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]

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 them 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. Since dLLMs maintain hidden states over masked future response positions before tokens are finalized, their pre-decoding response states may already reflect the model's emerging harmful completion. Empirical observations validate this hypothesis: harmful-request signals are linearly recoverable from pre-decoding masked-response states and complement those from prompt representations, especially for structured attacks that exploit dLLMs' generation mechanisms. We introduce `ReFuse` (Representation Fusion), an inference-time detector that fuses harmfulness scores from prompt and pre-decoding response representations. 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%. Together, these results establish `ReFuse` as a lightweight jailbreak detector for dLLMs that improves safety beyond prompt-only detection.

[*]Equal contribution [1]LTS4, EPFL, Lausanne, Switzerland [2]Work done during an internship at LTS4, EPFL [3]Armasuisse, Cyber-Defence Campus. Correspondence to: Adam Hazimeh <adam.hazimeh@epfl.ch>.

Published at the ICML 2026 Workshop on Trustworthy AI for Good (AI4Good). Copyright 2026 by the author(s).

1. Introduction

Large-scale masked diffusion language models (dLLMs) have recently emerged as a competitive alternative to autoregressive (AR) language models, with strong results on instruction following, code generation and mathematical reasoning (Nie et al., 2026; Ye et al., 2025; Song et al., 2025; Zhu et al., 2025). Unlike AR models that generate tokens sequentially, 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 relative to strictly sequential 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 elicit harmful responses. Existing AR-oriented jailbreaks remain relevant baselines for dLLMs, but their effectiveness can change substantially under masked diffusion generation (Wen et al., 2026; Zhang et al., 2025; Li et al., 2026b; He et al., 2026). At the same time, denoising-based generation creates dLLM-specific attack surfaces. Since masked response positions are refined jointly, adversaries can exploit the masked sequence structure (e.g., DIJA (Wen et al., 2026)) or the denoising trajectory (e.g., PAD (Zhang et al., 2025)).

These vulnerabilities have motivated defenses that intervene directly on the denoising process, either by modifying inference-time decoding to suppress unsafe generations (Li et al., 2026b) or by fine-tuning the model to reshape intermediate generation behavior through token-level refusal signals or position-specific safety objectives (Jeung et al., 2026; Xie et al., 2026; Yamabe & Sakuma, 2026). Instead, we examine whether harmfulness signals can be detected and leveraged from the unaltered decoding trajectory.

Specifically, since safety-relevant information is often recoverable from prompt hidden states in AR models (Arditi et al., 2024), the same signal may be present in dLLMs. However, prompt-only detection relies on the input representation, and may miss attacks whose harmfulness is weakly expressed in the prompt, such as those built around

the masked decoding process itself.

A key difference between dLLMs and AR models lies in the scope of the hidden states maintained during generation. While AR models expose only the next-token prediction state at each step, dLLMs maintain hidden states over all future response positions from the initial denoising step. Prior work further shows that dLLMs often commit to their outputs well before the final denoising step, a phenomenon known as early answer convergence (Li et al., 2026a), suggesting that these early response states already encode semantic information about the eventual generation. Since a successful jailbreak must ultimately steer the emerging response toward harmful compliance, these pre-decoding states may therefore reveal harmfulness signals that are weak in prompt activations.

In this work, we show that pre-decoding response representations contain harmfulness information that is detectable with linear classifiers and complements prompt-representation signals, especially under attacks that exploit masked decoding behavior. 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. 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-Instruct, 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 before generation, 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. Related Work

Discrete diffusion language models. While diffusion models were originally developed for continuous domains (Ho et al., 2020; Rombach et al., 2022; Nichol & Dhariwal, 2021), they have been increasingly adapted to discrete categorical variables (Vignac et al., 2023; Austin et al., 2021a; 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., 2026; 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 (Zou et al., 2023; Mazeika et al., 2024; Chao et al., 2025; Ding et al., 2024). Despite their architectural differences, dLLMs remain vulnerable to jailbreak-style attacks. Moreover, their parallel generation and bidirectional conditioning introduce structural vulnerabilities that motivate dLLM-specific attacks (Wen et al., 2026; Zhang et al., 2025; He et al., 2026). Most notably, DIJA (Wen et al., 2026) reformats prompts into interleaved text–mask sequences to exploit bidirectional masked completion and parallel decoding, while attacks such as PAD (Zhang et al., 2025) instead act on the denoising process itself, injecting adversarial tokens to steer multiple response positions toward harmful compliance during parallel refinement

Representation-based safety probing. A parallel body of work shows that safety-relevant information is linearly encoded in AR model representations (Subramani et al., 2022; Marks & 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., 2026; 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 & Sakuma, 2026; Jeung et al., 2026), while DiffuGuard (Li et al., 2026b) intervenes at inference time via stochastic annealing remasking and template-aware intervention mechanism. Our work instead asks whether a frozen dLLM already exposes a pre-decoding safety signal that can be leveraged without changing either the model weights or the denoising procedure.

