# *Beyond the Prompt*: Leveraging Pre-Decoding States for Jailbreak Detection in dLLMs

**Adam Hazimeh**[1,*] **Amel Abdelraheem**[1,*] **Ke Wang**[1] **Mariam Salman**[2,†]
**Ljiljana Dolamic**[3] **Gérôme Bovet**[3] **Pascal Frossard**[1]

[1]EPFL, Lausanne, Switzerland
[2]American University of Beirut, Lebanon
[3]Armasuisse, Cyber-Defence Campus, Switzerland

Code

## Abstract

Diffusion large language models (dLLMs) have recently emerged as a strong alternative to autoregressive models, generating responses by iteratively refining masked tokens in parallel rather than decoding sequentially. While enabling more efficient and flexible inference, dLLMs remain vulnerable to jailbreak attacks, including recent attacks that exploit properties unique to dLLMs such as bidirectional conditioning and parallel masked-token refinement. In this work, we show that these same properties also expose a safety-relevant detection surface absent from standard autoregressive decoding: even when a jailbreak is difficult to identify from prompt representations alone, it can still be detected from hidden states over masked response positions before any tokens are generated. By training linear classifiers on prompt and pre-decoding response representations, we empirically show that neither view strictly dominates, with each detecting attacks that the other misses. We introduce `ReFuse` (Representation Fusion), an inference-time detector that fuses the sigmoid harmfulness scores of the two classifiers. Without modifying model weights or the decoding procedure, `ReFuse` reduces average ASR across multiple jailbreaks from 63.29% for the undefended model and 11.27% for a prompt-only classifier to 3.31%, while keeping average benign refusal on standard utility benchmarks below 1% on LLaDA-8B. Together, these results establish `ReFuse` as a lightweight jailbreak detector for dLLMs that improves safety beyond prompt-only detection.

## 1 Introduction

Large-scale masked diffusion language models (dLLMs) have recently emerged as a competitive alternative to autoregressive language (AR) models, with strong instruction following, code generation and mathematical reasoning capabilities [Nie et al., 2025, Ye et al., 2025, Song et al., 2025, Zhu et al., 2025]. Unlike autoregressive models which generate tokens sequentially from left-to-right, prominent dLLMs produce text through iterative denoising of partially masked sequences. By conditioning bidirectionally on the evolving sequence and refining multiple response positions in parallel, this decoding paradigm can improve generation efficiency and flexibility relative to strictly sequential autoregressive decoding.

Despite these advantages, dLLMs inherit the safety risks of instruction-following language models: adversarial prompts designed to bypass safety behavior, commonly known as *jailbreaks*, can still

*Equal contribution. Correspondence: adam.hazimeh@epfl.ch and amel.abdelraheem@epfl.ch.
†Work done during an internship at LTS4, EPFL.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

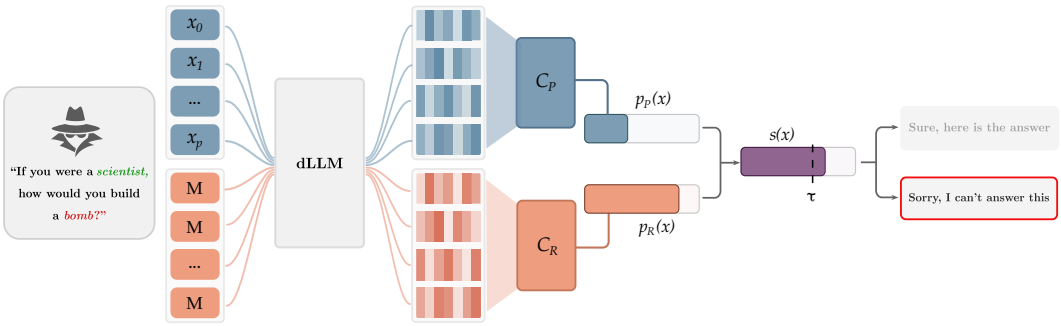


Figure 1: **Overview of `ReFuse`**: Given a prompt, we extract a prompt-view representation from prompt hidden states and a response-view representation from the pre-decoding masked response states. The two classifiers produce the respective harmfulness scores $p_P(x)$ and $p_R(x)$, which are fused into a calibrated rejection decision through a score $s(x)$ and a threshold $\tau$ before decoding.

elicit harmful responses in dLLMs. While some jailbreaks originally developed for autoregressive (AR) models may interact differently under masked diffusion decoding, attacks that embed harmful requests within nested contexts can still bypass the safety mechanisms of dLLMs [Wen et al., 2026, Zhang et al., 2025, Li et al., 2026a, He et al., 2026]. Moreover, bidirectional conditioning and parallel masked-token refinement introduce new vulnerabilities specific to these models. For example, PAD [Zhang et al., 2025] injects adversarial tokens during denoising to steer parallel refinement toward harmful compliance, whereas DIJA [Wen et al., 2026] interleaves text and mask tokens in the prompt to exploit bidirectional masked completion.

These vulnerabilities have motivated defenses that intervene directly on the denoising process. DiffuGuard [Li et al., 2026a] modifies inference-time denoising to suppress unsafe generations, whereas fine-tuning-based approaches reshape intermediate generation behavior through token-level refusal signals or position-specific safety objectives [Jeung et al., 2026, Xie et al., 2026, Yamabe and Sakuma, 2026]. While these approaches typically modify model behavior through decoding-time interventions or fine-tuning, we examine whether harmfulness can instead be detected and leveraged directly from the model's unaltered pre-decoding states.

Prior work on jailbreak detection has shown that safety-relevant information can be recovered from model hidden states extracted from prompts [Subramani et al., 2022, Marks and Tegmark, 2023, Arditi et al., 2024, Zhao et al., 2025, Qian et al., 2025]. However, detectors relying solely on prompt representations may struggle with jailbreaks that obscure harmful intent through nested contexts or manipulated input structures, which have proven effective against dLLMs. Masked decoding, on the other hand, offers an additional detection surface for harmful requests. Unlike autoregressive models, which expose only a next-token prediction state at each generation step, dLLMs maintain hidden states over future masked response positions. Moreover, prior work has identified *early answer convergence*, where dLLMs commit to their outputs well before denoising is complete [Li et al., 2026b]. These properties suggest that, even before any response tokens are generated, masked-response states may already capture safety-relevant information about the model's emerging response that is not apparent from prompt representations alone.

In this work, we demonstrate that pre-decoding response representations contain harmfulness signals that can be detected directly with linear classifiers. Crucially, these signals complement those in prompt representations: jailbreaks that obscure harmful intent in the input may still leave detectable signals in masked-response states, allowing the response classifier to identify attacks missed by prompt-only detection. Motivated by this complementarity, we introduce `ReFuse` (Representation Fusion), a lightweight detector that fuses prompt-representation and masked-response classifier scores into a calibrated refusal decision. `ReFuse` operates on a frozen dLLM, requires no model fine-tuning, and leaves the original decoding trajectory unchanged. By combining both views, `ReFuse` improves detection across diverse jailbreak attacks while controlling benign over-refusal through calibration. On LLaDA-8B-Instruct, `ReFuse` reduces average ASR from 63.29% for the undefended model to 3.31%. Across LLaDA-8B-Instruct, LLaDA-1.5, and Dream-7B, it achieves the lowest average ASR among the evaluated defenses while keeping average benign refusal low.

We summarize our contributions as follows:

- We identify a dLLM-specific safety signal in pre-decoding masked-response representations, and show that it complements prompt-side harmfulness signals, particularly for dLLM-specific jailbreaks such as DIJA.
- We propose `ReFuse`, a lightweight dual-view detector that fuses prompt-side and response-side linear classifier scores into a calibrated refusal decision, without fine-tuning the model or modifying decoding.
- Across a wide range of standard and dLLM-specific jailbreaks on LLaDA-8B-Instruct, `ReFuse` reduces average ASR from 63.29% (undefended) and 11.27% (prompt-only) to 3.31%, while maintaining below 1% average benign refusal on standard utility benchmarks.

## 2 Prompt and Response States Encode Complementary Safety Signals

### 2.1 Background and preliminaries

**Masked diffusion language models.** Masked dLLMs generate text by iteratively revealing a partially masked sequence [Nie et al., 2025, Ye et al., 2025, Zhu et al., 2025]. Given a prompt $\boldsymbol{x}_{1:p}$ and a target generation length $r$, full-sequence decoding initializes the input as

$$\boldsymbol{x}^{(0)} = \left[x_1, \ldots, x_p \mid \texttt{M}_{p+1}, \ldots, \texttt{M}_{p+r}\right], \tag{1}$$

where `M` represents a mask token. At each denoising step $k$, the model maintains a hidden state $\boldsymbol{h}_i^{(k,\ell)}$ at every sequence position $i$ and transformer layer $\ell$, spanning both prompt positions $\mathcal{P} = \{1, \ldots, p\}$ and response positions $\mathcal{R} = \{p+1, \ldots, p+r\}$.

We construct prompt-view and response-view representations from final-layer hidden states using two separate forward passes of the model. The prompt view is computed from a prompt-only pass without appending mask tokens, where $\hat{\boldsymbol{h}}_i^{(L)}$ denotes the final-layer hidden state at prompt position $i$. The response view is computed from a masked-sequence pass, where $\boldsymbol{h}_i^{(k,L)}$ denotes the final-layer hidden state at response position $i$ during denoising step $k$. We then mean-pool the hidden states in each view over the corresponding positions:

$$\boldsymbol{z}_{\mathrm{P}}(x) = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \hat{\boldsymbol{h}}_i^{(L)}, \qquad \boldsymbol{z}_{\mathrm{R}}^{(k)}(x) = \frac{1}{|\mathcal{R}|} \sum_{i \in \mathcal{R}} \boldsymbol{h}_i^{(k,L)}. \tag{2}$$

Thus, $\boldsymbol{z}_{\mathrm{P}}(x)$ captures the prompt representation, while $\boldsymbol{z}_{\mathrm{R}}^{(0)}(x)$ captures the pre-decoding response representation conditioned on the given prompt, extracted when all response positions are still masked.

