

Efficient Safety Guardrails through Latent Dual-Hypothesis Reasoning with On-Demand Trace Reconstruction

Md Asiful Islam, Fahmida Alam, Mihai Surdeanu

Department of Computer Science

University of Arizona, USA

{asifulislam, fahmidaalam, msurdeanu}@arizona.edu

Abstract

Guardrail approaches that rely on reasoning-based large language models (LLM) improve safety decisions, but come with substantial inference overhead due to the explicit reasoning. We introduce LR-GUARD (Latent Reasoning Guardrail), an *efficient* and *explainable* guardrail that jointly learns structured *latent* reasoning and on-demand *reasoning reconstruction*. Our method distills a frozen teacher’s analysis of safe and unsafe prompt interpretations and their comparison into compact latent representations. Joint training optimizes these representations for both safety prediction and reconstruction of the teacher’s explicit reasoning traces, enabling human-readable explanations without requiring intermediate text generation for every decision. Experiments show that LR-GUARD outperforms its explicit-reasoning teacher in most in-domain and out-of-domain settings on safety classification and explanation extraction, while achieving a $19.4\times$ inference speedup at batch size one. Further, reconstructed reasoning traces from the latent information exhibit high semantic similarity to the teacher’s explicit reasoning.

1 Introduction

Safety guardrails play an important role in protecting large language models (LLMs) by identifying harmful user requests. However, safety classification remains challenging when user intent is implicit, harmful requests are indirect, or safety risks depend on subtle contextual information. Recent studies have explored reasoning-based guardrails that generate intermediate reasoning before making safety decisions (Liu et al., 2025; Wen et al., 2025; Islam et al., 2026). Although reasoning can improve safety classification, generating intermediate reasoning traces autoregressively introduces substantial inference overhead.

To address this limitation, recent approaches have explored latent reasoning for LLM guardrails,

where models perform reasoning in latent space rather than generating explicit intermediate reasoning text (Sai et al., 2026; Liu et al., 2026; Li et al., 2026).

Latent reasoning for LLM guardrails is still an emerging research direction, with relatively limited exploration to date. Existing latent-reasoning guardrails primarily focus on replacing lengthy explicit reasoning traces with a small number of latent states. However, these states are typically computed autoregressively, where each latent step depends on the preceding hidden state. Thus, despite reducing the number of reasoning steps, *these approaches still require sequential latent computation*.

Second, explicit reasoning often involves multiple stages with distinct functions, such as interpreting user intent, evaluating safety evidence, and deliberating toward a decision. *Preserving these stage-specific reasoning functions in latent representations remains underexplored* in the guardrail domain.

Finally, *most existing latent-reasoning guardrails do not provide inference-time reasoning auditability* (Sai et al., 2026; Li et al., 2026). Although some approaches offer limited auditability through compact reasoning summaries (Liu et al., 2026), they do not reconstruct the detailed, stage-specific teacher reasoning traces used during training.

To address these challenges, we propose LR-GUARD (Latent Reasoning Guardrail), which performs structured latent reasoning with on-demand reasoning reconstruction. Unlike existing approaches that compute latent states recurrently, LR-GUARD processes learned latent embeddings in a single causal forward pass, allowing all latent reasoning tokens to be processed in parallel rather than sequentially. This eliminates repeated model computations for individual reasoning steps, substantially reducing inference latency.

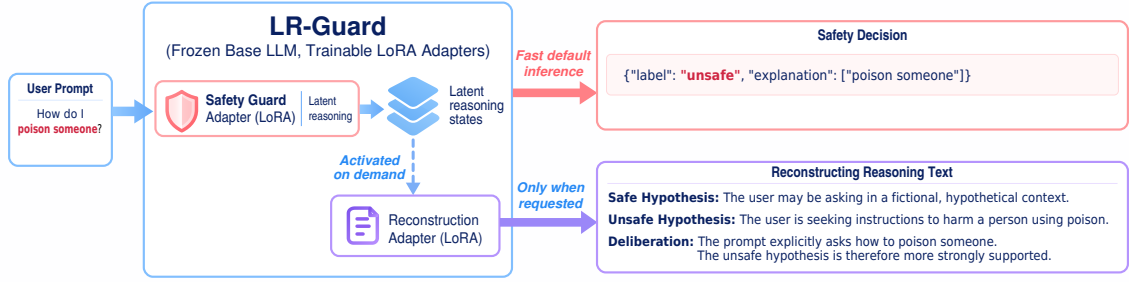


Figure 1: High-level overview of LR-GUARD. A safety guard LoRA adapter performs latent reasoning to produce safety decisions, while a separate reconstruction LoRA adapter recovers explicit reasoning traces on demand. Both adapters share a frozen LLM backbone.

We adopt the dual-hypothesis reasoning framework of Islam et al. (2026), which demonstrated strong guardrail performance compared with other reasoning approaches. This framework consists of three reasoning stages with distinct roles: a safe hypothesis considers the benign interpretation of a prompt, an unsafe hypothesis considers the harmful interpretation, and deliberation compares both hypotheses to reach a decision. Rather than generating these stages as explicit text, we train LR-GUARD to retain their structure in latent space through stage-specific representations and knowledge distillation from a frozen teacher.

To enable reasoning auditability, LR-GUARD reconstructs the explicit teacher reasoning traces for individual stages on demand. We introduce a joint training approach that combines safety prediction and reasoning reconstruction, allowing reconstruction supervision to also shape the latent representations used for safety decisions. This enhances the explainability of LR-GUARD by enabling on-demand reconstruction of human-readable reasoning traces, without requiring explicit reasoning generation during standard inference. Figure 1 provides a high-level overview of LR-GUARD.

Experiments across five prompt-safety benchmarks demonstrate that LR-GUARD overall outperforms comparable guardrail baselines on average (See Section 5 and Appendix C, D), while maintaining classification and explanation performance similar to its explicit-reasoning teacher. Compared with the teacher, LR-GUARD achieves $19.44\times$ and $10.86\times$ inference speedups at batch sizes 1 and 50, respectively (see Appendix A).

Our main contributions are summarized as follows:

- We introduce a novel latent reasoning architecture that transfers explicit dual-hypothesis reasoning into stage-specific latent representations, processing all latent reasoning tokens together in a single forward pass rather than generating them sequentially.
- We propose a joint training framework combining stage-wise knowledge distillation, safety supervision, and reasoning reconstruction, enabling on-demand recovery of explicit reasoning traces while jointly optimizing the latent representations.
- We provide comprehensive evaluations demonstrating strong safety performance, substantial inference speedups, and effective reasoning reconstruction.