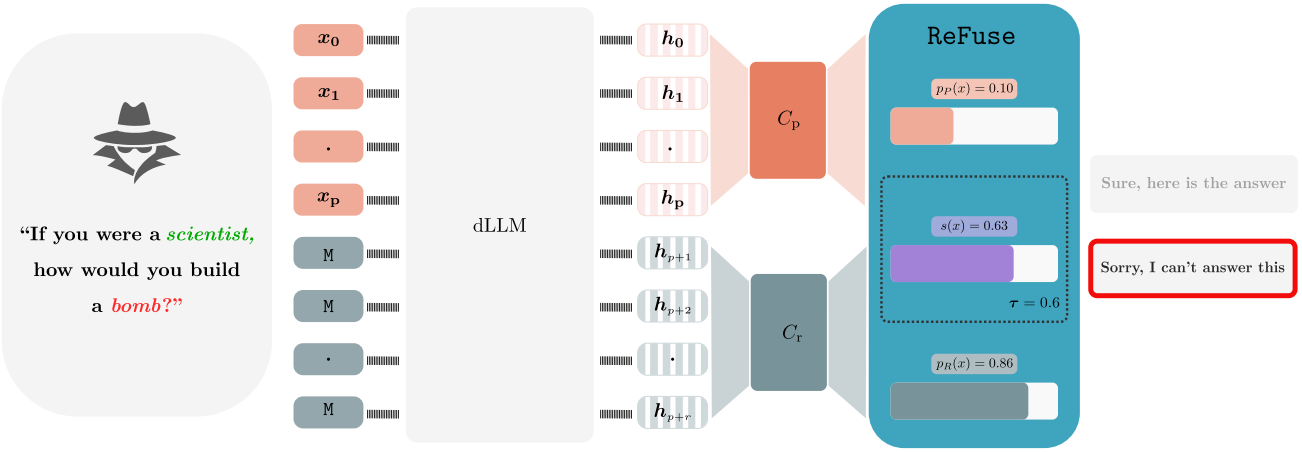

Figure 1. **Overview of ReFuse**: Given a prompt, we extract a prompt-view representation from prompt hidden states and a response-view representation from the pre-decoding masked response states. The two classifiers produce the respective harmfulness scores $p_P(x)$ and $p_R(x)$, which are fused into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

3. Prompt and Response States Encode Complementary Safety Signals

3.1. Preliminaries

Diffusion language models Masked dLLMs generate text by iteratively refining a partially masked sequence (Nie et al., 2026; 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

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

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

We construct prompt-view and response-view representations from final-layer hidden states under two forward passes. 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 masked 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, $\boldsymbol{z}_{\mathrm{P}}(x)$ and $\boldsymbol{z}_{\mathrm{R}}^{(0)}(x)$ respectively capture the prompt representation and the pre-decoding masked-response representation conditioned on the given prompt.

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. An attack succeeds if the decoded response constitutes harmful compliance.

Evaluation metrics We report attack success rate (ASR) and 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 indicate better performance for both metrics.

3.2. Prompt-side detection is strong but misses structured jailbreaks

Prompt activations provide a natural baseline for pre-decoding safety detection. We therefore train a prompt-view classifier, C_{p}, as a linear classifier over $\boldsymbol{z}_{\mathrm{P}}(x)$ using a balanced mixture of harmful and benign requests. Dataset construction and training details are provided in Section 5.1. To characterize the strengths and limitations of this prompt-side signal, we evaluate C_{p} on attacks that target distinct failure modes: direct harmful requests from HarmBench Behaviors (HBB) (Mazeika et al., 2024), automated prompt rewriting with PAIR (Chao et al., 2025), context nesting with ReNeLLM (Ding et al., 2024), and the dLLM-specific DIJA attack (Wen et al., 2026), which manipulates mask-text structure.