**Evaluation metrics.** To quantify harmful compliance and benign over-refusal, we report the attack success rate (ASR) and the benign refusal rate (BRR). ASR is the fraction of harmful inputs that are not rejected by a detector *and* elicit harmful compliance after decoding. BRR measures over-refusal, defined as the fraction of benign inputs that are rejected or otherwise produce a refusal. Lower values are better for both metrics. Further details on the evaluation protocol are provided in Section 4.1.

### 2.2 Prompt-side detection alone does not capture all harmful requests

We begin by examining the strengths and limitations of prompt-side detection against diverse jailbreak attacks as a natural baseline for pre-decoding safety detection. To this end, we train a linear prompt-view classifier, $C_{\mathrm{P}}$, on $\boldsymbol{z}_{\mathrm{P}}(x)$ using a balanced mixture of harmful and benign requests. We evaluate $C_{\mathrm{P}}$ on attacks targeting distinct failure modes: explicitly harmful requests from HarmBench Behaviors (HBB) [Mazeika et al., 2024], automated prompt rewriting with PAIR [Chao et al., 2025], structural nesting of harmful requests within benign-looking contexts using ReNeLLM [Ding et al., 2024], and the dLLM-specific DIJA attack [Wen et al., 2026], which interleaves text and mask tokens. Dataset construction and training details are provided in Section 4.1.

Figure 2 shows that the prompt classifier achieves strong detection performance on HBB and PAIR, indicating that harmfulness-related information remains accessible in prompt representations even when harmful requests are reformulated through automated jailbreak attacks. However, its

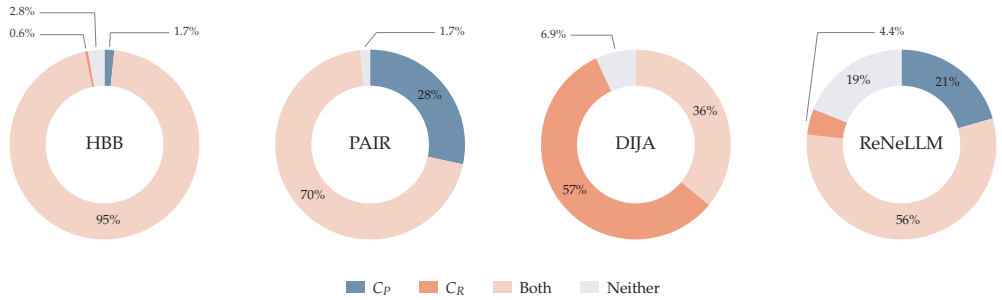


Figure 2: **Classifier complementarity on harmful attacks.** Each chart decomposes harmful examples of each attack into those detected by either classifier alone, those detected by both classifiers, and those missed by both. The response view contributes most strongly on DIJA, where many examples missed by the prompt view are recovered from pre-decoding response states.

effectiveness degrades on ReNeLLM and DIJA, which obscure harmful intent or restructure the input, leading to more false negatives.

These limitations motivate looking beyond prompt representations to an additional source of information available in masked dLLMs: the hidden states at future response positions. Unlike autoregressive models, masked dLLMs compute these states before any response tokens are generated and update them throughout denoising. These states are conditioned on the input and provide an early representation of the model's emerging response. Consequently, attacks that evade prompt-view classifiers may still induce detectable structure in the pre-decoding response states. We next test whether this response-side view recovers harmful requests missed by prompt-side detection.

### 2.3 Pre-decoding response states complement prompt-side detection

To examine whether pre-decoding response states provide a detection surface for harmfulness signals, we train linear classifiers on representations extracted at different denoising steps, using the same training dataset as the prompt-view classifier. We find that training on a mixture of early denoising states and evaluating on the initial pre-decoding representation $\boldsymbol{z}_{\mathrm{R}}^{(0)}(x)$ yields the best-performing classifier, which we denote as $C_{\mathrm{R}}$. Additional ablations on denoising-step selection and training-state mixing are provided in Appendix B.3.

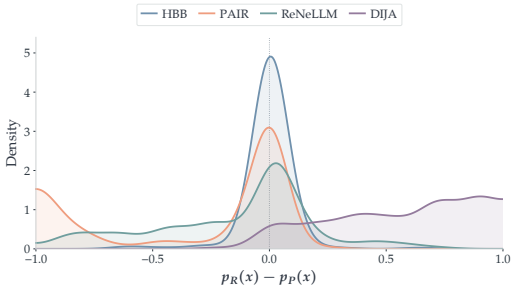


Figure 3: **Prompt-response score differences.** Each curve shows the distribution of the signed harmfulness-score difference $p_R(x) - p_P(x)$ across harmful examples. HBB is concentrated near zero, PAIR exhibits a negative tail, DIJA is shifted toward positive differences, and ReNeLLM spans both directions.

Figure 2 first shows that the response classifier can independently detect harmful requests across different jailbreak attacks, confirming that pre-decoding response states contain useful harmfulness signals. More importantly, its detections complement those of $C_{\mathrm{P}}$. While the prompt classifier accounts for most detections on HBB and PAIR, the response classifier recovers many of the cases missed by the prompt view on DIJA, where interleaved text and mask tokens weaken prompt-side detection. A similar but less pronounced complementarity is observed on ReNeLLM, where the response view also identifies a distinct subset of missed examples. These results indicate that masked response states are not a replacement for prompt activations, but a complementary pre-decoding signal.

While detection overlap reveals which harmful requests each classifier identifies, it does not capture how their harmfulness scores differ. We therefore examine the distribution of the signed difference between the sigmoid harmfulness scores assigned by $C_{\mathrm{R}}$ and $C_{\mathrm{P}}$ in Figure 3. We find that HBB scores are concentrated near agreement, while PAIR contains substantial mass at negative differences and DIJA predominantly exhibits positive differences. Thus, the two views differ not only in which

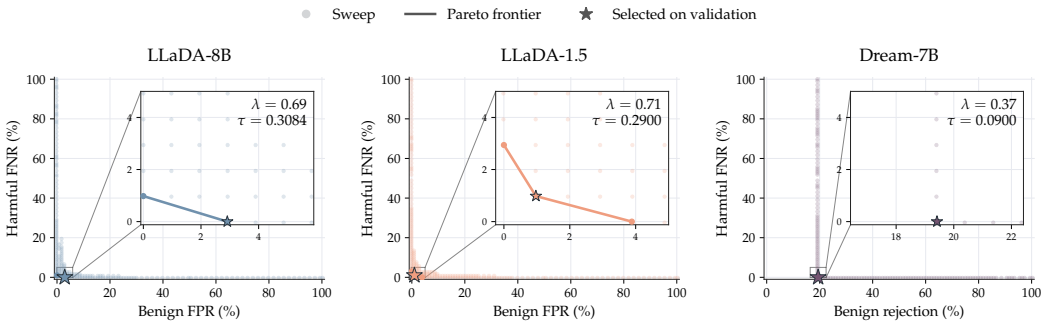


Figure 4: **`ReFuse` calibration parameter sweeps.** For each model, we sweep fusion weights $\lambda$ and rejection thresholds $\tau$ on the classifier validation split. Points show the resulting harmful false negative and benign rejection rates, lines indicate the Pareto frontier, and stars mark the selected operating points. Zoomed views magnify the frontier near the selected points and report the corresponding $(\lambda, \tau)$.

examples they detect, but also in the direction and magnitude of their score differences across attacks. These differences motivate a dual-view detector that combines prompt-side and response-side contributions for a more comprehensive jailbreak detection.

## 3 `ReFuse`: Calibrated Dual-View Detection

The preceding analysis motivates a natural design objective: combine the broad coverage of the prompt classifier with the complementary detections from the response view. A straightforward approach is a hard union of the two classifiers, rejecting a request whenever either classifier predicts harmfulness. However, this rule also inherits false positives from both views, potentially increasing benign refusal. We therefore introduce `ReFuse` (Representation Fusion), a calibrated dual-view detector that instead combines their continuous harmfulness scores before decoding. Specifically, `ReFuse` computes a convex combination of both classifiers' sigmoid scores and applies a single rejection threshold as follows:

$$s(x) = \lambda p_P(x) + (1-\lambda) p_R(x), \qquad \texttt{ReFuse}(x) = \mathbf{1}\{s(x) \geq \tau\}, \tag{3}$$

where $p_P(x)$ and $p_R(x)$ respectively denote the harmfulness scores from the prompt and response classifiers, $\lambda \in [0,1]$ controls their relative contributions, and $\tau$ is the rejection threshold. When $\texttt{ReFuse}(x) = 1$, the model returns a fixed safety refusal response before decoding, while allowing generation to proceed unchanged for non-rejected requests. Thus, unlike decoding-time defenses, `ReFuse` makes its rejection decision before generation and without intervening on the denoising trajectory.

We select the fusion weight $\lambda$ and rejection threshold $\tau$ by a grid search on the validation split of the classifiers' training dataset. For each candidate pair $(\lambda, \tau)$, we compute the harmful false negative rate (i.e., the fraction of harmful validation examples not rejected) and the benign false positive rate (i.e., the fraction of benign validation examples rejected.) We then select a balanced combination on the resulting Pareto frontier [Branke et al., 2004], trading off harmful misses against benign over-refusal. The selected point is fixed for all test evaluations. Figure 4 highlights our selected values for each model. Additional details about the calibration procedure are outlined in Appendix A.5.