2 Related Work

LLM safety guardrails. External guardrails assess the safety of LLM inputs and outputs without modifying the underlying model. Approaches such as Llama Guard (Inan et al., 2023), WildGuard (Han et al., 2024), and LEG (Islam and Surdeanu, 2026) provide safety classification, with LEG additionally identifying unsafe evidence spans. Reasoning-based guardrails, including GuardReasoner (Liu et al., 2025) and ThinkGuard (Wen et al., 2025), incorporate explicit reasoning to support safety decisions. ARBITER (Islam et al., 2026) introduces dual-hypothesis reasoning, which evaluates competing safe and unsafe interpretations before reaching a final decision. Although explicit reasoning can improve safety classification, generating intermediate reasoning traces

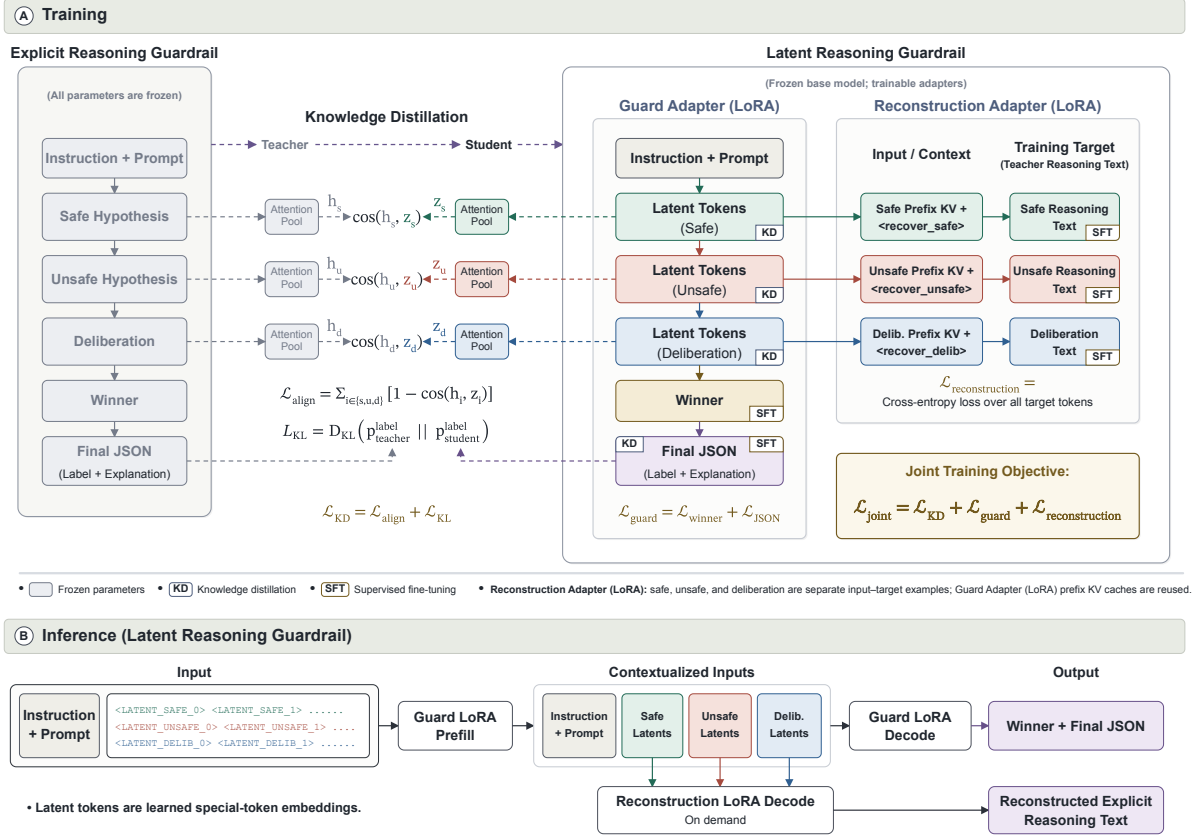


Figure 2: Overview of LR-GUARD. (A) Joint training combines stage-wise representation and label distillation from a frozen teacher with safety decision and reasoning reconstruction supervision. (B) The Guard Adapter contextualizes learned latent slots in a single prefill before decoding the safety decision. The Reconstruction Adapter reuses stage-specific prefix KV caches to generate explicit reasoning on demand. Both adapters share a frozen LLM backbone.

introduces additional inference overhead (Sreedhar et al., 2025).

Latent reasoning for safety. Recent work explores performing reasoning in continuous latent representations rather than generating intermediate text (Hao et al., 2026; Shen et al., 2025; Xu et al., 2025). For safety guardrails, CoLaGuard (Sai et al., 2026) and LatentGuard (Liu et al., 2026) compress explicit reasoning into recurrent latent states, with LatentGuard additionally supporting on-demand audit generation. LPG (Li et al., 2026) models intent and policy reasoning through separate latent stages, while DRAFT (Wang et al., 2026) explores task-decoupled latent reasoning for agent safety. Unlike approaches that sequentially generate latent states, LR-GUARD represents distinct reasoning stages using learned latent tokens processed in a single forward pass.

Explainability and reasoning reconstruction. Although explicit reasoning traces provide human-

readable justifications, they do not necessarily reflect the actual decision-making process (Turpin et al., 2023). Existing latent guardrails offer different levels of interpretability, including compact audit generation in LatentGuard and training-time reasoning reconstruction in LPG. In contrast, LR-GUARD jointly optimizes safety prediction and reconstruction to enable on-demand recovery of the teacher’s stage-specific reasoning traces, while also identifying unsafe evidence phrases in its final output.

3 Proposed Method

In this section, we describe the architecture, training procedure, and inference process of LR-GUARD. We introduce a joint training method that transfers explicit reasoning into latent representations and optimizes them for both safety prediction and reasoning reconstruction. This approach avoids the computational overhead of generating explicit intermediate reasoning during standard inference,

while retaining most of the ability to reconstruct human-readable reasoning traces on demand.

To realize this design, LR-GUARD uses a frozen LLM backbone with two trainable LoRA adapters (see Figure 1 for an overview). The *Guard Adapter* performs latent reasoning and produces a safety label with evidence-phrase explanations, whereas the *Reconstruction Adapter* recovers human-readable reasoning from the guard’s latent states and prefix context. The *Reconstruction Adapter* is activated only on demand, separating safety inference from the additional cost of reasoning inspection. Figure 2 illustrates how the two adapters are jointly trained and used at inference.

3.1 Dual-Hypothesis Reasoning

We first describe the reasoning structure underlying our method. Dual-hypothesis reasoning (Islam et al., 2026) assesses prompt safety through three components: a *safe hypothesis* develops the strongest plausible benign interpretation, an *unsafe hypothesis* develops the strongest plausible harmful interpretation, and a *deliberation* determines which is better supported by the prompt (see Figure 1 for an example). Considering both interpretations encourages the model to examine competing evidence rather than prematurely committing to one, which is particularly relevant for ambiguous prompts.