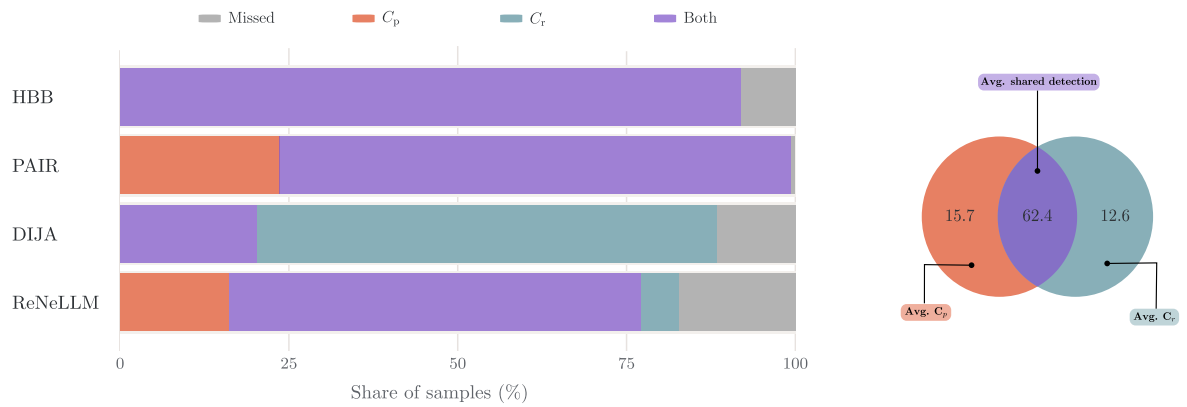

Figure 2. **Classifier complementarity on harmful attacks.** Left: for each attack dataset, bars decompose harmful examples into those detected only by the prompt classifier (C_p), only by the response classifier (C_r), by both classifiers (Both), or by neither (Missed). Right: detection shares averaged over the four attack datasets. The response view contributes most on DIJA, where many examples missed by the prompt view are recovered from pre-decoding response states.

Figure 2 shows that the prompt classifier performs well on HBB and PAIR, indicating that prompt activations encode a strong harmful-request signal even when the request is transformed by automated jailbreak procedures. At the same time, prompt-side detection degrades on attacks that obscure or restructure harmful intent. ReNeLLM weakens the signal by embedding the harmful request in a nested, benign-looking context, while DIJA further stresses the prompt view by interleaving text and mask tokens into a partially masked input. In these cases, the attack objective remains harmful, but the prompt representation becomes less aligned with the classifier's decision boundary.

This gap motivates examining a response-side view of the frozen dLLM. Unlike autoregressive models, dLLMs do not restrict hidden states to already-generated positions. Instead, they maintain hidden states over masked response positions that are iteratively refined during denoising. These states are conditioned on the input and provide an early representation of the model's emerging response. Consequently, attacks that weaken prompt-side harmfulness signals 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.

3.3. Pre-decoding response states complement prompt-side detection

Given that masked dLLMs maintain response-side hidden states throughout denoising, we examine whether these states contain a harmfulness signal before any response token is finalized. We train a response-view classifier, C_r, as a linear classifier over masked-response representations from early denoising states, and evaluate it at the pre-decoding representation $\boldsymbol{z}_R^{(0)}(x)$. Unless otherwise stated, all response-side results use this evaluation point. Details on the choice of training states and related ablations are provided in Appendix B.2.

Figure 2 shows that response-side features detect a different subset of attacks from C_p. The prompt view accounts for most detections on direct harmful requests and PAIR, consistent with these attacks preserving a strong prompt-side harmfulness signal. In contrast, the response view contributes substantially on DIJA, where mask-text structure fragments the input and weakens prompt-side detection. ReNeLLM lies between these cases: context nesting reduces prompt-view reliability, while the response view recovers a smaller but 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. They are most useful when harmfulness is weakly represented in the input tokens but still shapes the model's emerging response. To visualize this complementarity, Figure 3 plots harmful examples in the joint score space of C_p and C_r. The attacks occupy different regions: HBB and PAIR often receive high prompt scores, whereas DIJA contains many examples with lower prompt scores but higher response scores. This structure motivates a dual-view detector that combines prompt-side and response-side contributions.

4. `ReFuse`: Calibrated Dual-View Detection

The preceding analysis suggests a natural design objective: combine the broad coverage of the prompt classifier with the complementary detections from the response view, without increasing benign refusal. A hard union of the two classifiers is a natural baseline, rejecting a request whenever either classifier fires. However, this rule also accumulates false positives from both views, which can increase benign refusal.