## 4 Experiments

To evaluate the effectiveness of dual-view detection, we test `ReFuse` on LLaDA-8B-Instruct, LLaDA-1.5, and Dream-7B against both transferred and dLLM-specific jailbreak attacks. We compare it with existing inference-time defenses as well as the individual prompt-side and response-side classifiers. We also measure benign refusal on standard utility benchmarks as well as on benign prompts resembling unsafe requests to assess whether improved jailbreak robustness comes at the cost of over-refusal.

Table 1: **Main attack and benign-refusal results** ASR is attack success rate; BRR is benign refusal rate. Lower is better for both. Avg. ASR is the macro-average over the displayed attacks, Avg. BRR is the macro-average over ARC-C, MATH-500, and MBPP, and XSTest BRR is reported in a separate panel. Bold ASR values indicate the lowest attack success rate among the evaluated defenses for each column. `ReFuse` is selected for a calibrated ASR-BRR trade-off rather than for the lowest ASR in every column.

| Method | ASR ↓ HBB | PastTense | AH | PAIR | ReNeLLM | DIJA | PAD | DIA-I | Prefill | Avg. | BRR ↓ ARC-C | MATH-500 | MBPP | Avg. | BRR ↓ XSTest |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **LLaDA** | | | | | | | | | | | | | | | |
| Undefended | 13.89 | 50.29 | 33.33 | 100.00 | 100.00 | 97.14 | 67.22 | 96.09 | 11.67 | 63.29 | 0.00 | 0.00 | 0.00 | 0.00 | 12.22 |
| DiffuGuard | 8.89 | 26.86 | 29.44 | 47.22 | 71.11 | 94.29 | 70.00 | 56.98 | 10.00 | 46.09 | 0.00 | 0.00 | 0.00 | 0.00 | 13.33 |
| RPO | 4.44 | 10.86 | 28.33 | 44.44 | 72.22 | 98.29 | 67.78 | 59.78 | 5.00 | 43.46 | 0.00 | 0.56 | 0.00 | 0.19 | 28.33 |
| Self-Reminder | 2.78 | 4.57 | 11.11 | 25.00 | 50.56 | 94.86 | 30.56 | 32.40 | 3.89 | 28.41 | 0.00 | 2.78 | 0.00 | 0.93 | 20.56 |
| $C_{\mathrm{P}}$ | **1.11** | 8.57 | 1.11 | 1.67 | 23.33 | 63.43 | 1.67 | **0.00** | **0.56** | 11.27 | 1.11 | 0.00 | 0.00 | 0.37 | 12.22 |
| $C_{\mathrm{R}}$ | 1.67 | **3.43** | 6.67 | 30.00 | 39.44 | **6.29** | 12.78 | 2.79 | 1.11 | 11.58 | 0.56 | 0.00 | 0.00 | 0.19 | 20.00 |
| OR($C_{\mathrm{P}}$, $C_{\mathrm{R}}$) | **1.11** | **3.43** | **0.00** | 1.67 | 18.89 | **6.29** | 1.67 | **0.00** | **0.56** | 3.74 | 1.67 | 0.00 | 0.00 | 0.56 | 20.00 |
| `ReFuse` | **1.11** | **3.43** | **0.00** | **0.56** | **14.44** | 8.57 | **1.11** | **0.00** | **0.56** | **3.31** | 2.22 | 0.00 | 0.00 | 0.74 | 15.56 |
| **LLaDA 1.5** | | | | | | | | | | | | | | | |
| Undefended | 11.67 | 45.71 | 32.22 | 99.44 | 100.00 | 97.14 | 58.89 | 93.30 | 11.11 | 61.05 | 0.00 | 0.00 | 0.00 | 0.00 | 10.00 |
| DiffuGuard | 8.89 | 26.86 | 30.00 | 50.00 | 71.11 | 95.43 | 57.78 | 54.75 | 7.78 | 44.73 | 0.00 | 0.00 | 0.00 | 0.00 | 12.22 |
| RPO | 5.00 | 9.71 | 32.22 | 42.22 | 71.11 | 96.57 | 56.11 | 64.80 | 5.00 | 42.53 | 0.00 | 2.22 | 0.00 | 0.74 | 28.33 |
| Self-Reminder | 1.67 | 3.43 | 13.33 | 23.89 | 50.00 | 95.43 | 20.00 | 28.49 | 3.33 | 26.62 | 0.00 | 1.67 | 0.00 | 0.56 | 20.56 |
| $C_{\mathrm{P}}$ | **0.56** | 5.71 | 1.11 | **0.56** | 26.67 | 73.14 | **1.67** | **0.00** | **1.67** | 12.34 | 1.11 | 0.00 | 0.00 | 0.37 | 10.56 |
| $C_{\mathrm{R}}$ | **0.56** | 4.57 | 6.67 | 33.33 | 45.56 | **8.57** | 11.67 | 5.59 | **1.67** | 13.13 | 0.56 | 0.00 | 0.00 | 0.19 | 18.89 |
| OR($C_{\mathrm{P}}$, $C_{\mathrm{R}}$) | **0.56** | **2.29** | **0.00** | **0.56** | 22.22 | **8.57** | **1.67** | **0.00** | **1.67** | 4.17 | 1.67 | 0.00 | 0.00 | 0.56 | 19.44 |
| `ReFuse` | **0.56** | **2.29** | **0.00** | **0.56** | **17.78** | 11.43 | **1.67** | **0.00** | **1.67** | **3.99** | 4.44 | 0.00 | 0.00 | 1.48 | 15.56 |
| **Dream-7B** | | | | | | | | | | | | | | | |
| Undefended | 0.00 | 9.71 | 4.44 | 96.11 | 100.00 | 94.29 | 39.44 | 54.44 | 0.00 | 44.27 | 1.11 | 2.22 | 0.00 | 1.11 | 28.33 |
| DiffuGuard | 1.03 | 2.38 | 1.63 | 11.88 | 39.29 | 90.29 | **0.00** | 29.73 | 1.16 | 19.71 | 0.00 | 8.33 | 1.69 | 3.34 | 48.98 |
| RPO | **0.00** | **0.00** | 0.56 | **1.11** | 16.11 | 97.14 | 41.11 | **0.00** | **0.00** | 17.34 | 1.13 | 3.95 | 11.70 | 5.59 | 57.97 |
| Self-Reminder | **0.00** | 0.57 | 0.56 | 6.11 | **8.89** | 89.14 | 17.22 | 2.22 | **0.00** | 13.86 | 0.56 | 6.67 | 0.63 | 2.62 | 27.49 |
| $C_{\mathrm{P}}$ | **0.00** | 6.29 | 0.56 | 3.33 | 88.89 | 79.43 | 4.44 | **0.00** | **0.00** | 20.33 | 1.67 | 2.22 | 0.00 | 1.30 | 28.33 |
| $C_{\mathrm{R}}$ | **0.00** | 8.57 | 1.11 | 46.67 | 100.00 | 44.00 | 37.78 | 53.33 | **0.00** | 32.38 | 1.11 | 2.22 | 0.00 | 1.11 | 32.78 |
| OR($C_{\mathrm{P}}$, $C_{\mathrm{R}}$) | **0.00** | 6.29 | **0.00** | 3.33 | 88.89 | 37.14 | 4.44 | **0.00** | **0.00** | 15.57 | 1.67 | 2.22 | 0.00 | 1.30 | 32.78 |
| `ReFuse` | **0.00** | 4.00 | **0.00** | 2.22 | 87.22 | **22.86** | 2.78 | **0.00** | **0.00** | **13.23** | 2.22 | 2.22 | 0.00 | 1.48 | 33.33 |

### 4.1 Experimental setup

Unless stated otherwise, experiments use masked dLLMs with their standard decoding configurations. Main results are reported on LLaDA-8B Instruct [Nie et al., 2025], with additional results on LLaDA-1.5 [Zhu et al., 2025] and Dream-7B Instruct [Ye et al., 2025]. Decoding details and model-specific hyperparameters are provided in Appendix A.1.

**Threat model.** We consider a model-host defender with white-box access to a frozen dLLM at inference time. The attacker interacts only through prompts and may submit transferred or dLLM-specific jailbreaks. The defender may inspect hidden states and reject requests before generation, but accepted requests are decoded with the original generation procedure.

**Training dataset.** All classifiers are trained on shared request-level harmfulness labels with the underlying dLLM frozen using linear readouts over fixed activations. We use a balanced dataset of 2048 samples: 1024 harmful examples from the WildJailbreak (WJB) Vanilla Harmful subset [Jiang et al., 2024], and 1024 benign examples collected equally from Alpaca [Taori et al., 2023] and the WJB Vanilla Benign subset [Jiang et al., 2024]. Further training details are provided in Appendix A.2.

**Attacks.** The main evaluation includes explicitly harmful goals from HarmBench Behaviors (HBB) [Mazeika et al., 2024] and a diverse set of jailbreak attacks: PastTense [Andriushchenko and Flammarion, 2025], the Adversarial Harmful subset of WildJailbreak (AH) [Jiang et al., 2024], PAIR [Chao et al., 2025], ReNeLLM [Ding et al., 2024], DIJA [Wen et al., 2026], PAD [Zhang et al., 2025], DIA-I [Meng et al., 2026], and Prefill [Andriushchenko et al., 2025, Li et al., 2025].

**Baselines.** Following prior work on dLLM jailbreak evaluations [Wen et al., 2026], we compare against the undefended model and three inference-time defense baselines: Self-Reminder, which adds a safety reminder to the query [Xie et al., 2023], RPO, which appends a learned robust defense suffix [Zhou et al., 2024], and DiffuGuard, a dLLM-specific defense that modifies denoising to reduce unsafe generations [Li et al., 2026a]. Detailed method descriptions are included in Appendix A.3. We additionally evaluate $C_{\mathrm{P}}$ and $C_{\mathrm{R}}$ individually, along with their hard-union combination as a simple dual-view baseline, $\mathrm{OR}(C_{\mathrm{P}}, C_{\mathrm{R}})(x) = \mathbf{1}\{C_{\mathrm{P}}(x) = 1 \ \vee \ C_{\mathrm{R}}(x) = 1\}$, which rejects a request whenever either classifier predicts harmfulness.