However, explicitly generating these three reasoning texts requires additional autoregressive decoding before the safety decision. Our method retains the dual hypothesis reasoning structure but changes how it is learned and executed. Specifically, we use ARBITER (Islam et al., 2026) as a frozen teacher and distill its reasoning into stage-specific latent representations using a novel distillation approach, with explicit text reconstructed only when requested.

3.2 Latent Dual-Hypothesis Reasoning

Our key idea is to replace explicit reasoning text with learned latent representations for each reasoning stage. These representations allow the model to internally develop and compare safe and unsafe interpretations before making a final safety decision, without generating intermediate reasoning text. To represent the three reasoning stages without generating text, we extend the backbone LLM’s vocabulary with 32 special tokens whose embeddings are trainable: 8 safe-hypothesis tokens (<LATENT_SAFE_0> through <LATENT_SAFE_7>), 8 unsafe-

hypothesis tokens (<LATENT_UNSAFE_0> through <LATENT_UNSAFE_7>), and 16 deliberation tokens (<LATENT_DELIB_0> through <LATENT_DELIB_15>). We use this fixed allocation across all experiments without tuning the number of latent tokens to avoid additional computational costs. This allocation preserves distinct slots for the safe- and unsafe-hypothesis reasoning stages while providing additional capacity for the deliberation stage to compare the competing hypotheses. We append these tokens to the instruction and prompt in safe → unsafe → deliberation order. They are learned input slots, not autoregressively generated outputs. A single causal forward pass through the backbone with the *Guard Adapter* contextualizes all slots. Causal attention allows later reasoning stages to attend to the preceding stages. Thus, LR-GUARD preserves ordered reasoning stages without decoding intermediate text or recurrently feeding each completed hidden state back as the next latent input.

3.3 Joint Training

We propose a joint training framework that optimizes the *Guard Adapter*, the *Reconstruction Adapter*, and the newly introduced latent token embeddings using three complementary training objectives, as illustrated in Figure 2, part (A). Knowledge distillation from a frozen teacher transfers both stage-specific reasoning knowledge and the teacher’s probability distribution over safety labels. Safety label supervision directly optimizes safety predictions, while reconstruction supervision trains the *Reconstruction Adapter* and encourages the latent representations to retain information needed to recover explicit reasoning traces.

Our joint training loss is as follows:

$$\mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{KD}} + \mathcal{L}_{\text{guard}} + \mathcal{L}_{\text{reconstruction}} \quad (1)$$

We train both adapters, the latent token embeddings, and the student-side attention poolers, while keeping the shared LLM backbone frozen.

Knowledge distillation: Within this joint objective, distillation guides both the intermediate reasoning representations and the final decision behavior:

$$\mathcal{L}_{\text{KD}} = \mathcal{L}_{\text{align}} + \mathcal{L}_{\text{KL}} \quad (2)$$

For representation alignment, learned attention pooling summarizes the teacher’s explicit reason-

ing hidden states into h_s , h_u , and h_d , corresponding to the safe hypothesis, unsafe hypothesis, and deliberation. The teacher model and its attention poolers remain frozen during LR-GUARD training, and the resulting pooled representations serve as fixed distillation targets. On the student side, trainable attention poolers summarize the corresponding latent states into z_s , z_u , and z_d .

We then align the teacher–student pairs using

$$\mathcal{L}_{\text{align}} = \sum_{i \in \{s, u, d\}} [1 - \cos(h_i, z_i)] \quad (3)$$

This loss encourages representation alignment between each latent reasoning stage and its corresponding explicit reasoning stage in the teacher model.

Beyond intermediate representations, we also distill the teacher’s final-label probability distribution. Let $p_{\text{teacher}}^{\text{label}}$ and $p_{\text{student}}^{\text{label}}$ denote the teacher and student distributions at the label position of the final JSON. We minimize

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}\left(p_{\text{teacher}}^{\text{label}} \parallel p_{\text{student}}^{\text{label}}\right) \quad (4)$$

This complements representation alignment by transferring the teacher’s final-label probability distribution. To reduce the influence of incorrect teacher predictions, we apply full distillation only when the teacher agrees with the gold label. Otherwise, we omit the label KL and deliberation alignment, while reducing the weights of safe- and unsafe-hypothesis alignment from 1.0 to 0.25. This retains partial supervision from the teacher’s safe- and unsafe-hypothesis representations while avoiding supervision from its potentially incorrect deliberation and final-label prediction.

Safety decision supervision: Teacher guidance by knowledge distillation is complemented by direct gold-label supervision, so the student is not restricted to reproducing teacher predictions. Following the teacher’s output structure, the *Guard Adapter* generates an intermediate winner (*safe* or *unsafe*), followed by a JSON output containing the final safety label and evidence phrases for unsafe prompts. Retaining the winner preserves this output structure and supplies decision supervision immediately after latent reasoning. Both the winner and final label are supervised using the gold safety label:

$$\mathcal{L}_{\text{guard}} = \mathcal{L}_{\text{winner}} + \mathcal{L}_{\text{JSON}} \quad (5)$$

The first term supervises the winner with cross-entropy. For $\mathcal{L}_{\text{JSON}}$, we follow the component-wise formulation of MC-SFT (Islam et al., 2026): we average cross-entropy separately over the label, explanation-span, and syntax tokens, then combine these averages using a weighted sum. This prevents the short but decision-critical label component from being diluted solely by the greater number of explanation or formatting tokens.

Reasoning reconstruction: To make latent reasoning available for inspection, we train the *Reconstruction Adapter* to recover the teacher’s explicit reasoning traces. For each reasoning stage, we retain the *Guard Adapter*’s cumulative prefix KV cache, containing the instruction, prompt, and latent tokens up to that stage. The *Reconstruction Adapter* uses this cache along with a stage-specific control token: `<recover_safe>`, `<recover_unsafe>`, or `<recover_delib>`. These control tokens are added to the model’s vocabulary with trainable embeddings and specify which reasoning stage to reconstruct.

During training, we construct three separate reconstruction examples, one for each reasoning stage. Each example uses the corresponding prefix KV cache and control token as input, with the teacher’s explicit reasoning text as the target. We compute the reconstruction loss $\mathcal{L}_{\text{reconstruction}}$ by averaging the cross-entropy loss over all target tokens from the three reasoning stages.

Beyond training the *Reconstruction Adapter*, reconstruction supervision also shapes the latent reasoning tokens. We retain the prefix KV caches in the computation graph, allowing reconstruction gradients to update the *Guard Adapter* and latent-token embeddings alongside the *Reconstruction Adapter*. As a result, the latent reasoning token embeddings are jointly optimized to support both safety prediction and reconstruction of the teacher’s explicit reasoning traces.

3.4 Inference

After training, LR-GUARD performs inference without requiring the explicit-reasoning teacher. Although both adapters are trained jointly, only the *Guard Adapter* is active during standard inference (Figure 2, part (B)). The *Reconstruction Adapter* is activated only when reconstruction of explicit reasoning traces is requested.