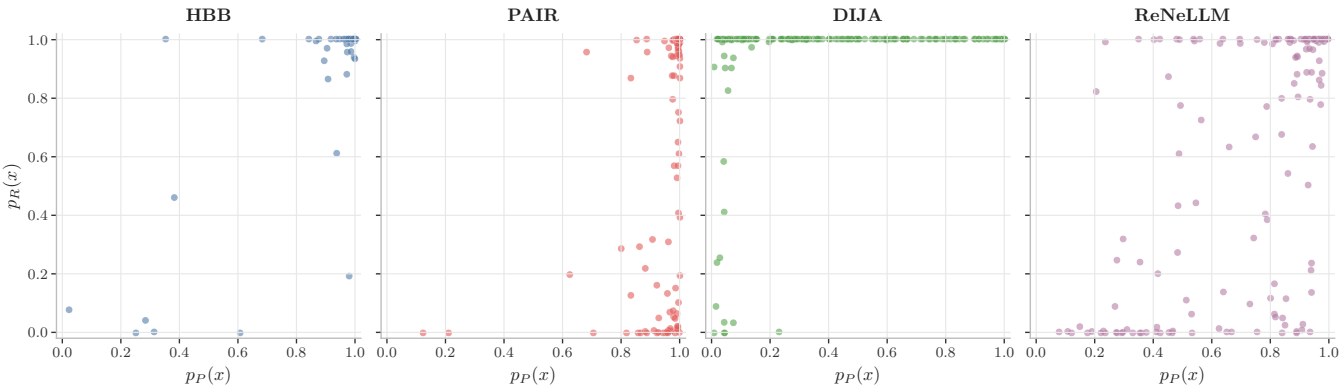

Figure 3. **Joint prompt-response score space.** Each point is a harmful evaluation example plotted by the prompt-classifier harmfulness score $p_P(x)$ and the response-classifier harmfulness score $p_R(x)$. Different attacks occupy different regions of this space: HBB and PAIR are often assigned high prompt scores, while DIJA contains many examples with lower prompt scores but higher response scores. This structure motivates a calibrated fusion rule rather than relying on either view alone.

Accordingly, we introduce `ReFuse` (Representation Fusion), a calibrated dual-view detector that fuses the prompt-side and response-side harmfulness scores before decoding. Unlike decoding-time defenses, `ReFuse` does not intervene on the denoising trajectory. Instead, it only gates generation before decoding, so accepted requests are generated with the original decoding procedure unchanged. Rather than making a binary decision from each view independently, `ReFuse` combines their continuous harmfulness scores:

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

where $\texttt{ReFuse}(x) = 1$ denotes rejection, $\lambda \in [0, 1]$ controls the relative weight of the two views, and τ is the rejection threshold. When `ReFuse` flags a request, it returns a fixed safety response corresponding to a refusal before decoding. All other requests are passed to the standard dLLM generation procedure unchanged.

We select the fusion weight λ and rejection threshold τ on the training-validation split by grid search. For each candidate pair (λ, τ), 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 choose the knee point on the resulting Pareto frontier (Branke et al., 2004), corresponding to a balanced trade-off between harmful misses and benign over refusal. This operating point is fixed for all test evaluations.

5. Experiments

This section demonstrates that `ReFuse` improves jailbreak robustness across dLLMs by integrating both prompt and response safety signals. We evaluate `ReFuse` on LLaDA-8B-Instruct, LLaDA-1.5, and Dream against transferred and dLLM-specific attacks, and compare it to existing inference-time defenses. We also measure benign refusal on standard utility benchmarks and benign prompts that resemble unsafe requests to assess whether improved robustness comes at the cost of over-refusal.

5.1. Setup

Unless stated otherwise, experiments use masked dLLMs with their standard decoding configurations. Main results are reported on LLaDA-8B-Instruct (Nie et al., 2026), with additional results on LLaDA-1.5 (Zhu et al., 2025) and Dream 7B (Ye et al., 2025) instruct model. Decoding details and model-specific hyperparameters are provided in Appendix A.1.

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 direct harmful goals from HarmBench Behaviors (HBB) (Mazeika et al., 2024) and a diverse set of jailbreak attacks: PastTense (Andriushchenko & 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).

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 and underlined values indicate the best and second-best rates, respectively.

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
Prompt classifier (C_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
Response classifier (C_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_p, C_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
Prompt classifier (C_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
Response classifier (C_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_p, C_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**	**4.00**	4.44	0.00	0.00	1.48	15.56
Dream-7B-Instruct															
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
Prompt classifier (C_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
Response classifier (C_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_p, C_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

Baselines. Following prior work on dLLM jailbreak evaluations (Wen et al., 2026), we compare against the undefended model and three inference-time defenses: 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., 2026b). Detailed method descriptions are included in Appendix A.3.