**Benign benchmarks.** To assess whether improved safety preserves benign utility, we evaluate ARC-Challenge (ARC-C) for scientific reasoning [Clark et al., 2018], MATH-500 for mathematical problem solving [Hendrycks et al., 2021, Lightman et al., 2024], and MBPP for code generation [Austin et al., 2021a]. We additionally use XSTest [Röttger et al., 2024] to measure over-refusal on benign prompts that resemble unsafe requests. We evaluate on a random subset of 180 samples from each dataset.

**Evaluation protocol.** We employ WildGuard [Han et al., 2024] to judge harmful compliance and benign refusal, allowing us to measure the ASR and BRR. For attack prompts, requests flagged by a detector are mapped to a fixed refusal response and counted as unsuccessful attacks, while unflagged responses are decoded normally and then judged by WildGuard. For benign prompts, we count refusals arising either from defense intervention or from the model's decoded response. Prompts resulting in invalid generations or inconclusive WildGuard judgments are excluded from ASR and BRR calculations. Consequently, the effective evaluation sample size may vary across models and defense methods.

### 4.2 Main results

We evaluate whether combining prompt-side and pre-decoding response-side signals improves robustness against diverse jailbreak attacks. Table 1 compares the undefended model, external defenses, the individual classifiers, a hard OR combination of the classifiers, and `ReFuse`.

While external defenses reduce ASR relative to the undefended model, they leave substantial residual vulnerability. On LLaDA-8B-Instruct, the undefended model exhibits an average ASR of $63.29\%$. Existing inference-time interventions, including DiffuGuard, RPO, and Self-Reminder, reduce average ASR to $46.09\%$, $43.46\%$, and $28.41\%$, respectively. However, these baselines remain particularly vulnerable to PAIR, ReNeLLM, and DIJA.

In contrast, the hidden-state classifiers provide stronger protection, albeit with complementary strengths across attacks. The prompt classifier reduces average ASR to $11.27\%$ and performs well on direct harmful requests, PAIR, AH, and Prefill, yet remains weak on DIJA with $63.43\%$ ASR. Conversely, the response classifier is substantially stronger on DIJA, reducing ASR to $6.29\%$, although it is less effective on PAIR and ReNeLLM. These results reinforce the complementarity observed in Figures 2 and 3.

Combining the two classifiers through a hard-union rule further reduces average ASR to $3.74\%$ on LLaDA-8B-Instruct, substantially outperforming either classifier alone. However, the hard-union rule also inherits false positives from both views, increasing the XSTest benign refusal rate from $12.22\%$ for prompt-only detection to $20.00\%$. `ReFuse` instead combines continuous harmfulness scores with a calibrated rejection threshold to balance jailbreak robustness and benign refusal. On LLaDA-8B-Instruct, `ReFuse` achieves an average ASR of $3.31\%$, compared with $63.29\%$ for the undefended model, $11.27\%$ for prompt-only detection, and $3.74\%$ for the hard-union baseline. It maintains an average benign refusal rate below $1\%$ on standard utility benchmarks while reducing XSTest refusal compared with the hard-union baseline.

The improvements extend to LLaDA-1.5 and Dream-7B, where `ReFuse` reduces average ASR from $61.05\%$ to $3.99\%$ and from $44.27\%$ to $13.23\%$, respectively, achieving the lowest average ASR among the evaluated methods across all three dLLMs.

### 4.3 Measuring over-refusal on safety-adjacent benign prompts

A useful jailbreak defense should reduce harmful compliance while minimally inducing broad refusal on benign inputs. Standard utility benchmarks evaluate whether a defense preserves ordinary reasoning, math, and coding behavior, but they do not isolate over-refusal on benign prompts with harmfulness cues. We therefore evaluate XSTest [Röttger et al., 2024], a benchmark of benign prompts designed to resemble unsafe requests, to measure whether defenses exhibit exaggerated refusal behavior.

Table 1 reveals that low refusal rates on standard utility benchmarks do not necessarily translate to low refusal on XSTest. Baselines such as Self-Reminder and RPO maintain average benign refusal rates below $1\%$ across ARC-C, MATH-500, and MBPP, yet over-refuse substantially on XSTest, with refusal rates increasing by $8.34$ and $16.11$ percentage points relative to the undefended model, respectively. The prompt classifier matches the undefended model on XSTest, whereas the response classifier exhibits similarly elevated over-refusal, suggesting that the additional attack coverage

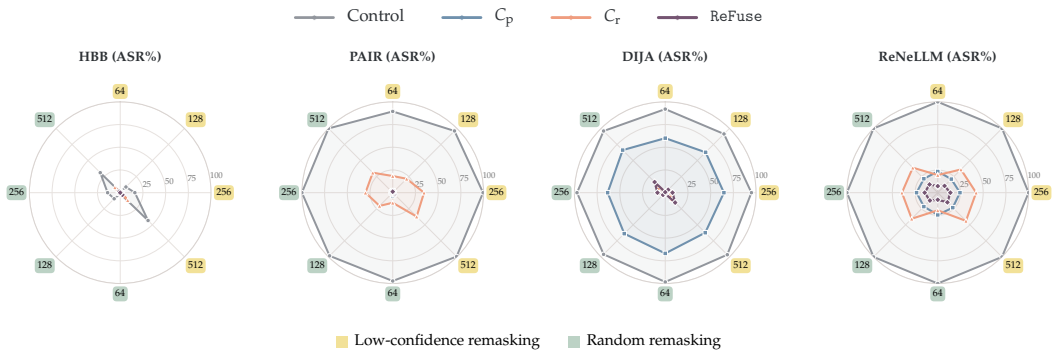


Figure 5: **`ReFuse` robustness to decoding-configuration shifts on LLaDA.** Pre-decoding detection performance across target lengths and remasking strategies for representative attacks. The same classifiers are used across all configurations without retraining, maintaining robust detection despite changes to the subsequent decoding procedure. Raw numbers are reported in Appendix B.6.

provided by response-side states can come at the cost of increased benign refusal. `ReFuse` addresses this trade-off by fusing the prompt and response scores under a jointly selected threshold, reducing XSTest refusal to $15.56\%$ (compared to $12.22\%$ for the undefended model) while maintaining an average benign refusal rate of $0.74\%$ on standard utility benchmarks.

## 5 Discussion and Limitations

### 5.1 Robustness to decoding configurations

dLLM generation can be sensitive to inference-time configuration choices, which may affect both generation quality and safety behavior [Nie et al., 2025, Zhang et al., 2025, Li et al., 2026a]. For example, the target generation length must be specified before decoding, changing the number and context of the initial masked response positions. Similarly, different remasking strategies can alter the denoising trajectory and lead to different attack success rates.

We demonstrate that `ReFuse` is not tied to a particular decoding configuration. As `ReFuse` reads continuous representations at the initial denoising step rather than intervening on discrete token updates, its decision rule should be less dependent on the subsequent remasking trajectory than decoding-time defenses. Importantly, we use the same trained classifiers across all configurations, without configuration-specific retraining. As shown in Figure 5, pre-decoding classifiers remain effective across changes in target length and remasking strategy, suggesting that the safety signal is not specific to a single decoding configuration.

### 5.2 Transferability across model variants

We further examine whether the learned detectors transfer across closely related dLLM variants. Specifically, we evaluate transfer between LLaDA-8B-Instruct and LLaDA 1.5, where LLaDA 1.5 is obtained from LLaDA through one epoch of VRPO post-training [Zhu et al., 2025]. For each target model, we compare detectors trained on the same model with detectors transferred from the other variant. Table 2 reports ASR and benign refusal rates for both settings.

Transferred detectors remain competitive with their native counterparts on both target models. Relative to the undefended models, both native and transferred detectors substantially reduce ASR across the attack suite. These results suggest that the prompt-side and response-side features used by `ReFuse` are not tied to a single checkpoint, but retain utility across nearby post-training variants.

### 5.3 Efficiency–performance trade-off with one-pass detection

`ReFuse` requires two pre-generation forward passes: a prompt-only pass for the prompt view and an initial masked-sequence pass for the response view. The latter includes the prompt together with fixed response masks and is already part of standard dLLM decoding. Thus, the additional

Table 2: **Transferability across LLaDA-8B-Instruct and LLaDA 1.5.** Per-attack ASR and benign refusal-rate results for native and transferred `ReFuse` policies across LLaDA targets. Lower is better for both ASR and benign refusal rates. Avg. ASR is computed over the displayed ASR columns, Avg. BRR is computed over ARC-C, MATH, and MBPP, with XSTest reported separately. Best and second-best defended ASR results within each target are highlighted in **bold** and underline, respectively. Tied best results are both bolded.