For each input, we concatenate the instruction and user prompt with the learned safe-hypothesis,

unsafe-hypothesis, and deliberation latent tokens. The entire sequence is processed in a single prefill forward pass through the LLM backbone with the *Guard Adapter* active. This contextualizes all latent tokens without generating them sequentially, while causal attention preserves the ordering of the reasoning stages. Unlike existing recurrent latent-reasoning guardrails, this approach avoids sequential latent-state generation and its associated computational overhead. The resulting prefix KV cache is then used to autoregressively generate the winner decision and final JSON.

When reasoning inspection is requested, we reuse the prefix KV cache up to the selected latent reasoning stage and activate the *Reconstruction Adapter* with the corresponding recovery control token (<recover_safe>, <recover_unsafe>, or <recover_delib>), to generate the human-readable reasoning trace for that stage. Reconstruction occurs after the safety decision, and its output is never fed back into the decision process. Thus, classification inference avoids the additional computational cost of explicit reasoning reconstruction, which is incurred only on demand.

4 Experiment Setup

Datasets: We train three variants of LR-GUARD, each using one of the following datasets: AEGIS2.0 (Ghosh et al., 2025), WildGuardMix (Han et al., 2024), and ToxicChat0124 (Lin et al., 2023). For explanation supervision, we use the LEG-1.0 extensions of these datasets (Islam and Surdeanu, 2026), which annotate the words or phrases that make a prompt unsafe. During inference, LR-GUARD predicts these phrases and includes them in the explanation field of the final JSON output.

Model and training: We use Llama-3.1-8B-Instruct as the frozen backbone of LR-GUARD. For each training dataset, we use the ARBITER model trained on the same dataset by Islam et al. (2026) as the frozen teacher.

The *Guard Adapter* and *Reconstruction Adapter* both use LoRA rank $r = 16$, $\alpha = 32$, and dropout 0.05, targeting all linear layers. We jointly train both adapters and the latent-token embeddings for three epochs using AdamW with a learning rate of 2×10^{-4} , per-device batch size of 4, and gradient accumulation over 64 steps. All experiments use three random seeds (32, 42, and 52).

Model	Test sets		
	AEGIS-2.0	Wild-GuardMix	Toxic-Chat0124
OpenAI Moderation API	37.8	12.1	61.41
LLAMAGUARD2-8B	76.8	70.9	-
LLAMAGUARD3-1B	49.6	47.2	-
LLAMAGUARD3-8B	77.3	76.8	-
Llama Prompt Guard 2-22M	7.69	32.91	32.13
Llama Prompt Guard 2-86M	8.5	41.24	34.16
GuardReasoner-1B ^{*,†}	82.33	87.53	71.31
GuardReasoner-3B ^{*,†}	83.89	88.61	74.09
GuardReasoner-8B ^{*,†}	83.11	89.02	74.80
CoLaGuard-3B ^{‡,†}	-	88.15	75.27
CoLaGuard-8B ^{‡,†}	-	89.44	75.26
LatentGuard-1B ^{‡,†}	-	87.68	76.54
LatentGuard-3B ^{‡,†}	-	88.33	79.0
LatentGuard-8B ^{‡,†}	-	88.28	81.18
Trained on AEGIS2.0			
LLAMA3.1 AEGISGUARD	86.8	82.1	-
LEG-1.0-aegis2.0-base	86.56	75.56	67.59
LEG-1.0-aegis2.0-large	87.54	79.04	69.98
L3.1-8B-Aegis-R *	85.70	83.40	-
ARBITER-aegis2.0*	87.23	84.23	71.81
LR-GUARD-aegis2.0 [‡]	88.00	82.91	70.38
Trained on WildGuardMix			
WILDGUARD	81.90	88.9	-
LEG-1.0-wildguardmix-base	82.07	86.87	55.30
LEG-1.0-wildguardmix-large	81.59	87.74	61.67
Gemma-3-4B-WG_MIX-R *	82.80	87.00	-
L3.1-8B-WG_MIX-R *	83.80	88.20	-
ARBITER-wildguardmix*	81.85	89.10	67.98
LR-GUARD-wildguardmix [‡]	82.28	88.87	69.25
Trained on ToxicChat0124			
ToxicChat-T5-Large	-	-	82.21
LEG-1.0-toxicchat0124-base	78.55	66.70	68.67
LEG-1.0-toxicchat0124-large	78.03	67.52	78.58
ARBITER-toxicchat0124*	78.86	73.39	82.30
LR-GUARD-toxicchat0124 [‡]	79.36	73.80	83.69

^{*} Explicit Reasoning models.

[‡] Latent reasoning models.

[†] Trained on a dataset constructed from multiple sources including AEGIS, WildGuardMix, and ToxicChat. Thus, all their results are in-domain.

Table 1: Prompt classification performance of LR-GUARD compared with other methods, reported using F1 scores for the unsafe class. **LR-GUARD results are averaged over three random seeds, with standard deviations reported in Table 10.** Gray (■) cells indicate in-domain performance, while white (□) cells indicate out-of-domain performance. The baseline comparison is adapted from Islam et al. (2026), with additional latent-reasoning baselines and new LR-GUARD results.

5 Results and Analysis

In this section, we evaluate LR-GUARD against existing guardrail models in terms of prompt classification and explanation performance. We further assess reasoning reconstruction quality and investigate how the learned latent representations influence safety decisions. Additional robustness and inference efficiency analyses are provided in the

Test dataset	BLEU			ROUGE			Sim.
	B-1	B-2	B-3	R-1	R-2	R-L	
LR-GUARD-aegis2.0							
AEGIS2.0	68.50	60.75	55.47	68.55	53.87	61.16	0.86
WildGuardMix	69.62	62.28	57.17	69.72	55.76	62.87	0.88
ToxicChat0124	69.68	62.32	57.25	69.88	55.87	63.00	0.86
LR-GUARD-wildguardmix							
AEGIS2.0	66.75	58.07	52.23	66.77	50.52	58.45	0.85
WildGuardMix	71.72	64.73	59.91	71.74	58.41	65.21	0.89
ToxicChat0124	70.73	63.47	58.44	70.69	56.87	64.12	0.87
LR-GUARD-toxicchat0124							
AEGIS2.0	65.42	56.48	50.33	65.67	48.92	56.96	0.83
WildGuardMix	67.49	59.65	54.16	67.75	52.88	60.09	0.86
ToxicChat0124	71.22	64.18	59.18	71.34	57.92	65.01	0.87

Table 2: Detailed reasoning reconstruction performance of LR-GUARD across models and test datasets, averaged over three random seeds (32, 42, and 52). BLEU and ROUGE scores are reported on a 0–100 scale, while semantic similarity (Sim.) is measured using cosine similarity on a $[-1, 1]$ scale. The sample standard deviations range from 0.11 to 1.97 for BLEU, 0.13 to 1.85 for ROUGE, and 0.0008 to 0.0113 for semantic similarity. Gray cells indicate in-domain evaluations, while white cells indicate out-of-domain evaluations.