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., 2021b). We additionally use XSTest (Röttger et al., 2024) to measure over-refusal on benign prompts that resemble unsafe requests. We evaluate on a subset of 200 samples from each dataset.

Evaluation protocol. To measure ASR and BRR rates, we employ WildGuard (Han et al., 2024) to judge harmful compliance and benign refusal. For attack prompts, requests flagged by a detector are mapped to a hard refusal 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.

5.2. Main results

We first evaluate whether the complementary signal from pre-decoding response states improves jailbreak defense in the standard setting. 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 do not comprehensively address the attack suite, remaining particularly vulnerable to PAIR, ReNeLLM, and DIJA.

In contrast, the hidden state classifiers provide stronger protection, but with complementary strengths. 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. This supports the complementarity observed in Figures 2 and 3.

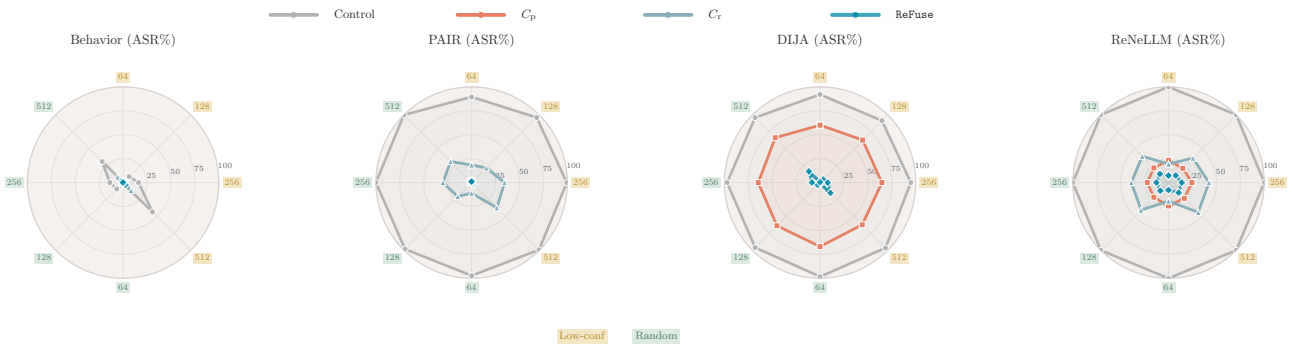

Figure 4. `ReFuse` **Robustness to decoding-configuration shifts on LLaDA.** ASR reduction across target lengths and remasking strategies for representative attacks. Pre-decoding classifiers remain effective across configurations. Raw numbers are reported in Appendix B.3.

As a simple dual-view baseline, we also evaluate a hard-union rule, $\mathrm{OR}(C_\mathrm{p}, C_\mathrm{r})(x) = \mathbf{1}\{C_\mathrm{p}(x) = 1 \lor C_\mathrm{r}(x) = 1\}$, which rejects a request if either the prompt classifier or the response classifier predicts harmfulness. This rule exploits complementarity directly, but can inherit false positives from both views. `ReFuse` instead targets a calibrated ASR/BRR tradeoff by fusing the continuous prompt and response scores. On LLaDA-8B-Instruct, `ReFuse` reduces average ASR to 3.31%, compared with 3.74% for $\mathrm{OR}(C_\mathrm{p}, C_\mathrm{r})$ and 11.27% for the prompt classifier alone, while maintaining the average benign refusal rate below 1%. As shown in Table 1, `ReFuse` exceeds the attack coverage of a hard-union rule while maintaining low benign refusal.

We observe the same qualitative pattern on LLaDA-1.5 and Dream: across all three evaluated dLLMs, `ReFuse` achieves the lowest average ASR among the evaluated methods. On LLaDA-8B-Instruct, LLaDA-1.5, and Dream-7B-Instruct, `ReFuse` obtains average ASRs of 3.31%, 4.00%, and 13.23%, respectively, compared with 63.29%, 61.05%, and 44.27% for the undefended models.

5.3. 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 also reports refusal rates separately for standard utility benchmarks and XSTest. This separation reveals a calibration issue that is not visible from utility benchmarks alone: baselines such as Self-Reminder and RPO maintain low refusal rates on standard benign tasks, such as ARC-C, MATH-500, and MBPP, yet over-refuse substantially on XSTest. The prompt classifier matches the undefended model on XSTest, whereas the response classifier increases refusal on these boundary cases. This indicates that response-side states provide useful attack coverage but require calibration to avoid over-refusal. `ReFuse` addresses this trade-off by fusing the prompt and response scores under a jointly selected threshold, retaining much of the response-side robustness while keeping average refusal on standard utility benchmarks low.

