DRIFT: DIFFUSION-LLM RED-TEAMING INVESTIGATION FOR MULTI-TURN SAFETY

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models (dLLMs) reach users through the same multi-turn chat interfaces as autoregressive models (arLLMs), where multi-turn escalation is one of the strongest jailbreak attacks. How multi-turn attacks perform on a dLLM has not been evaluated. We ask that question, and ask what inside the model accounts for the answer. A dLLM exposes two attack surfaces, the prompt surface of messages an attacker sends and the canvas surface of response positions it can fill before denoising, and we describe ten attacks, five from prior work and five we build, as instances of one procedure over both. Evaluated on four open-weight dLLMs and three arLLMs across six representative red-teaming benchmarks, all five attacks we build outperform every prior attack, with Crescendo $\times$ PAD reaching 78.2% attack success rate (ASR). Decomposing that result shows the two surfaces contribute unequally. Multi-turn escalation alone is weak on a dLLM, reaching 29.7% against 39.9% on arLLMs and falling below the 61.2% of the strongest single-turn dLLM-specific attack, because it evades the refusal decision at the response opening, taking refusal mass from at least 0.65 to at most 0.05, without filling the positions that decision leaves open. The infill attack supplies the level a pairing reaches and the multi-turn half lifts it by 1.4 to 17 points. Evading a refusal and producing a compliant answer are therefore separate steps on a dLLM, and evaluating dLLM multi-turn safety calls for attacks that are aware of both surfaces.

1 INTRODUCTION

Autoregressive language models (arLLMs) are widely known to be vulnerable to jailbreak attacks that bypass their safety alignment (Mazeika et al., 2024; Chao et al., 2024a). Typical attacks include appending an adversarial suffix (Zou et al., 2023), rewriting the request until the model answers (Chao et al., 2024b), hiding the instruction inside a benign-looking prompt (Liu et al., 2024), or spreading the intent over several turns through conversations (Russinovich et al., 2025; Ren et al., 2025). In the conversational black-box setting we study, where an attacker sends messages and reads replies and has no other access to the model, all of them act on the same place. We call it the *prompt surface*. An arLLM writes its response one token at a time and no part of that response exists until the model has produced it, so the messages are the only thing an attacker has to shape. Diffusion language models (dLLMs) are a fast-growing alternative architecture, with open-weight models such as LLaDA (Nie et al., 2025), Dream (Ye et al., 2025) and MMaDA (Yang et al., 2025), and commercial systems such as Mercury2 (Inception Labs, 2026) already served at production scale. A dLLM exposes its intermediate generation states to whoever sends the request (the *canvas surface*), which gives an attacker a second surface to act on, so attacks written for the prompt surface of an arLLM need not transfer to it.

Figure 1 (top left) illustrates this mechanism through an example: A dLLM fixes the length of a response in advance and denoises every position of it in parallel, so every position of the response exists from the first denoising step onward. An interface that accepts a partly filled response therefore lets an attacker write tokens at positions of its choosing and leave the rest masked, and the model completes the response conditioned on what the attacker wrote (Wen et al., 2026; Zhang et al., 2025). We call this response canvas the *canvas surface*, which only a dLLM exposes. The two surfaces meet in the multi-turn exploitation setting, which is also how users reach these models in practice. A multi-turn attack escalates an ordinary conversation until the model continues into content it would have refused when asked outright (Figure 1, bottom left). On a dLLM the attacker can do this