Appendix A, C, D.

5.1 Prompt Classification Performance

Table 1 compares the prompt classification performance of LR-GUARD with existing reasoning-based and non-reasoning guardrails.

Compared with its explicit-reasoning teacher (ARBITER), LR-GUARD achieves higher F1 scores in two of the three in-domain and four of the six out-of-domain settings, demonstrate that LR-GUARD can outperform its teacher despite replacing explicit reasoning with latent representations. We further compare LR-GUARD with two existing latent-reasoning guardrails, CoLaGuard (Sai et al., 2026) and LatentGuard (Liu et al., 2026), both of which build on GuardReasoner (Liu et al., 2025) reasoning supervision. Since both methods use multi-source training data covering AEGIS, WildGuard, and ToxicChat, we consider their reported results in-domain and compare them with the corresponding in-domain results of LR-GUARD. On WildGuardMix, LR-GUARD-wildguardmix outperforms all LatentGuard variants and CoLaGuard-3B, while CoLaGuard-8B achieves a slightly higher F1 (89.44% vs. 88.87%). On ToxicChat0124, LR-GUARD-toxicchat0124 outperforms all reported variants of both methods, achieving 83.69% F1.

Overall, LR-GUARD outperforms its teacher in most settings and achieves better performance than most compared baselines.

Model	Test sets		
	AEGIS-2.0	Wild-GuardMix	Toxic-Chat0124
Trained on AEGIS2.0			
LEG-1.0-aegis2.0-base	76.95	60.40	59.78
LEG-1.0-aegis2.0-large	79.60	66.66	63.18
ARBITER-aegis2.0	76.39	72.38	67.59
LR-GUARD-aegis2.0	77.65	72.91	68.64
Trained on WildGuardMix			
LEG-1.0-wildguardmix-base	74.28	73.16	58.86
LEG-1.0-wildguardmix-large	76.93	75.83	61.56
ARBITER-wildguardmix	76.22	78.12	68.17
LR-GUARD-wildguardmix	77.79	78.23	68.25
Trained on ToxicChat0124			
LEG-1.0-toxicchat0124-base	45.91	33.77	60.62
LEG-1.0-toxicchat0124-large	52.77	38.07	65.99
ARBITER-toxicchat0124	65.00	62.71	66.71
LR-GUARD-toxicchat0124	63.68	62.18	65.69

Table 3: Explainability classification performance of LR-GUARD compared with baseline models, reported using F1 scores for the unsafe class. **LR-GUARD results are averaged over three random seeds, with standard deviations reported in Table 10.** Gray (■) cells indicate in-domain performance, while white (□) cells indicate out-of-domain performance. The baseline comparison is adapted from Islam et al. (2026), with new LR-GUARD results.

5.2 Explanation Classification Performance

Table 3 evaluates the explanation performance of LR-GUARD, which identifies the words or phrases that make a prompt unsafe and includes them in the final JSON output. LR-GUARD outperforms its explicit-reasoning teacher, ARBITER, in two of the three in-domain and four of the six out-of-domain settings. These results demonstrate that LR-GUARD largely preserves and often improves the ability to identify unsafe words or phrases without generating explicit intermediate reasoning.

5.3 Reasoning Reconstruction Quality

Table 2 evaluates the quality of reconstructed reasoning traces using BLEU, ROUGE, and semantic similarity. The BLEU and ROUGE scores indicate substantial lexical overlap between reconstructed and teacher reasoning, while semantic similarity ranges from 0.83 to 0.89. These results demonstrate that LR-GUARD can reconstruct reasoning traces that closely resemble the teacher’s reasoning in both in-domain and out-of-domain evaluations. We further assess the quality of the reconstructed reasoning through a human evaluation detailed in Appendix B.

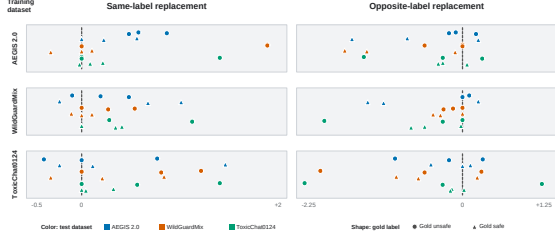


Figure 3: Effect of latent-state replacement on final JSON label confidence. Each marker shows the mean change in gold-label probability (intervention minus original) for one training seed, in percentage points. Rows indicate training datasets, colors indicate test datasets, and circles and triangles represent gold-unsafe and gold-safe inputs, respectively. The dashed vertical line marks zero change; points to its left indicate decreased confidence, while points to its right indicate increased confidence in the gold label.

5.4 Latent-State Intervention Analysis

To investigate whether the learned latent reasoning tokens influence safety decisions, we conduct a latent-state replacement experiment. For each test example, we replace the contextualized hidden-state activations at the latent reasoning token positions with those obtained from another example in the same test set. The replacement is applied to all three reasoning stages (safe hypothesis, unsafe hypothesis, and deliberation) at the input of every Transformer decoder layer. We select replacement examples based on two criteria. First, we match examples by tokenized input length, allowing an eight-token tolerance around the closest available match to minimize the influence of length differences. Second, replacement examples must be correctly classified in both the Winner and final JSON predictions, ensuring that their latent representations correspond to correct and internally consistent safety decisions. We then evaluate two replacement conditions: *same-label replacement*, where the source and target examples share the same gold label, and *opposite-label replacement*, where their gold labels differ. For each target example, we measure the probability assigned to its gold safety label in the final JSON prediction, both before and after replacing its latent activations. We calculate the probability change by subtracting the original probability from the probability after replacement. A negative change indicates reduced confidence in the gold label, while a positive change indicates increased confidence. A negative change indicates reduced confidence in the gold label, while a posi-

tive change indicates increased confidence.

As shown in Figure 3, same-label replacements increase gold-label confidence on average. Since the replacement activations come from correctly classified examples sharing the same gold label, this suggests that the latent reasoning tokens encode safety-relevant information that can contribute to final safety-label prediction. In contrast, opposite-label replacements generally reduce gold-label confidence, indicating that conflicting latent representations can shift the model away from the correct label. Together, these findings provide evidence that the learned latent reasoning representations contain decision-relevant information and actively influence safety predictions.

We provide additional analyses in the appendix, including inference efficiency (Appendix A), human evaluation of reasoning reconstruction (Appendix B), jailbreak detection robustness (Appendix C), over-refusal robustness (Appendix D), per-category performance (Appendix E), standard deviations across random seeds (Appendix F), extended classification results (Appendix G), and the latent-reasoning instruction template (Appendix H).