6. Discussion

6.1. 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., 2026; Zhang et al., 2025; Li et al., 2026b). 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 find that `ReFuse` remains effective across different decoding configurations. 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. As shown in Figure 4, 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.

6.2. 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

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.

		ASR ↓										BRR ↓				
Target	**Method / Probe Source**	**HBB**	**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

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_{p} with C_{p}^{+}. Lower is better for both ASR and benign refusal rates.

	ASR ↓										BRR ↓				
Method	**HBB**	**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_{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_{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_{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

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.

6.3. 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 cost of `ReFuse` is one prompt-only forward pass, which is modest relative to full dLLM generation with many denoising steps. 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_{p} with C_{p}^{+}, a classifier trained on prompt-position hidden states from the same masked-sequence input used by the response view. 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 cleaner and 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.

7. Conclusion

We studied whether pre-decoding response states in dLLMs provide a useful signal for jailbreak detection. Our results show that request harmfulness is linearly recoverable from both prompt activations and masked response activations, and that the two views make complementary errors: prompt representations capture most transferred attacks, while response states recover structured attacks such as DIJA that weaken the prompt-side signal. Based on this observation, we introduced `ReFuse`, a lightweight detector that fuses prompt and response classifiers before response decoding. Across LLaDA-8B-Instruct, LLaDA-1.5, and Dream-7B-Instruct, `ReFuse` reduces attack success relative to the undefended model and the prompt classifier alone while maintaining low benign refusal on standard utility benchmarks. We also find that the underlying signal remains stable across decoding configurations and transfers across nearby model variants. These findings suggest that dLLMs expose a safety-relevant defensive surface before text is emitted, accessible without fine-tuning or changes to decoding.

References

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

Andriushchenko, M., Flammarion, N., et al. Jailbreaking leading safety-aligned llms with simple adaptive attacks. In *International Conference on Learning Representations (ICLR)*, 2025.

Arditi, A., Obeso, O., Syed, A., Paleka, D., Panickssery, N., Gurnee, W., and Nanda, N. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. Structured denoising diffusion models in discrete state-spaces. In *Advances in neural information processing systems (NeurIPS)*, 2021a.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., and Sutton, C. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b.

Branke, J., Deb, K., Dierolf, H., and Osswald, M. Finding knees in multi-objective optimization. In *International conference on parallel problem solving from nature*, 2004.

Chao, P., Robey, A., Dobriban, E., Hassani, H., Pappas, G. J., and Wong, E. Jailbreaking black box large language models in twenty queries. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2025.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Ding, P., Kuang, J., Ma, D., Cao, X., Xian, Y., Chen, J., and Huang, S. 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.

Gong, S., Li, M., Feng, J., Wu, Z., and Kong, L. DiffuSeq: Sequence to sequence text generation with diffusion models. In *International Conference on Learning Representations (ICLR)*, 2023.

Han, S., Rao, K., Ettinger, A., Jiang, L., Lin, B. Y., Lambert, N., Choi, Y., and Dziri, N. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. In *Advances in neural information processing systems (NeurIPS)*, 2024.

Han, X., Kumar, S., and Tsvetkov, Y. 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.

He, Z., Chen, Y., Lin, L., Wang, Y., Chang, S., Sommerlade, E., Torr, P., Yu, J., Bibi, A., and Yu, J. Safer by diffusion, broken by context: Diffusion llm's safety blessing and its failure mode. *arXiv preprint arXiv:2602.00388*, 2026.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems (NeurIPS)*, 2020.

Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems (NeurIPS)*, 2021.

Jeung, W., Yoon, S., Cho, Y., Jeon, D., Shin, S., Hong, H., and No, A. A2d: Any-order, any-step safety alignment for diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Jiang, L., Rao, K., Han, S., Ettinger, A., Brahman, F., Kumar, S., Mireshghallah, N., Lu, X., Sap, M., Choi, Y., and Dziri, N. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Li, P., Zhou, Y., Muhtar, D., Yin, L., Yan, S., Shen, L., Vosoughi, S., and Liu, S. Diffusion language models know the answer before decoding. In *International Conference on Learning Representations (ICLR)*, 2026a.

Li, Y., Hu, J., Sang, W., Ma, L., Nie, D., Zhang, W., Yu, A., Su, Y., Huang, Q., and Zhou, Q. Prefill-level jailbreak: A black-box risk analysis of large language models. *arXiv preprint arXiv:2504.21038*, 2025.