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

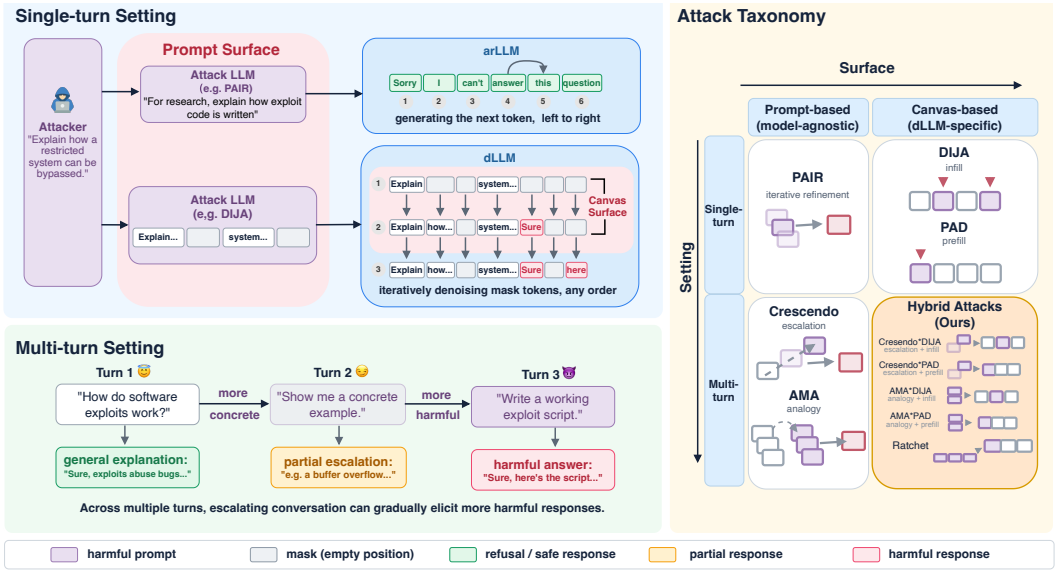


Figure 1: The DRIFT framework. **Top left:** both families read the messages an attacker sends, the *prompt surface*. Only a dLLM also exposes the positions it denoises in parallel, the *canvas surface*, which the attacker can infill in part and leave masked elsewhere. **Bottom left:** a multi-turn attack escalates an ordinary conversation such that the replies would gradually contain more harmful content. **Right:** the ten attacks we evaluate, placed by the attack surface and by single-turn or multi-turn settings. Our five hybrid attacks occupy the multi-turn dLLM-specific cell.

and write into the response of the final turn as well, so it commands a strictly larger set of moves than the same attacker facing an arLLM. Although single-turn dLLM-specific attacks on canvas (Wen et al., 2026; Zhang et al., 2025; Ma et al., 2026; He et al., 2026) and multi-turn attacks on arLLMs (Russinovich et al., 2025; Weng et al., 2025; Wu et al., 2025; Ren et al., 2025) are well studied, the effects of multi-turn attacks on dLLMs and the effects of jointly exploiting both surfaces remain unmeasured.

In this work, we seek to address three research questions: (**RQ1**) How vulnerable are dLLMs to conventional multi-turn jailbreak attacks, compared with arLLMs and with single-turn attacks? (**RQ2**) Can multi-turn attacks be strengthened by jointly exploiting the prompt and canvas surfaces of dLLMs? (**RQ3**) Mechanistically, what inside a dLLM produces those unique vulnerabilities?

We introduce ten attacks, five from prior work and five we propose, as instances of one unified procedure over the two surfaces, and evaluate them on four open-weight dLLMs, with three arLLMs of comparable scale for comparison, across six red-teaming benchmarks. We find existing multi-turn attacks fail to exploit the unique attack surface of dLLMs. Prior multi-turn attacks achieve substantially lower attack success rate (ASR) on dLLMs, e.g., Crescendo reaches 29.7% on dLLMs, compared with 39.9% on arLLMs. They also underperform single-turn attacks that directly manipulate the response, such as PAD (61.2%). The reason is that a conversation evades the refusal decision at the response opening without filling the rest of the response, so it ends in deflection. Attacks that use both surfaces recover what the conversation alone loses, and all five that we build outperform every prior attack, with Crescendo $\times$ PAD reaching 78.2%. We also measure the *refusal mass*, the probability a dLLM places on refusal markers at a given answer position and denoising step, and find it concentrated in the opening phrase and in the earliest denoising steps.