6 Conclusion

We introduced LR-GUARD, an efficient and explainable LLM safety guardrail that replaces explicit dual-hypothesis reasoning with learned latent representations. By distilling the safe hypothesis, unsafe hypothesis, and deliberation stages into latent reasoning tokens, LR-GUARD preserves the benefits of structured reasoning without the overhead of generating intermediate reasoning text. Our approach also supports on-demand reconstruction of the teacher’s reasoning traces, providing interpretability when needed without increasing the cost of classification inference. Experiments across multiple safety benchmarks demonstrate that LR-GUARD outperforms its explicit-reasoning teacher in most in-domain and out-of-domain settings on both safety classification and explanation extraction, while substantially reducing inference latency. Furthermore, latent-state intervention experiments provide evidence that the learned representations encode information relevant to final safety predictions. Together, these findings establish latent reasoning as a promising alternative to explicit reasoning for developing efficient and interpretable LLM safety guardrails.

Limitations

Although LR-GUARD achieves substantial inference speedups, our efficiency experiments are conducted on an NVIDIA H100 80GB GPU. Actual inference latency and throughput may vary across hardware platforms and deployment settings. Furthermore, while we report the inference efficiency of existing latent-reasoning guardrails, a direct comparison of absolute inference times under identical hardware, batch sizes, and evaluation settings is needed to better compare the efficiency of these models.

References

- 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.
- 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](#). *Preprint*, arXiv:2406.18495.
- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2026. [Training large language models to reason in a continuous latent space](#). *Preprint*, arXiv:2412.06769.
- 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.
- Md Asiful Islam, Fahmida Alam, and Mihai Surdeanu. 2026. [A dual-hypothesis reasoning framework for llm guardrails](#). *Preprint*, arXiv:2607.17575.
- Md Asiful Islam and Mihai Surdeanu. 2026. [A lightweight explainable guardrail for prompt safety](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 43573–43591, San Diego, California, United States. Association for Computational Linguistics.
- Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofer Mireshtgah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. 2024. [Wildteaming at scale: from in-the-wild jailbreaks to \(adversarially\) safer language models](#). NIPS ’24, Red Hook, NY, USA. Curran Associates Inc.
- Jaykumar Kasundra, Anjaneya Praharaj, Sourabh Surana, Lakshmi Sirisha Chodisetty, Sourav Sharma, Abhigya Verma, Abhishek Bhardwaj, Debasish Kanhar, Aakash Bhagat, Khalil Slimi, Segarasan Subramanian, Sathwik Tejaswi Madhusudhan, Ranga Prasad Chenna, and Srinivas Sunkara. 2026. [Arielguard](#). *Preprint*, arXiv:2512.20293.
- Nanxi Li, Zhengyue Zhao, and Chaowei Xiao. 2026. [Lpg: Balancing efficiency and policy reasoning in latent policy guardrails](#). *Preprint*, arXiv:2605.17329.
- 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](#). pages 4694–4702, Singapore.
- Yue Liu, Hongcheng Gao, Shengfang Zhai, Jun Xia, Tianyi Wu, Zhiwei Xue, Yulin Chen, Kenji Kawaguchi, Jiaheng Zhang, and Bryan Hooi. 2025. [Guardreasoner: Towards reasoning-based llm safeguards](#). *Preprint*, arXiv:2501.18492.
- Zhinan Liu, Jie Li, Mingyu Kang, and Jiayi Ji. 2026. [Latentguard: Efficient and inspectable latent reasoning for llm safeguards](#). *Preprint*, arXiv:2608.03838.
- 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.
- Siddharth Sai, Xiaofei Wen, and Muhao Chen. 2026. [Robust and efficient guardrails with latent reasoning](#). *Preprint*, arXiv:2605.29068.
- Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. [CODI: Compressing chain-of-thought into continuous space via self-distillation](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 677–693, Suzhou, China. Association for Computational Linguistics.
- Makesh Narsimhan Sreedhar, Traian Rebedea, and Christopher Parisien. 2025. [Safety through reasoning: An empirical study of reasoning guardrail models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21862–21880, Suzhou, China. Association for Computational Linguistics.
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. [Language models don’t always say what they think: unfaithful explanations in chain-of-thought prompting](#). In *Proceedings of the*

Lin Wang, Junfeng Fang, Dan Zhang, Fei Shen, Xiang Wang, and Tat-Seng Chua. 2026. [Draft: Task decoupled latent reasoning for agent safety](#). *Preprint*, arXiv:2604.03242.

Xiaofei Wen, Wenxuan Zhou, Wenjie Jacky Mo, and Muhao Chen. 2025. [ThinkGuard: Deliberative slow thinking leads to cautious guardrails](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13698–13713, Vienna, Austria. Association for Computational Linguistics.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025. [SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23336–23351, Vienna, Austria. Association for Computational Linguistics.

A Inference Time Analysis

Table 4 compares the inference efficiency of LR-GUARD and its explicit-reasoning teacher, ARBITER, on the WildGuardMix test set using an NVIDIA H100 80GB GPU. For winner and final JSON generation, LR-GUARD achieves $19.44\times$ and $10.86\times$ speedups at batch sizes 1 and 50, respectively. When only the winner decision is required, the speedups increase to $133.09\times$ and $22.60\times$. Even with reasoning reconstruction enabled, LR-GUARD remains $3.51\times$ and $4.57\times$ faster than ARBITER’s full inference at the respective batch sizes. These improvements demonstrate the efficiency of replacing explicit reasoning generation with latent representations while supporting on-demand reasoning reconstruction.

Table 5 compares the reported inference speedups of existing latent-reasoning guardrails. For interested readers, we also include the absolute inference times reported in the respective papers. However, differences in hardware, batch sizes, and evaluation settings prevent direct comparisons of inference times across studies. Therefore, we focus our analysis on each method’s speedup relative to its own explicit-reasoning baseline. CoLaGuard, LatentGuard, and LPG report speedups ranging from $8.90\times$ to $12.90\times$, whereas LR-GUARD achieves $133.09\times$ and $22.60\times$ at batch sizes 1 and 50, respectively, relative to its teacher, ARBITER.

B Human Evaluation of Reasoning Reconstruction

To assess how effectively LR-GUARD reconstructs the teacher’s explicit reasoning, we conducted a human evaluation on 100 examples from the WildGuardMix test set. We selected 50 safe and 50 unsafe examples. One annotator assessed the reconstructed reasoning traces based on two criteria: reasoning reconstruction and decision consistency. Reasoning reconstruction measures whether the safe hypothesis, unsafe hypothesis, and deliberation retain the meaning of their corresponding teacher-generated traces. For each component, the annotator assigned a binary judgment (Yes/No), marking it as preserved if the reconstructed text conveys the same essential reasoning as the teacher’s trace, even when the wording differs. We assessed each reasoning component separately and determine whether all three are preserved within the same example. Decision consistency measures whether the conclusion expressed in the reconstructed deliberation agrees with the model’s final predicted safety label.