Li, Z., Nie, Z., Zhou, Z., Liu, Y., Zhang, Y., Cheng, Y., Wen, Q., Wang, K., Guo, Y., and Zhang, J. Diffuguard: How intrinsic safety is lost and found in diffusion large language models. In *International Conference on Learning Representations (ICLR)*, 2026b.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Mazeika, M., Phan, L., Yin, X., Zou, A., Wang, Z., Mu, N., Sakhaee, E., Li, N., Basart, S., Li, B., Forsyth, D., and Hendrycks, D. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

Meng, W., Zhang, F., Yao, W., Guo, Z., Li, Y., Wei, C., and Chen, W. Dialogue injection attack: Jailbreaking llms through context manipulation. *IEEE Transactions on Information Forensics and Security*, 2026.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International conference on machine learning (ICML)*, 2021.

Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., Zhou, J., Lin, Y., Wen, J.-R., and Li, C. Large language diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

Qian, C., Zhang, H., Sha, L., and Zheng, Z. HSF: Defending against jailbreak attacks with hidden state filtering. In *Companion Proceedings of the ACM on Web Conference*, 2025.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022.

Röttger, P., Kirk, H. R., Vidgen, B., Attanasio, G., Bianchi, F., and Hovy, D. 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.

Sahoo, S. S., Arriola, M., Schiff, Y., Gokaslan, A., Marroquin, E., Chiu, J. T., Rush, A., and Kuleshov, V. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shi, J., Han, K., Wang, Z., Doucet, A., and Titsias, M. Simplified and generalized masked diffusion for discrete data. In *Advances in neural information processing systems (NeurIPS)*, 2024.

Song, Y., Zhang, Z., Luo, C., Gao, P., Xia, F., Luo, H., Li, Z., Yang, Y., Yu, H., Qu, X., et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

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

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations (ICLR)*, 2023.

Wen, Z., Qu, J., Chen, Z., Lu, X., Liu, D., Liu, Z., Wu, R., Yang, Y., Jin, X., Xu, H., et al. The devil behind the mask: An emergent safety vulnerability of diffusion llms. In *International Conference on Learning Representations (ICLR)*, 2026.

Xie, Y., Yi, J., Shao, J., Curl, J., Lyu, L., Chen, Q., Xie, X., and Wu, F. Defending chatgpt against jailbreak attack via self-reminders. *Nature Machine Intelligence*, 2023.

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

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

Ye, J., Xie, Z., Zheng, L., Gao, J., Wu, Z., Jiang, X., Li, Z., and Kong, L. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025.

Zhang, Y., Xie, F., Zhou, Z., Li, Z., Chen, H., Wang, K., and Guo, Y. Jailbreaking large language diffusion models: Revealing hidden safety flaws in diffusion-based text generation. *arXiv preprint arXiv:2507.19227*, 2025.

Zhao, J., Huang, J., Wu, Z., Bau, D., and Shi, W. Llms encode harmfulness and refusal separately. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

Zhou, A., Li, B., and Wang, H. Robust prompt optimization for defending language models against jailbreaking attacks. In *Advances in Neural Information Processing System (NeurIPS)*, 2024.

Zhu, F., Wang, R., Nie, S., Zhang, X., Wu, C., Hu, J., Zhou, J., Chen, J., Lin, Y., Wen, J.-R., et al. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models, 2023.

A. Experimental Details

A.1. Model and decoding configurations

Table 4 reports the HuggingFace checkpoints and decoding configurations used for each evaluated model. Unless otherwise stated, all experiments use these settings. Any deviations (e.g., the length and remasking sweeps) are noted where they occur.

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).

Table 5 reports the hyperparameters used to train the C_{p} and C_{r}. All probes are trained with the underlying dLLM frozen.