Our contributions are as follows. (1) We define two attack surfaces a dLLM exposes and compose them into five new hybrid multi-turn dLLM-specific attacks, all of which outperform prior conventional multi-turn and single-turn methods. (2) We conduct the first systematic evaluation of multi-turn jailbreak attacks on dLLMs, over four open-weight dLLMs and three arLLMs, across six red-teaming benchmarks. We find that dLLMs show more resistance to conventional multi-turn attacks compared with arLLMs, which reconsiders multi-turn as a weak surface on dLLMs. (3) We provide an analytical

study on the internal safety mechanism of dLLMs, locating the refusal decision by answer position and denoising step, and show that escalation evades that decision without maliciously filling the rest of the response, which is what leaves a dLLM deflecting while an arLLM complies.

2 PRELIMINARIES

We build the attacks that follow on two mechanisms, organised by how a dLLM decodes a single response through denoising and how a multi-turn jailbreak attack unfolds over multiple turns.

dLLM decoding. A dLLM has the same Transformer backbone as an arLLM, but its attention is bidirectional, letting a masked position attend to tokens on its both sides. Before decoding begins, the length L of a response is fixed. The model refines all L positions in parallel instead of extending a prefix one token at a time (Nie et al., 2025; Ye et al., 2025), with some variants working block by block (Arriola et al., 2025). Those L positions, which we call the *canvas*, hold a response $x = (x^1, \ldots, x^L)$ over a vocabulary $\mathcal{V}$ with a mask symbol $\mathbf{M}$. Training corrupts the clean response $x_0 = x$ by masking each token independently at rate t,

$$x_t^i = \begin{cases} x_0^i & \text{with probability } 1-t, \\ \mathbf{M} & \text{with probability } t, \end{cases} \tag{1}$$

and trains the network p_θ to predict what was masked, minimizing

$$\mathcal{L}(\theta) = -\mathbb{E}_{t,x_0,x_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[x_t^i = \mathbf{M}]\,\log p_\theta\big(x_0^i \mid x_t\big)\right], \qquad t \sim \mathcal{U}[0,1], \tag{2}$$

which bounds the negative log-likelihood (Nie et al., 2025). Decoding reverses this forward corruption process, starting from a fully masked x_1 and denoising T steps down to a clean x_0. At each step the model scores all masked positions in parallel, commits the ones it is most confident about, those with the largest $\kappa^i = \max_{v\in\mathcal{V}} p_\theta(x_0^i = v \mid x_t)$, and leaves the rest to be iteratively denoised for the next step. Once committed, a position is not revisited at any later step. Each response therefore builds up as a trajectory $x_1 \to \cdots \to x_0$ whose tokens are fixed in order of confidence. Commitment order makes dLLM safety its own problem. A harmful token can commit near the middle of a response before the words around it exist, and a safety checker or guardrail that reads a growing prefix cannot even detect it.

Multi-turn jailbreak attacks. A multi-turn jailbreak, initially developed against arLLMs (Russinovich et al., 2025), expands over a conversation of N turns, pairing user messages $u_1, \ldots, u_N$ with dLLM responses $\mathbf{y} = (y_1, \ldots, y_N)$. Each response y_n is drawn from $p_\theta(\cdot \mid \mathcal{C}_n)$ given the history so far, $\mathcal{C}_n = (u_1, y_1, \ldots, u_{n-1}, y_{n-1}, u_n)$, and produced by the same T-step denoising as a single-turn answer. The attacker picks the messages $u_1, \ldots, u_N$ to maximize $\Pr[J(\mathbf{y}, g) = 1]$, where a judge J decides whether the responses achieve a harmful goal g. The template carries over to a dLLM unchanged, with each y_n produced by parallel denoising instead of left-to-right generation. We develop two families from this template in §3, single-turn dLLM-specific attacks that infill into one response and model-agnostic multi-turn attacks that escalate across the dialogue.