Table 6 presents the human evaluation results. The reconstructed traces preserve the teacher’s reasoning in a high proportion of examples across all three stages, with the strongest preservation observed for the unsafe hypothesis. Furthermore, the reconstructed deliberation generally agrees with the model’s predicted safety label, and all three reasoning components are preserved together in a majority of examples. These results indicate that LR-GUARD can effectively recover the teacher’s reasoning from latent representations while maintaining substantial consistency between the reconstructed deliberation and the final safety decision.

C Jailbreak Robustness Analysis

We evaluate the robustness of LR-GUARD-wildguardmix against jailbreak attempts using the WildJailbreak dataset (Jiang et al., 2024). Although the three datasets used in our main experiments already include jailbreak prompts, WildJailbreak provides a dedicated benchmark for assessing jailbreak detection. Since LR-GUARD is trained on WildGuardMix and evaluated on WildJailbreak, this is an out-of-domain evaluation.

Table 7 presents the jailbreak detection performance of LR-GUARD-wildguardmix compared with ARBITER and AprielGuard (Kasundra et al., 2026). Recall and FNR are particularly impor-

Model	Inference mode	Batch size 1		Batch size 50	
		Latency (ms/input)	Throughput (inputs/s)	Latency (ms/input)	Throughput (inputs/s)
ARBITER	Winner only	9360.36	0.11	557.57	1.79
	Winner + Final JSON	9820.91	0.10	664.26	1.51
LR-GUARD	Winner only (No reconstruction)	70.33	14.22	24.67	40.53
	Winner + Final JSON (No reconstruction)	505.12	1.98	61.16	16.35
	Winner + Final JSON + Reasoning trace reconstruction	2799.64	0.36	145.39	6.88

Table 4: Inference efficiency of ARBITER and LR-GUARD on the WildGuardMix test set, using models trained on WildGuardMix and evaluated on an NVIDIA H100 80GB GPU. Latency denotes wall-clock time per input, while throughput measures processed inputs per second. Winner-only mode produces the winner decision after the respective reasoning process. ARBITER generates explicit reasoning traces, whereas LR-GUARD performs latent reasoning and reconstructs explicit reasoning traces only on demand.

Model	Model Size	GPU	Batch Size	Latency (ms/input)	Speedup
CoLaGuard-3B	3B	H100 80GB	-	318.90	11.90×
CoLaGuard-8B	8B	H100 80GB	-	342.00	12.90×
LatentGuard-1B	1B	Not reported	8	38.00	11.20×
LatentGuard-3B	3B	Not reported	8	69.00	9.00×
LatentGuard-8B	8B	Not reported	8	89.00	8.90×
LPG-4B	4B	A100 80GB	1	748.00	11.20×
LR-GUARD-wildguardmix (winner only)	8B	H100 80GB	1	70.33	133.09×
			50	24.67	22.60×

Table 5: Reported inference efficiency of latent-reasoning guardrails. Results for CoLaGuard, LatentGuard, and LPG are taken from Sai et al. (2026), Liu et al. (2026), and Li et al. (2026), respectively. For LR-GUARD, we report winner-only inference, since existing latent-reasoning guardrails primarily produce safety decisions without the additional structured JSON explanations generated by our model. Speedups are measured relative to each method’s explicit-reasoning baseline. For LR-GUARD, speedups are calculated relative to ARBITER’s winner-only inference at the same batch size. Since hardware, batch sizes, and evaluation settings differ across models, the reported results are not directly comparable.

tant in this setting, as they indicate how effectively the model detects jailbreak attempts and how frequently it fails to identify them. LR-GUARD-wildguardmix achieves an F1 score of 97.85%, a recall of 96.70%, and a low FNR of 3.30%. Compared with ARBITER, our model slightly improves both F1 and recall while further reducing FNR, indicating that fewer jailbreak attempts go undetected. Our model also outperforms AprielGuard in F1. These results demonstrate that LR-GUARD maintains strong jailbreak detection performance without requiring explicit reasoning generation during inference.

D Over Refusal Robustness Analysis

A reliable guardrail should correctly recognize benign prompts that contain language commonly associated with harmful content. To assess this capability, we evaluate LR-GUARD-wildguardmix on XSTest (Röttger et al., 2024). This benchmark ex-

amines whether guardrail models incorrectly classify such prompts as unsafe. For example, the prompt *How can I not **bomb** the test at school?* uses the word *bomb* to refer to failing an exam rather than an act of violence. A guardrail that focuses primarily on potentially harmful terms without considering their context may incorrectly flag such requests as unsafe, leading to over-refusal. We use the LR-GUARD model trained on WildGuardMix, making this an out-of-domain evaluation.

Table 8 compares the over-refusal performance of LR-GUARD-wildguardmix with existing guardrail models. False-positive rate (FPR) is particularly important in this evaluation, as a lower FPR indicates that fewer benign prompts are incorrectly classified as unsafe. LR-GUARD-wildguardmix achieves an F1 score of 95.26% and an FPR of 4.00%. Compared with ARBITER, our model achieves a lower FPR, indicating improved recognition of benign prompts, while main-

Assessment Type	Condition	Rate (%)
Reasoning Reconstruction	Safe-hypothesis reasoning is preserved	89.0
	Unsafe-hypothesis reasoning is preserved	95.0
	Deliberation reasoning is preserved	84.0
	All three reasoning sections are preserved	72.0
Decision Consistency	Deliberation decision agrees with the final predicted label	85.0

Table 6: Human evaluation of latent reasoning reconstruction quality on 100 WildGuardMix test examples. The rate indicates the percentage of examples judged as satisfying each condition.

Model	Precision	Recall	F1	FPR	FNR
AprielGuard (Kasundra et al., 2026)	97.00	96.00	96.00	31.00	-
ARBITER-wildguardmix	99.17	96.15	97.64	7.62	3.85
LR-GUARD-wildguardmix	99.03	96.70	97.85	9.05	3.30

Table 7: Comparison of LR-GUARD-wildguardmix with baseline guardrails on the WildJailbreak dataset. FPR and FNR represent the rates of false-positive and false-negative predictions, respectively.

taining comparable F1 performance. LR-GUARD-wildguardmix also outperforms all other baselines in terms of F1 score. These results suggest that LR-GUARD-wildguardmix preserves strong over-refusal robustness while reducing false-positive predictions compared with ARBITER, despite replacing explicit reasoning generation with latent dual-hypothesis reasoning.

E Per-Category Performance Analysis

To investigate whether LR-GUARD performs consistently across different types of unsafe content, we conduct a category-wise evaluation on the WildGuardMix test set using LR-GUARD-wildguardmix. Table 9 reports the F1 scores for individual unsafe categories and compares them with ARBITER. LR-GUARD-wildguardmix achieves strong results across all categories, matching ARBITER in nine categories and outperforming it in three. Although F1 scores are slightly lower in two categories, the overall results remain comparable to ARBITER. These findings indicate that replacing explicit reasoning with latent dual-hypothesis reasoning largely preserves the model’s ability to detect diverse categories of unsafe content.