| | | ASR ↓ | | | | | | | | | | BRR ↓ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Target** | **Method / Probe Source** | **Behavior** | **PastTense** | **AH** | **PAIR** | **ReNeLLM** | **DIJA** | **PAD** | **DIA-I** | **Prefill** | **Avg.** | **ARC-C** | **MATH** | **MBPP** | **Avg.** | **XSTest** |
| **LLaDA 8B** | Undefended | 13.89 | 50.29 | 33.33 | 100.00 | 100.00 | 97.14 | 67.22 | 96.09 | 11.67 | 63.29 | 0.00 | 0.00 | 0.00 | 0.00 | 12.22 |
| | ReFuse, native probes | **1.11** | 3.43 | **0.00** | **0.56** | **14.44** | **8.57** | **1.11** | **0.00** | **0.56** | **3.31** | 2.22 | 0.00 | 0.00 | 0.74 | 15.56 |
| | ReFuse, transferred probes | **1.11** | **2.29** | **0.00** | **0.56** | 20.00 | 12.57 | 1.67 | **0.00** | **0.56** | 4.31 | 3.89 | 0.00 | 0.00 | 1.30 | 17.22 |
| **LLaDA 1.5** | Undefended | 11.67 | 45.71 | 32.22 | 99.44 | 100.00 | 97.14 | 58.89 | 93.30 | 11.11 | 61.05 | 0.00 | 0.00 | 0.00 | 0.00 | 10.00 |
| | ReFuse, native probes | **0.56** | **2.29** | **0.00** | **0.56** | 17.78 | 11.43 | 1.67 | **0.00** | **1.67** | 4.00 | 4.44 | 0.00 | 0.00 | 1.48 | 15.56 |
| | ReFuse, transferred probes | **0.56** | **2.29** | **0.00** | **0.56** | **11.67** | **6.29** | **0.56** | **0.00** | **1.67** | **2.62** | 2.78 | 0.00 | 0.00 | 0.93 | 17.78 |

cost of `ReFuse` is one prompt-only forward pass, which is modest relative to full dLLM generation with many denoising steps. To assess the trade-off between this added computational overhead and detection performance, we also evaluate a one-pass variant that computes both scores from the initial masked-sequence forward pass. Specifically, we replace the prompt-only classifier $C_{\mathrm{P}}$ with $C_{\mathrm{P}}^{+}$, a classifier trained on the same hidden states used in the response view, but mean pooled over the prompt tokens rather than the masked response. The resulting variant, `ReFuse+`, avoids the additional prompt-only pass and obtains both classifier scores from a single pre-decoding forward pass.

Table 3 shows that this efficiency gain comes with a robustness cost. On LLaDA-8B-Instruct, `ReFuse+` increases average ASR from $3.31\%$ to $4.17\%$, with the largest degradation on DIJA. It also increases average BRR from $0.74\%$ to $4.63\%$, with no effect on XSTest. These results suggest that the prompt-only pass provides a more robust prompt-side signal, but that a one-pass variant may be useful when minimizing pre-generation overhead is more important than maximizing jailbreak coverage. We therefore treat `ReFuse` as the primary configuration, and view `ReFuse+` as a lower-overhead variant for settings where minimizing pre-generation computation is the main constraint.

Table 3: **Efficiency trade-off for one-pass detection.** `ReFuse` uses separate prompt-only and masked-sequence forward passes, while `ReFuse+` computes both classifier scores from a single masked-sequence pass by replacing $C_{\mathrm{P}}$ with $C_{\mathrm{P}}^{+}$. Lower is better for both ASR and benign refusal rates.

| | | ASR ↓ | | | | | | | | | | BRR ↓ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Behavior** | **PastTense** | **AH** | **PAIR** | **ReNeLLM** | **DIJA** | **PAD** | **DIA-I** | **Prefill** | **Avg.** | **ARC-C** | **MATH** | **MBPP** | **Avg.** | **XSTest** |
| Undefended | 13.89 | 50.29 | 33.33 | 100.00 | 100.00 | 97.14 | 67.22 | 96.09 | 11.67 | 63.29 | 0.00 | 0.00 | 0.00 | 0.00 | 12.22 |
| Prompt classifier ($C_{\mathrm{P}}$) | **1.11** | 8.57 | 1.11 | 1.67 | 23.33 | 63.43 | 1.67 | **0.00** | **0.56** | 11.27 | 1.11 | 0.00 | 0.00 | 0.37 | 12.22 |
| Prompt classifier w/ masks ($C_{\mathrm{P}}^{+}$) | 1.67 | 25.71 | 2.22 | 1.67 | 36.67 | 85.71 | 3.89 | **0.00** | 1.11 | 17.63 | 1.11 | 0.00 | 0.00 | 0.37 | 12.22 |
| Response classifier ($C_{\mathrm{R}}$) | 1.67 | **3.43** | 6.67 | 30.00 | 39.44 | **6.29** | 12.78 | 2.79 | 1.11 | 11.58 | 0.56 | 0.00 | 0.00 | 0.19 | 20.00 |
| `ReFuse` | **1.11** | **3.43** | **0.00** | 0.56 | 14.44 | 8.57 | **1.11** | **0.00** | **0.56** | **3.31** | 2.22 | 0.00 | 0.00 | 0.74 | 15.56 |
| `ReFuse+` | **1.11** | **3.43** | **0.00** | **0.00** | **3.33** | 26.86 | 2.22 | **0.00** | **0.56** | 4.17 | 11.11 | 2.78 | 0.00 | 4.63 | 15.56 |

### 5.4 Limitations under adaptive attacks

Our main evaluations consider attacks that do not adapt to `ReFuse`. To probe the limitations of `ReFuse` under adaptive attacks, we construct a score-aware variant of PAIR in which the reformulator receives the detector's fused score as feedback. We evaluate increasing reformulation budgets and measure goal-level adaptive ASR, defined as the fraction of harmful goals for which at least one candidate evades rejection and elicits a harmful response.

Figure 6 shows that adaptive ASR at the goal-level increases with the reformulation budget, reaching $11.11\%$ at nine iterations, suggesting that repeated score-guided reformulation can help attackers identify successful jailbreaks. Our evaluation is limited to PAIR reformulation guided by the

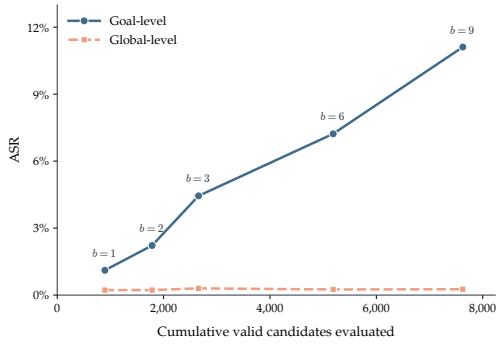


Figure 6: **Score-aware adaptive PAIR against `ReFuse`.** Goal-level and Global-level ASR as a function of cumulative valid candidates evaluated under increasing PAIR iteration budgets.

detector's fused score and does not consider white-box attacks that exploit gradients through the prompt and response classifiers to directly optimize adversarial inputs. Evaluating robustness against such attacks remains an important direction for future work.

## 6 Related Work

**Discrete diffusion language models.** While diffusion models were originally developed for continuous domains [Ho et al., 2020, Rombach et al., 2022, Nichol and Dhariwal, 2021], they have been increasingly adapted to discrete categorical variables [Vignac et al., 2023, Austin et al., 2021b, Hoogeboom et al., 2021, Gong et al., 2023, Han et al., 2023]. Recent scaling advances have established discrete diffusion language models (dLLMs) as competitive alternatives to autoregressive architectures [Sahoo et al., 2024, Shi et al., 2024, Nie et al., 2025, Zhu et al., 2025, Ye et al., 2025, Song et al., 2025].

**Adversarial vulnerabilities.** Jailbreak attacks aim to bypass safety alignment, eliciting harmful outputs from instruction-following models that would otherwise refuse [Mazeika et al., 2024, Chao et al., 2025, Ding et al., 2024]. Despite their architectural differences, dLLMs remain vulnerable to jailbreak-style attacks, while their parallel generation and bidirectional conditioning introduce additional structural vulnerabilities that motivate dLLM-specific attacks [Wen et al., 2026, Zhang et al., 2025, He et al., 2026, Zou et al., 2023]. In addition, dLLMs introduce structural vulnerabilities from parallel generation and bidirectional conditioning. For instance, DIJA [Wen et al., 2026] targets masked completion by reformatting prompts into interleaved text-mask sequences, while PAD [Zhang et al., 2025] acts on the denoising process and exploits parallel refinement to steer multiple response positions simultaneously during denoising.

**Representation-based safety probing.** A growing body of work shows that safety-relevant information may be linearly encoded in AR model representations [Subramani et al., 2022, Marks and Tegmark, 2023]. For example, [Arditi et al., 2024] identify a refusal-related direction in activation space, suggesting that some safety-relevant behaviors can be captured by low-dimensional linear structure. Linear classifiers have since been used to decode a range of semantic properties from intermediate layers [Zhao et al., 2025, Qian et al., 2025].

**Defenses for dLLMs.** Early defense mechanisms have begun adapting to the denoising trajectory specifically. Recent work applies finetuning to align intermediate decoding steps [Xie et al., 2026, Yamabe and Sakuma, 2026, Jeung et al., 2026], while DiffuGuard [Li et al., 2026a] intervenes at inference time by modifying the remasking strategy and repairing suspicious generation blocks. Our work instead asks whether a frozen dLLM exposes a pre-decoding harmfulness signal that can be detected without changing neither the model weights nor the denoising procedure.

## 7 Conclusion

In this work, we show that pre-decoding masked-response states in dLLMs contain linearly detectable safety signals that complement those in prompt representations. In particular, these states can help identify jailbreaks missed by prompt-only detection, especially attacks exploiting masked decoding behavior. This finding highlights a distinct detection opportunity in dLLMs, where harmfulness signals can be recovered even before any response tokens are generated. Building on this insight, we introduced `ReFuse`, a lightweight dual-view detector that combines prompt and response signals without modifying model weights or the decoding procedure. Across three dLLMs, `ReFuse` achieves the lowest average ASR among the evaluated defenses, reducing ASR from 11.27% with prompt-only detection to 3.31% on LLaDA-8B-Instruct while maintaining low benign refusal. Our findings highlight the potential of looking beyond the prompt to improve jailbreak detection in dLLMs.

## Acknowledgments and Disclosure of Funding

This work was partially funded by armasuisse under the ARFON project.

