024. Shieldgemma: Generative ai content moderation based on gemma. *Preprint*, arXiv:2407.21772.

Yuyi Zhang, Boyu Tang, Tianjie Ju, Sufeng Duan, and Gongshen Liu. 2025. Do latent tokens think? a causal and adversarial analysis of chain-of-continuous-thought. *Preprint*, arXiv:2512.21711.

Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, Baosong Yang, Chen Cheng, Jialong Tang, Jiandong Jiang, Jianwei Zhang, Jijie Xu, Ming Yan, Minmin Sun, Pei Zhang, and 24 others. 2025. Qwen3guard technical report. *Preprint*, arXiv:2510.14276.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, and 14 others. 2025. A survey on latent reasoning. *Preprint*, arXiv:2507.06203.

A Safety Evaluation

A.1 Description of Benchmarks

To assess the performance and efficiency of our latent reasoning guardrail model, we evaluate it across eight unique safety-related benchmarks.

WildGuard (Han et al., 2024): WildGuardMix is a large-scale safety moderation dataset with 92,000 labeled examples that cover both normal and adversarial prompt behaviors that come coupled with corresponding refusal and compliance responses. The WildGuardTest split is human-annotated and covers 5,000 safety labeled examples.

ToxicChat (Lin et al., 2023): ToxicChat is a benchmark that includes 10,000 real user queries, leveraged as adversarial prompts for testing content moderation and toxicity detection in human-AI interactions.

Aegis Safety Test 1.0 (Ghosh et al., 2024): A dataset of approximately 11,000 manually annotated examples, Aegis Safety Test 1.0 was curated with the purpose of testing LLM safety alignment in accordance with Nvidia's content safety taxonomy.

HarmBench (Mazeika et al., 2024): HarmBench is a framework that is systematically designed to address the lack of standardized evaluation frameworks in the field of automated red teaming. By leveraging various behaviors, this framework can be used to generate red-teaming test cases for evaluating the adversarial robustness of LLMs.

OpenAI Moderation (Markov et al., 2023): A benchmark for assessing LLMs' ability to detect harmful content based on OpenAI's safety guidelines, covering violence, self-harm, and misinformation.

SafeRLHF (Ji et al., 2024): A dataset of 82,000 questions with two responses each, every entry in SafeRLHF includes safety meta-labels as well as preference between the two responses.

BeaverTails (Ji et al., 2023): The Beavertails dataset was introduced to further research on safety alignment in LLMs. The complete dataset includes over 300,000 question-answer pairs that are annotated with safety meta-labels and corresponding, violated safety categories.

XSTest (Röttger et al., 2024): Developed to evaluate refusal behaviors and identify systematic failure modes in large language models, XSTest is comprised of 250 safe prompts across ten prompt types and contrasting 200 unsafe prompts that human-aligned models should refuse.

Baseline	Reference	Model Size
GPT-4o	OpenAI (2024)	Unknown
GPT-4o + CoT	OpenAI (2024)	Unknown
o1-preview	OpenAI (2024)	Unknown
LLaMA Guard	Inan et al. (2023)	7B
LLaMA Guard 2	Llama Team (2024)	8B
LLaMA Guard 3	Grattafiori et al. (2024)	8B
Aegis Guard Defensive	Ghosh et al. (2024)	7B
Aegis Guard Permissive	Ghosh et al. (2024)	7B
Aegis Guard 2.0	Ghosh et al. (2025)	8B
ShieldGemma	Zeng et al. (2024)	2B / 9B
WildGuard	Han et al. (2024)	7B
QwQ-preview	Qwen Team (2024)	32B
HarmBench LLaMA	Mazeika et al. (2024)	13B
HarmBench Mistral	Mazeika et al. (2024)	7B
MD-Judge	Li et al. (2024)	7B
BeaverDam	Ji et al. (2023)	7B
GuardReasoner	Liu et al. (2025)	1B / 3B / 8B

Table 5: Baseline references and model sizes for Tables 2 and 3.

A.2 Baseline Details

Details are presented in Table 5.

A.3 Latent Budget Sweep

To test whether latent recurrence performs computation that is functionally load-bearing rather than acting as a fixed placeholder, we evaluate the same CoLaGuard 8B checkpoint at inference-time latent budgets of 0, 2, 4, and 6 steps. We take the six-step checkpoint and truncate the latent span at inference, allowing the model to decode safety labels after the truncated number of recurrent steps. Figure 4 reports combined macro-F1 (averaged across the prompt and response moderation benchmarks in Tables 2 and 3) at each budget, with GuardReasoner-8B and LlamaGuard-3-8B included as reference points.

Combined macro-F1 increases monotonically with step count, from 80.92 at zero latent steps to 83.78 at the full six-step budget. Notably, CoLaGuard already outperforms LlamaGuard-3-8B by 5.37 points at zero latent steps, indicating that reasoning-augmented training alone provides a strong accuracy baseline independent of latent recurrence. Each additional latent step then contributes a further, roughly monotonic gain, closing the remaining gap to GuardReasoner-8B. This provides interventional evidence, complementary to the representational analysis in Figure 2, that the latent recurrence is not redundant and that each step contributes meaningfully to the final safety decision.

Table 6 additionally reports the corresponding latency at each budget. Latency increases only modestly across the sweep, from 236.34 ms/query

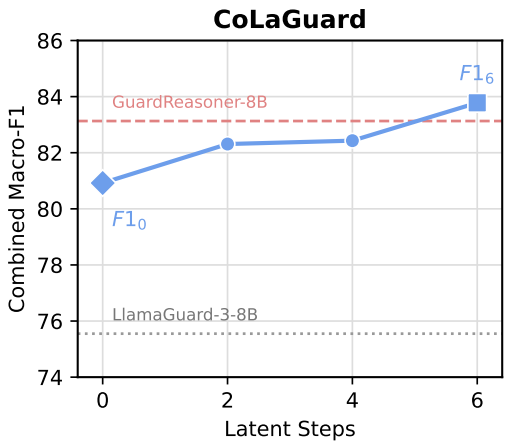

Figure 4: Latent budget sweep. Combined macro-F1 of CoLaGuard 8B increases monotonically with the number of latent recurrence steps used at inference, from 80.92 at zero steps to 83.78 at the full six-step budget, closing the gap to GuardReasoner-8B (dashed) while remaining well above the non-reasoning LlamaGuard-3-8B floor (dotted).

at zero latent steps to 342.00 ms/query at six steps, remaining an order of magnitude below GuardReasoner-8B's 4,407.8 ms/query throughout, so the accuracy gains from additional latent steps come at comparatively little additional inference cost.

Latent Steps	Time (ms/query)	Macro-F1
LlamaGuard-3-8B	110.56	75.55
0	236.34	80.92
2	255.77	82.31
4	293.63	82.43
6	342.00	83.78
GuardReasoner-8B	4407.8	83.13

Table 6: Latency and combined macro-F1 across latent budgets for CoLaGuard 8B, with LlamaGuard-3-8B and GuardReasoner-8B as reference points.