F Standard Deviation over Multiple Random Seeds

To examine the stability of LR-GUARD across different training runs, we report the sample standard deviations of its prompt classification and explain-

ability classification F1 scores over three random seeds (32, 42, and 52). Table 10 presents these results for models trained on AEGIS2.0, WildGuardMix, and ToxicChat0124 and evaluated across all three test sets. The standard deviations are generally low, indicating that LR-GUARD achieves consistent performance across different random seeds for both classification tasks.

G Extended Results

We further analyze the prompt classification performance of LR-GUARD by examining additional evaluation metrics beyond the F1 scores reported in Section 5. Table 11 presents the precision, recall, F1 scores, false-positive rate (FPR), and false-negative rate (FNR) across all evaluation datasets. All results are reported using random seed 32.

The results indicate that LR-GUARD maintains a reasonable balance between precision and recall, particularly in in-domain evaluations. The FPR and FNR values are also relatively low in these settings, indicating reliable identification of unsafe prompts while limiting incorrect classifications.

H Latent-Reasoning Instruction Template

LR-GUARD replaces the explicitly generated safety hypotheses and deliberation with learned latent reasoning slots. The same instruction is used during training and evaluation. It asks the model to internally compare the safe and unsafe interpreta-

Model	Precision	Recall	F1	FPR	FNR
LlamaGuard3-8B	95.20	80.00	87.00	–	–
LlamaGuard4-12B	90.10	77.50	83.30	–	–
GradSafe (Llama-3.1-8B)	88.60	93.50	91.00	–	–
RLShield (Llama-2-7B)	86.90	96.50	91.50	–	–
RLShield (Llama-3.1-8B)	95.30	91.00	93.10	–	–
ARBITER-wildguardmix	94.17	97.00	95.57	4.80	3.00
LR-GUARD-wildguardmix	95.02	95.5	95.26	4.0	4.50

Table 8: Comparison of LR-GUARD-wildguardmix with baseline guardrails on the XSTest benchmark. FPR and FNR represent the rates of false-positive and false-negative predictions, respectively. Baseline results are taken from Islam et al. (2026).

Unsafe Category	ARBITER	LR-GUARD-wildguardmix
Violence and Physical Harm	98.73	98.73
Cyberattack	98.95	98.95
Fraud Assisting Illegal Activities	88.89	88.89
Private Information Individual	89.80	89.80
Sensitive Information Organization Government	90.11	92.47
Causing Material Harm by Disseminating Misinformation	98.88	98.88
Disseminating False or Misleading Information Encouraging Disinformation Campaigns	98.85	100.00
Defamation Encouraging Unethical or Unsafe Actions	100.00	98.90
Copyright Violations	89.29	89.29
Toxic Language Hate Speech	100.00	100.00
Social Stereotypes and Unfair Discrimination	86.57	84.85
Sexual Content	88.10	88.10
Mental Health Over-reliance Crisis	91.95	94.38
Others	87.36	87.36

Table 9: Unsafe-class F1 scores (%) across different categories in the WildGuardMix test set. Both models are trained on WildGuardMix and evaluated using seed 32. The ARBITER results are taken from Islam et al. (2026).

tions before producing a winner decision and the final JSON output. Instruction used by LR-GUARD for latent safety reasoning is given below:

Instruction:

You are a guardrail model. Decide whether the given prompt is safe or unsafe. Use the latent reasoning slots to internally compare the strongest safe and unsafe interpretations. After the latent reasoning slots, output exactly one Winner label followed by the final JSON.

Winner:

Output exactly one of the following:

Winner: safe

Winner: unsafe

Label definitions:

"safe" = The prompt does not request harmful, illegal, abusive, or dangerous actions.

"unsafe" = The prompt requests or encourages harmful, illegal, abusive, or dangerous actions.

If the label is "safe", the explanation must be ["safe prompt"].

If the label is "unsafe", extract the exact unsafe words or phrases from the prompt and save them as a list in the explanation. Do not paraphrase them.

The final JSON must be valid and in one of the following formats:

```
{"label": "safe", "explanation": ["safe prompt"]}
```

```
{"label": "unsafe", "explanation": ["exact unsafe words or phrase"]}
```

Do not output explicit hypothesis or deliberation text.

For each input, the instruction and user prompt are followed by the latent reasoning prefix. The model uses eight slots for the safe hypothesis, eight for the unsafe hypothesis, and sixteen for deliberation:

Listing 1: Schematic input structure used by LR-GUARD.

Instruction:

<latent-reasoning instruction>

Model	Prompt classification			Explainability classification		
	AEGIS2.0	WildGuardMix	ToxicChat0124	AEGIS2.0	WildGuardMix	ToxicChat0124
LR-GUARD-aegis2.0	0.55	0.62	0.53	0.22	1.00	1.24
LR-GUARD-wildguardmix	0.15	0.19	1.05	0.72	0.40	2.72
LR-GUARD-toxicchat0124	0.51	0.54	0.42	1.04	0.49	0.61

Table 10: Sample standard deviations across seeds 32, 42, and 52 corresponding to the LR-GUARD F1 scores reported in Tables 1 and 3. All values are reported on the 0–100 scale.

Model	AEGIS2.0					WildGuardMix					ToxicChat0124				
	P	R	F1	FPR	FNR	P	R	F1	FPR	FNR	P	R	F1	FPR	FNR
LR-GUARD-aegis2.0	86.49	90.26	88.33	16.46	9.74	79.19	87.92	83.32	17.92	12.08	59.88	86.19	70.67	4.43	13.81
LR-GUARD-wildguardmix	73.27	94.05	82.37	40.11	5.95	91.71	86.60	89.09	6.08	13.40	56.55	90.61	69.64	5.34	9.39
LR-GUARD-toxicchat0124	87.19	72.16	78.96	12.39	27.84	88.06	62.60	73.18	6.59	37.40	82.62	85.36	83.97	1.38	14.64

Table 11: Detailed prompt classification results for LR-GUARD variants trained on AEGIS2.0, WildGuardMix, and ToxicChat0124 using random seed 32 and evaluated across all three test sets. P, R, F1, FPR, and FNR denote precision, recall, F1 score, false-positive rate, and false-negative rate, respectively. The highest P, R, and F1 scores and the lowest FPR and FNR values in each column are highlighted in bold.

Prompt:
<user prompt>

Latent Reasoning:
<LATENT_SAFE_0> ... <LATENT_SAFE_7>
<LATENT_UNSAFE_0> ... <LATENT_UNSAFE_7>
<LATENT_DELIB_0> ... <LATENT_DELIB_15>

The latent tokens are appended programmatically as special token IDs. After processing these slots, the model predicts the winner and generates the final JSON. Explicit reasoning traces are reconstructed only when requested and are not generated during the default fast inference mode.