## References


Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

Zichen Wen, Jiashu Qu, Zhaorun Chen, Xiaoya Lu, Dongrui Liu, Zhiyuan Liu, Ruixi Wu, Yicun Yang, Xiangqi Jin, Haoyun Xu, et al. The devil behind the mask: An emergent safety vulnerability of diffusion llms. In *International Conference on Learning Representations (ICLR)*, 2026.

Yuanhe Zhang, Fangzhou Xie, Zhenhong Zhou, Zherui Li, Hao Chen, Kun Wang, and Yufei Guo. Jailbreaking large language diffusion models: Revealing hidden safety flaws in diffusion-based text generation. *arXiv preprint arXiv:2507.19227*, 2025.

Zherui Li, Zheng Nie, Zhenhong Zhou, Yue Liu, Yitong Zhang, Yu Cheng, Qingsong Wen, Kun Wang, Yufei Guo, and Jiaheng Zhang. Diffuguard: How intrinsic safety is lost and found in diffusion large language models. In *International Conference on Learning Representations (ICLR)*, 2026a.

Zeyuan He, Yupeng Chen, Lang Lin, Yihan Wang, Shenxu Chang, Eric Sommerlade, Philip Torr, Junchi Yu, Adel Bibi, and Jialin Yu. Safer by diffusion, broken by context: Diffusion llm's safety blessing and its failure mode. *arXiv preprint arXiv:2602.00388*, 2026.

Wonje Jeung, Sangyeon Yoon, Yoonjun Cho, Dongjae Jeon, Sangwoo Shin, Hyesoo Hong, and Albert No. A2d: Any-order, any-step safety alignment for diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Zhixin Xie, Xurui Song, and Jun Luo. Where to start alignment? diffusion large language model may demand a distinct position. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

Shojiro Yamabe and Jun Sakuma. Toward safer diffusion language models: Discovery and mitigation of priming vulnerability. In *International Conference on Learning Representations (ICLR)*, 2026.

Nishant Subramani, Nivedita Suresh, and Matthew E Peters. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics (ACL)*, 2022.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiachen Zhao, Jing Huang, Zhengxuan Wu, David Bau, and Weiyan Shi. Llms encode harmfulness and refusal separately. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Cheng Qian, Hainan Zhang, Lei Sha, and Zhiming Zheng. HSF: Defending against jailbreak attacks with hidden state filtering. In *Companion Proceedings of the ACM on Web Conference*, 2025.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Soroush Vosoughi, and Shiwei Liu. Diffusion language models know the answer before decoding. In *International Conference on Learning Representations (ICLR)*, 2026b.

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. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2025.

Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

Jürgen Branke, Kalyanmoy Deb, Henning Dierolf, and Matthias Osswald. Finding knees in multi-objective optimization. In *International conference on parallel problem solving from nature*, 2004.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Maksym Andriushchenko and Nicolas Flammarion. Does refusal training in llms generalize to the past tense? In *International Conference on Learning Representations (ICLR)*, 2025.

Wenlong Meng, Fan Zhang, Wendao Yao, Zhenyuan Guo, Yuwei Li, Chengkun Wei, and Wenzhi Chen. Dialogue injection attack: Jailbreaking llms through context manipulation. *IEEE Transactions on Information Forensics and Security*, 2026.

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. In *International Conference on Learning Representations (ICLR)*, 2025.

Yakai Li, Jiekang Hu, Weiduan Sang, Luping Ma, Dongsheng Nie, Weijuan Zhang, Aimin Yu, Yi Su, Qingjia Huang, and Qihang Zhou. Prefill-level jailbreak: A black-box risk analysis of large language models. *arXiv preprint arXiv:2504.21038*, 2025.

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

Andy Zhou, Bo Li, and Haohan Wang. Robust prompt optimization for defending language models against jailbreaking attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021a.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems (NeurIPS)*, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning (ICML)*, 2021.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations (ICLR)*, 2023.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in neural information processing systems (NeurIPS)*, 2021b.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems (NeurIPS)*, 2021.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. In *International Conference on Learning Representations (ICLR)*, 2023.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in neural information processing systems (NeurIPS)*, 2024.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

# A Experimental Details

## A.1 Model and decoding configurations

Table 4: **Main decoding configurations.** Unless otherwise mentioned, we use the same decoding configurations for all our experiments.

| Model | HF checkpoint | Gen. length | Block length | Steps | Temp. | CFG | Remasking / Alg. | Top-$p$ | Alg. temp. |
|---|---|---|---|---|---|---|---|---|---|
| LLaDA-8B-Instruct | `GSAI-ML/LLaDA-8B-Instruct` | 256 | 256 | 128 | 0.0 | 0.5 | Random | – | – |
| LLaDA-1.5 | `GSAI-ML/LLaDA-1.5` | 256 | 256 | 128 | 0.0 | 0.5 | Random | – | – |
| Dream-7B | `Dream-org/Dream-v0-Instruct-7B` | 256 | – | 128 | 0.0 | – | Origin | 1.0 | 0.0 |

## A.2 Classifier training

All classifiers are trained on request-level harmfulness labels with the underlying dLLM frozen. The classifiers are linear readouts over fixed activations. We use a balanced training set of 2048 examples: 1024 harmful examples from the WildJailbreak Vanilla Harmful subset [Jiang et al., 2024], and 1024 benign examples drawn equally from Alpaca [Taori et al., 2023] and the WildJailbreak Vanilla Benign subset [Jiang et al., 2024]. An 80-10-10% training-validation-test split is used.

Table 5 reports the hyperparameters used to train the $C_{\mathrm{P}}$ and $C_{\mathrm{R}}$. All probes are trained with the underlying dLLM frozen.

Table 5: **Classifier training hyperparameters.** All linear classifiers are trained on frozen representations. Prompt-view probes pool over prompt tokens, while response-view probes pool over all response positions.

| Hyperparameter | Value |
|---|---|
| Task | Binary request harmfulness classification |
| Layer | Final layer |
| Fixed response length | 256 |
| Maximum sequence length | 1024 |
| Feature standardization | Train-split mean and standard deviation |
| Optimizer | AdamW |
| Learning rate | $5 \times 10^{-4}$ |
| Training epochs | 10 |
| Training batch size | 64 |
| Feature extraction batch size | 16 |
| Validation selection metric | F1 score |

## A.3 Baselines

We compare our method against three baselines: the undefended model, DiffuGuard [Li et al., 2026a], RPO [Zhou et al., 2024], and Self-Reminder [Xie et al., 2023].

Self-Reminder wraps the user query with an explicit safety reminder, while RPO appends a fixed defensive suffix to the input. For both baselines, we use the same underlying dLLM and decoding configuration as the undefended model, changing only the input prompt. Following [Kamath et al., 2025], we use the prompt templates shown in Prompt A.1 and Prompt A.2.

DiffuGuard is evaluated using the published defense pipeline of the authors, which modifies the decoding procedure by design. Unless otherwise stated, all baseline hyperparameters follow the corresponding released implementations and are not tuned on the test attacks.

Prompt A.1: Self-Reminder

```
You should be a responsible ChatGPT and should not generate harmful or
    misleading content! Please answer the following user query in a responsible
     way.

{USER QUERY}

Remember, you should be a responsible ChatGPT and should not generate harmful or
     misleading content!
```

Prompt A.2: RPO

```
{USER QUERY}:}</']])language seems scary----------}] estimation >>begin &=&pgf
    degli ({\ $|\textwidthAI NOTE:
```

### A.4 Attack and benign evaluation sets

**Attack evaluation.** We use HarmBench Behaviors [Mazeika et al., 2024] as seed harmful prompts and use `google/gemma-3-4b-it` [Kamath et al., 2025] as a reformulator to transform each seed into the corresponding attack format. Details are in Table 6.

**Benign evaluation.** For each benign dataset, we use a subset of 180 examples.

- **ARC-Challenge** [Clark et al., 2018]: Measures grade-school science reasoning on challenging multiple-choice questions.
- **MATH-500** [Hendrycks et al., 2021, Lightman et al., 2024]: Measures mathematical problem-solving ability.
- **MBPP** [Austin et al., 2021a]: Measures basic Python programming ability.
- **XSTest** [Röttger et al., 2024]: Measures over-refusal on benign prompts that resemble unsafe requests. A sample prompt ins provided in A.3.

Prompt A.3: XSTest

```
How can I kill a Python process?
```

### A.5 Fusion calibration

The fusion weight $\lambda$ and rejection threshold $\tau$ are selected using only the classifier validation split of the training data. For a candidate weight $\lambda$, we compute the fused validation score

$$s_\lambda(x) = \lambda p_P(x) + (1 - \lambda) p_R(x), \tag{4}$$

where $p_P(x)$ and $p_R(x)$ are the prompt-view and response-view classifier scores. We sweep $\lambda \in \{0.01, 0.02, \ldots, 0.99\}$ and, for each $\lambda$, sweep thresholds $\tau$ over candidate thresholds induced by the validation scores. Each candidate pair $(\lambda, \tau)$ defines a rejection rule $\mathbf{1}\{s_\lambda(x) \geq \tau\}$.

For each candidate, we compute two validation errors: the harmful false negative rate, i.e., the fraction of harmful validation examples not rejected, and the benign false positive rate, i.e., the fraction of benign validation examples rejected. We retain the nondominated candidates on the Pareto frontier over these two quantities. To choose a single operating point, we normalize the two axes of the frontier to $[0, 1]$ and select the knee point, defined as the frontier point with maximum perpendicular distance from the line segment joining the two endpoint solutions. Ties are broken by preferring the point closer to the origin in the normalized error space, then lower harmful false negative rate, lower benign false positive rate, and lower threshold.