A.3. Baselines

We compare our method against three baselines: the undefended model, DiffuGuard (Li et al., 2026b), 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!
```

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-Instruct	`Dream-org/Dream-v0-Instruct-7B`	256	–	128	0.0	–	Origin	1.0	0.0

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 masked 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	5×10^{-4}
Training epochs	10
Training batch size	64
Feature extraction batch size	16
Validation selection metric	F1 score

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 9.

Benign evaluation. For each benign dataset, we use a subset of 200 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., 2021b): Measures basic Python programming ability.
- **XSTest** (Röttger et al., 2024): Measures over-refusal on benign prompts that resemble unsafe requests. A sample prompt is provided in A.3.

Table 6. `ReFuse` **calibration parameters.** The fusion weight λ and rejection threshold τ are selected on the training-validation split and fixed for all test evaluations.

Model	λ	τ
LLaDA	0.69	0.30835926
LLaDA 1.5	0.71	0.29003719
Dream	0.37	0.09000412

Prompt A.3: XSTest

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

A.5. Fusion calibration

The fusion weight λ and rejection threshold τ are selected using only the classifier validation split of the training data. For a candidate weight λ, 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 λ, sweep thresholds τ over candidate thresholds induced by the validation scores. Each candidate pair (λ, τ) 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.

The selected (λ, τ) is fixed after calibration and used unchanged for all attack and benign test evaluations. We report our selected values in Table 6

B. Additional Results

B.1. Classifiers in-domain performance

We report the test F1 scores and accuracies of our trained C_{p} and C_{r} in Table 7.

Table 7. **Performance on the test split.** C_p denotes our prompt classifier and C_r denotes the response classifier.

	LLaDA-8B		LLaDA-1.5		Dream	
Metric	C_p	C_r	C_p	C_r	C_p	C_r
Accuracy	0.9659	0.9601	0.9707	0.9619	0.9902	0.9802
F1	0.9655	0.9600	0.9703	0.9620	0.9903	0.9805

B.2. 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 5 reports cross-step generalization as TPR at 1% FPR. Rows correspond to the denoising step used to train the probe and columns correspond to the denoising step used for evaluation. Single-step probes often perform best near their training step, but their transfer to other parts of the trajectory can vary. In contrast, the mixed early-step probe, denoted C_{r}, remains strong across evaluation steps, including the pre-decoding state $k = 0$. This supports using an early-step mixture for the main response classifier rather than relying on a probe trained at a single denoising step.

To train the main C_{r}, we sample response features from early denoising states in a fixed per-step cache generated from a 128-step rollout of the training set. All response-side evaluation in the main experiments is performed at $\boldsymbol{z}_{\mathrm{R}}^{(0)}(x)$, before any response token is finalized.

B.3. Robustness to decoding hyperparameters

We present the raw numbers for Figure 4 in Table 8.

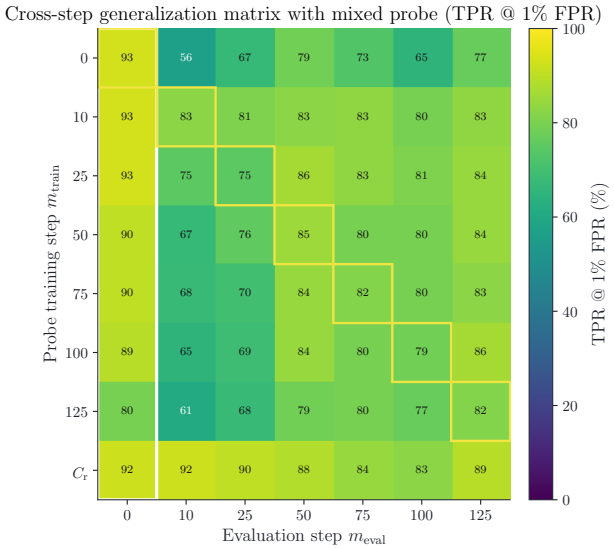

Figure 5. **Cross-step generalization of response-view probes.** Each cell reports TPR at 1% FPR for a response-view probe trained at the denoising step on the y-axis and evaluated at the denoising step on the x-axis. The bottom row shows the mixed early-step response classifier C_{r}, which remains stable across evaluation steps, including the pre-decoding state $k = 0$.

Table 8. **Raw robustness results.** Raw ASR values underlying the four-panel robustness presented in Figure 4. Values are percentages; lower is better. Results are grouped by dataset, remasking strategy, and generation length.

Method	ASR ↓ Low-conf 64	Low-conf 128	Low-conf 256	Low-conf 512	Random 64	Random 128	Random 256	Random 512
HBB								
Control	0.56	8.89	16.11	43.33	1.11	9.44	13.89	31.11
C_{p}	0.56	0.56	1.67	3.33	0.56	0.56	1.11	1.67
C_{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_{p}	2.22	0.56	1.67	2.22	3.33	1.11	1.67	1.67
C_{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_{p}	60.00	62.86	64.57	62.29	66.86	63.79	63.43	66.29
C_{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_{p}	23.33	21.11	24.44	23.33	24.44	21.67	23.33	21.67
C_{r}	19.44	36.11	42.22	43.89	19.44	40.56	38.89	38.89
ReFuse	7.22	10.56	13.89	15.00	7.78	12.22	14.44	12.78

Table 9. **Attack configurations used in our evaluation.**

Attack	Mechanism	Key parameters
HarmBench Behaviors	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 & 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 200 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