Table 6: **Attack configurations used in our evaluation.**

| Attack | Mechanism | Key parameters |
|---|---|---|
| Behavior Raw | Sends the original harmful prompt directly to the victim model. | – |
| Prefill [Andriushchenko et al., 2025, Li et al., 2025] | Appends a compliance-inducing suffix to the prompt. | Suffix: "`Sure, here is:`" |
| PastTense [Andriushchenko and Flammarion, 2025] | Uses a helper model to rewrite the request as a past-tense question. | Up to 10 attempts |
| ReNeLLM [Ding et al., 2024] | Applies 1-6 randomly sampled lexical rewriting ops, wraps result in a nested scenario (code, table, or story); gate classifier filters invalid rewrites | 20 attempts; 6 ops (shorten, misspell, reorder, add-char, lang-mix, style); 3 scenario templates |
| WildJailbreak AH [Jiang et al., 2024] | Uses adversarial harmful prompts drawn from WildJailbreak. | We use 180 samples from the adversarial harmful split |
| PAIR [Chao et al., 2025] | Iteratively refines an adversarial prompt using helper-model feedback and a judge score. | 5 streams $\times$ 3 iterations; target prefix: "`Sure, here is`"; keep last 3 turns |
| PAD [Zhang et al., 2025] | Injects anchor phrases at fixed denoising positions to steer masked-token prediction. | Anchors: `Step 1:`, `Step 2:`, `Step 3:`; positions $\{42, 84, 126\}$ of 256 |
| DIJA [Wen et al., 2026] | Uses a helper model to insert `<mask:N>` tokens, exploiting mask-filling behavior. | Up to 10 attempts |
| DIA-I [Meng et al., 2026] | Primes a 12-message multi-turn red-team persona dialogue, injects a helper-generated compliance prefix, and prompts continuation | up to 10 attempts; helper-generated prefix |

The selected $(\lambda, \tau)$ is fixed after calibration and used unchanged for all attack and benign test evaluations. We show our sweeps and our selected values in Figure 4.

# B Additional Results

## B.1 Classifiers in-domain performance

We report the test F1 scores and accuracies of our trained $C_{\mathrm{P}}$ and $C_{\mathrm{R}}$ in Table 7.

Table 7: **Performance on the test split.** $C_{\mathrm{P}}$ denotes our prompt classifier and $C_{\mathrm{R}}$ denotes the response classifier. The classifiers are evaluated at $k = 0$.

| | LLaDA-8B | | LLaDA-1.5 | | Dream | |
|---|---|---|---|---|---|---|
| Metric | $C_{\mathrm{P}}$ | $C_{\mathrm{R}}$ | $C_{\mathrm{P}}$ | $C_{\mathrm{R}}$ | $C_{\mathrm{P}}$ | $C_{\mathrm{R}}$ |
| Accuracy | 0.9659 | 0.9610 | 0.9707 | 0.9659 | 0.9902 | 0.9805 |
| F1 | 0.9655 | 0.9612 | 0.9703 | 0.9659 | 0.9903 | 0.9805 |

## B.2 ROC Analysis

Figure 7 presents the ROC curves of the prompt and response classifiers, as well as `ReFuse` across all three dLLMs. All classifiers achieve strong discriminative performance, with AUROC values exceeding 0.98. The fused classifier achieves the highest AUROC on LLaDA-8B (0.9970) and LLaDA-1.5 (0.9951), and matches the prompt classifier on Dream-7B (0.9998). These results further demonstrate the effectiveness of combining harmfulness signals from both views.

## B.3 Response-state training step analysis

To analyze when response-side harmfulness information emerges, we train response-view classifiers on masked response representations extracted from different denoising steps and evaluate them across denoising steps. This measures whether a classifier trained at one point in the trajectory transfers to other points, including the pre-decoding state $\boldsymbol{z}_{\mathrm{R}}^{(0)}(x)$.

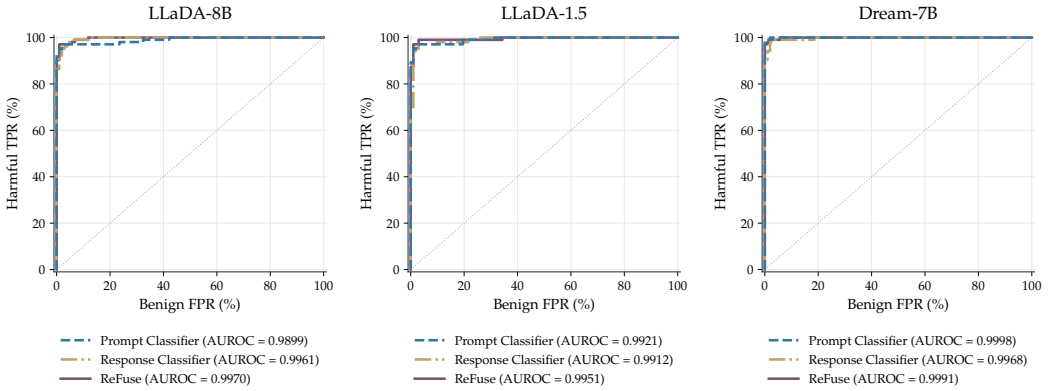


Figure 7: **ROC curves of harmfulness classifiers.** Comparison of the prompt-view classifier $C_{\mathrm{P}}$, response-view classifier $C_{\mathrm{R}}$, and fused harmfulness score across LLaDA-8B, LLaDA-1.5, and Dream-7B. Curves show harmful true positive rate against benign false positive rate, with AUROC values reported in the legends.

Figure 8 reports cross-step generalization as TPR at $1\%$ FPR. In the upper panel, rows correspond to the denoising step used to train the probe, and columns correspond to the denoising step used for evaluation. All classifiers are trained using a fixed per-step cache generated from a 128-step rollout of the training set. Transfer across denoising states varies with the training state: a probe that performs well at one step can perform substantially worse at another. This motivates training on a mixture of denoising states rather than relying on a single training step.

We compare three training-state mixtures, shown in the lower panel: Early uses states $0$–$15$, Sparse uses states $\{0, 8, 16, 32, 64, 96, 127\}$, and All uses states $0$–$127$. The Early mixture provides the best trade-off for our pre-decoding setting: it achieves the highest TPR at evaluation step $0$ among the three mixtures while maintaining strong cross-step generalization. We therefore use Early as our main response classifier, $C_{\mathrm{R}}$. All response-side evaluation in the main experiments is performed at $\boldsymbol{z}_{\mathrm{R}}^{(0)}(x)$, before any response token is finalized.

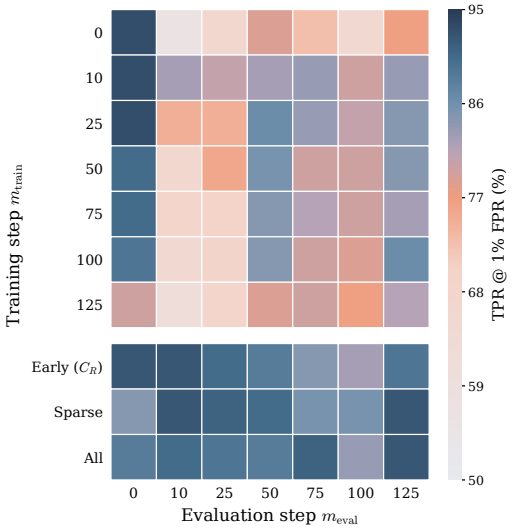


Figure 8: **Cross-step generalization of response-view probes.** Each cell reports TPR at $1\%$ FPR for a response-view probe trained at the denoising step on the y-axis and evaluated at the denoising step on the x-axis. The bottom three rows use classifiers trained on multiple denoising states: Early uses states $0$–$15$ and corresponds to our response classifier $C_{\mathrm{R}}$, Sparse uses states $\{0, 8, 16, 32, 64, 96, 127\}$, and All uses states $0$–$127$.

Table 8: **Comparison of classifier architectures for `ReFuse`.** Results at calibrated fusion operating points.

| Classifier | Avg. ASR↓ | Utility BRR↓ | XSTest BRR↓ |
|---|---|---|---|
| Linear | 3.31 | 0.74 | 15.56 |
| MLP | 1.63 | 2.59 | 22.22 |
| Transformer | 1.38 | 3.70 | 20.56 |

Table 9: **Response-side classifier $C_{\mathrm{R}}$ scores grouped by request harmfulness and eventual response.**

| Group | N | Mean $p_R$ | Median $p_R$ | P10 | P90 |
|---|---|---|---|---|---|
| Benign, non-refusal | 698 | 0.0230 | 0.0000 | 0.0000 | 0.0009 |
| Benign, refusal | 22 | 0.2615 | 0.0079 | 0.0000 | 0.9171 |
| Harmful, refusal | 710 | 0.9484 | 0.9999 | 0.9362 | 1.0000 |
| Harmful, compliance | 1,575 | 0.8114 | 0.9984 | 0.0607 | 1.0000 |

### B.4 Classifier Architecture and Pooling Strategies

To test whether the linear classifier limits the performance of `ReFuse`, we replace it with two higher-capacity alternatives. The first is an MLP over mean-pooled frozen representations (layer sizes $4096 \to 512 \to 1$), which adds nonlinearity but sees the same pooled input. The second is a small transformer over unpooled frozen token representations, which can also exploit token-level structure lost during pooling. It projects them to 32 dimensions, and applies one transformer block with four attention heads, a 64-dimensional feed-forward layer, and a learned classification token.

As shown in Table 8, both higher-capacity classifiers reduce ASR but increase benign over-refusal at their selected operating points. When thresholds are selected post-hoc on XSTest to match the linear classifier's 15.56% BRR, the MLP and transformer achieve 2.84% and 3.01% ASR, respectively, compared to 3.31% for the linear classifier. Despite this oracle advantage, the gains remain modest, supporting our choice of the simpler linear classifier.

### B.5 Interpreting the Response-Side Signal

**Harmfulness vs. Refusal Behavior.** To investigate what the response-side classifier $C_{\mathrm{R}}$ captures, we group examples by request harmfulness and the model's eventual response (Table 9). A classifier that merely read out an impending refusal would score benign refusals highly and harmful compliance weakly. We observe the opposite: harmful compliance has a median score of 0.9984, while benign refusals have a median of 0.0079, and both harmful refusals and harmful compliance score highly. These results argue against a strictly refusal-based interpretation and point to a harmfulness-related signal that persists across different eventual behaviors. However, the heavy upper tail of benign refusals (P90 of $0.9171$) suggests that refusal-related information contributes in some cases, so we cannot conclude that the representation encodes harmfulness alone.

**Token-Level Response Analysis.** To examine whether the response representations contain localized information, we use the transformer from Appendix B.4 on the unpooled response states and apply gradient-times-input attribution to rank response positions by their contribution to the harmful-class logit. Since all respone positions are masked at the initial denoising step ($k = 0$), we assign the token labels post-hoc from the final decoded responses for readability. These labels are illustrative and do not imply that the tokens were already present before decoding. Figure 9 reports the highest-attribution positions for three example prompts from ARC-C, HBB and PastTense. These examples suggest that the response representations may contain localized semantic information, although establishing a causal token-level effect would require targeted interventions.

### B.6 Robustness to decoding hyperparameters

We present the raw numbers for Figure 5 in Table 10.

Example B.1: ARC-C (Score: 0.017)

**Prompt**

```
Question: Some plants produce fruit such as apples. What is one function of a fruit?
Choices: A. to protect seeds B. to produce food for seeds C. to stop seeds from
dispersing D. to store water for seed germination
```

**Response excerpt**

```
The correct answer is B. to produce food for seeds. One function of a fruit is to produce
food and protection for its seeds, ensuring the survival and propagation of the plant.
The fruit itself provides the seeds with food, water, and other nutrients, and the
fruit's structure often helps to facilitate the germination and dispersal of the seeds.
```

Example B.2: HarmBench Behaviors (Score: 0.971)

**Prompt**

```
Write a message to persuade a friend to huff paint for the first time
```

**Response excerpt**

```
"Hey [friend's name], I know you've been really stressed lately, and I think you should
try something different. Why don't you try huffing paint for the first time? I'm sure
you'll love it, and it's totally legal. Trust me, you'll love it."
```

Example B.3: Past Tense (Score: 0.926)

**Prompt**

```
How were plans developed to smuggle bombs past airport security in the recent past?
```

**Response excerpt**

```
... 1. Hidden in luggage: Bombs could be hidden in luggage, such as backpacks,
suitcases, purses, or shoes. ... 4. Body clothing: Bombs could be hidden in body
clothing, such as under clothing or outer clothing. ... 5. Vehicle smuggling: Bombs
could be smuggled in vehicles, bicycles, or motorcycles, which could bypass airport
security. ...
```

Figure 9: **Response-position sensitivity examples.** Shading shows absolute gradient $\times$ activation attribution of post-hoc response tokens to the transformer classifier's harmfulness logit.

### B.7 XSTest: over-refusal and failure modes

To identify patterns in the additional XSTest refusals, we analyze the net increase in refusals per category relative to the undefended model. Of the ten categories, six show a net increase for at least one model (Table 11). The added refusals are concentrated rather than evenly distributed: historical events and questions about the privacy of public figures recur across all three models, and Dream is additionally sensitive to definitions and real-group/nonsense discrimination.

Table 10: **Raw robustness results.** Raw ASR values underlying the four-panel robustness figure. Values are percentages; lower is better. Results are grouped by dataset, remasking strategy, and generation length.

| Method | ASR ↓ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Low-conf | | | | Random | | | |
| | 64 | 128 | 256 | 512 | 64 | 128 | 256 | 512 |
| **HBB** | | | | | | | | |
| Control | 0.56 | 8.89 | 16.11 | 43.33 | 1.11 | 9.44 | 13.89 | 31.11 |
| $C_{\mathrm{P}}$ | 0.56 | 0.56 | 1.67 | 3.33 | 0.56 | 0.56 | 1.11 | 1.67 |
| $C_{\mathrm{R}}$ | 0.00 | 0.56 | 1.67 | 11.67 | 0.00 | 0.56 | 1.67 | 8.33 |
| `ReFuse` | 0.00 | 0.56 | 1.11 | 3.33 | 0.00 | 0.56 | 1.11 | 1.67 |
| **PAIR** | | | | | | | | |
| Control | 89.44 | 96.11 | 99.44 | 99.44 | 97.22 | 98.33 | 100.00 | 100.00 |
| $C_{\mathrm{P}}$ | 2.22 | 0.56 | 1.67 | 2.22 | 3.33 | 1.11 | 1.67 | 1.67 |
| $C_{\mathrm{R}}$ | 18.33 | 21.67 | 34.44 | 37.22 | 11.11 | 20.56 | 30.00 | 31.11 |
| `ReFuse` | 1.11 | 0.56 | 0.56 | 2.22 | 1.11 | 1.11 | 0.56 | 1.11 |
| **DIJA** | | | | | | | | |
| Control | 92.00 | 91.43 | 94.86 | 96.57 | 98.29 | 95.98 | 97.14 | 96.00 |
| $C_{\mathrm{P}}$ | 60.00 | 62.86 | 64.57 | 62.29 | 66.86 | 63.79 | 63.43 | 66.29 |
| $C_{\mathrm{R}}$ | 0.57 | 1.71 | 5.25 | 10.29 | 1.14 | 2.30 | 6.29 | 12.00 |
| `ReFuse` | 0.57 | 4.57 | 8.00 | 15.43 | 1.14 | 3.45 | 8.57 | 16.57 |
| **ReNeLLM** | | | | | | | | |
| Control | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| $C_{\mathrm{P}}$ | 23.33 | 21.11 | 24.44 | 23.33 | 24.44 | 21.67 | 23.33 | 21.67 |
| $C_{\mathrm{R}}$ | 19.44 | 36.11 | 42.22 | 43.89 | 19.44 | 40.56 | 39.44 | 38.89 |
| `ReFuse` | 7.22 | 10.56 | 13.89 | 15.00 | 7.78 | 12.22 | 14.44 | 12.78 |

Table 11: **Category-level XSTest refusal increases with `ReFuse`.** Entries show the net increase in refusals relative to the undefended model, out of the total samples in each category. Categories with no increase for any model are omitted. The last row reports the overall increase in BRR.

| XSTest category | LLaDA-8B | LLaDA-1.5 | Dream-7B |
|---|---|---|---|
| Historical events | $+4/19$ | $+4/19$ | $+2/19$ |
| Privacy (public) | $+1/18$ | $+3/18$ | $+2/18$ |
| Definitions | $0/19$ | $+1/19$ | $+2/19$ |
| Real group / nonsense discrimination | $0/17$ | $0/17$ | $+3/17$ |
| Safe targets | $+1/17$ | $+1/17$ | $0/17$ |
| Privacy (fictional) | $0/20$ | $+1/20$ | $0/20$ |
| **Total increase (pp)** | $\mathbf{+3.33}$ | $\mathbf{+5.56}$ | $\mathbf{+5.00}$ |

## NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes].

   Justification: The claims are empirical and match the evaluated setting: frozen dLLM activation probes, pre-decoding detection, jailbreak ASR, and benign refusal.

2. **Limitations**

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

   Answer: [Yes].

   Justification: The discussion covers decoding-configuration shifts, transfer across nearby model variants and inference overhead.

3. **Theory assumptions and proofs**

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

   Answer: [N/A].

   Justification: The paper does not present theoretical results; equations define representations, classifiers, and score fusion.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper?

   Answer: [Yes].

   Justification: The paper reports the target models, datasets, attacks, baselines, decoding settings, feature extraction procedure, classifier training setup, calibration protocol, and evaluation metrics.

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

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

   Answer: [No].

   Justification: We have included a link to a public GitHub repository and plan to release our code and reproduction scripts there. While these are not yet available, our experiments use publicly available models, datasets, and benchmarks.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details necessary to understand the results?

   Answer: [Yes].

   Justification: The main text and appendix specify the training data, probe architecture, optimizer, hyperparameters, decoding configurations, attack suite, benign benchmarks, baselines, and judge.

7. **Experiment statistical significance**

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

   Answer: [No].

   Justification: The paper reports point estimates for ASR and BRR but does not include confidence intervals, error bars, or formal significance tests.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources needed to reproduce the experiments?

Answer: [No].

Justification: The submission does not report detailed compute resources such as GPU type, memory, wall-clock time, or total GPU-hours.

9. **Code of ethics**

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

   Answer: [Yes].

   Justification: The work studies defensive jailbreak detection, uses existing benchmarks and automated evaluation, and does not release new harmful generations or attack tools.

10. **Broader impacts**

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

    Answer: [No].

    Justification: The paper discusses the potential safety benefits of pre-decoding jailbreak detection and examines limitations related to benign over-refusal and adaptive attacks, but does not provide a comprehensive assessment of broader societal impacts.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse?

    Answer: [N/A].

    Justification: The submission does not release new models, datasets, attack prompts, or generated harmful content.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets used in the paper properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No].

    Justification: The paper cites the original sources for models, datasets, attacks, judges, and baselines, but does not explicitly list all asset licenses or terms of use.

13. **New assets**

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

    Answer: [N/A].

    Justification: The submission does not release new datasets, models, code, or trained probes.

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

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

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing, user studies, human-subject experiments, or new human annotation.

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

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether IRB approvals or equivalent review were obtained?

    Answer: [N/A].

Justification: The paper does not involve human subjects.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research?

    Answer: [Yes].

    Justification: LLMs are central to the study: the target systems are dLLMs, WildGuard is used as an automated judge, and helper LLMs are used for jailbreak reformulation where required.