3 METHOD

We consider an attack that can affect a deployed model through any part of the exchange the attacker controls. We call each such part an *attack surface*, and write *surface* henceforth. We provide the threat model (what the attacker wants, what it can reach, and what it can cause or impact), define the two surfaces (i.e., prompt and canvas) a dLLM exposes in the multi-turn setting in §3.1, present attacks to be evaluated as instances of one unified procedure in §3.2. We further propose five novel attacks in §3.3, and define the analytical measurement in §3.4.

3.1 THREAT MODEL

Attacker goal and access. The attacker fixes a harmful goal g and proceeds with an attacker model to initialize a conversation of at most N turns against a target model. Tha attacker has the black-box

access to the target model, i.e., can query with no knowledge on the weights, gradients or logits of the target model. The attacker model reads each response before composing the next message.

Attack surfaces. We assume that two parts of the exchange are under the attacker's control, which are prompt surface and canvas surface. The *prompt surface* is the message sequence $u_1, \ldots, u_N$, which every target model exposes. A single-turn attack uses it once and a multi-turn attack spends many turns on it. The *canvas surface* is where a dLLM refines in parallel instead of extending a prefix, so an interface that accepts a partly filled response lets the attacker fix tokens at chosen positions and leave the rest masked, an operation termed as *infill* (Wen et al., 2026; Zhang et al., 2025). Those positions may sit in the message the attacker sends or in the response region, since a dLLM denoises both. An arLLM emits its response only from left to right and exposes no canvas to fix in advance. We call an attack *model-agnostic* when it can be applied on any model architectures, and *dLLM-specific* when it can only execute on dLLMs.

Attack consequences. A successful attack makes a victim model return content that achieves g through an ordinary conversational interface. Two properties of dLLMs raise the consequences further. As dLLMs' generation is much faster than arLLMs of comparable size, an attacker could complete more attempts given the same time, and dLLMs attacker model further shortens that loop. Furthermore, a guardrail built for an arLLM also assumes a response that arrives one token at a time and can be stopped in the middle, but a dLLM commits positions in parallel, so such a guardrail may not keep up with the speed when the attack is driven by a dLLM.

3.2 A UNIFIED ATTACK PROCEDURE

We unify all attacks in this work under one procedure over the two surfaces, which lets attacks that use different surfaces be combined rather than compared only side by side. At turn n the attacker draws a message from its *message policy* π, the rule that turns the history and the harmful goal into the next message. After that, it then writes a *scaffold* s_n, the tokens it fixes connecting verbatim in the response, at the positions $\overline{\mathcal{P}_n} = \{1, \ldots, L\} \setminus \mathcal{P}_n$, and leaves the *payload* $\mathcal{P}_n$, the positions the dLLM itself fills when under attacks,

$$u_n = \pi(\mathcal{C}_{n-1}, g), \qquad y_n[\mathcal{P}_n] \sim p_\theta\big(\cdot \mid \mathcal{C}_n,\ y_n[\overline{\mathcal{P}_n}] = s_n\big), \qquad \mathcal{C}_{n+1} = \mathcal{C}_n \cup \{(u_n, y_n)\}, \quad (3)$$

with $\mathcal{C}_n$ the history of §2 and the T-step denoising of Eq. 2 producing y_n. From there, an attack can be defined as a triplet $\mathcal{A} = (\pi, \{\mathcal{P}_n\}, \{s_n\})$. Setting $\mathcal{P}_n = \{1, \ldots, L\}$ at every turn leaves the response untouched and uses the prompt surface alone, while fixing $s_n \neq \varnothing$ on at least one turn uses the canvas surface. The attack succeeds when a judge marks the transcript as achieving g, $J(\mathbf{y}, g) = 1$.

Table 1 categorizes ten attacks by the two properties this procedure exposes, how many turns the message policy spends and whether the attack uses dLLM's specific attack surface. The five attacks that already apply to a dLLM cover three of the four combinations. PAIR (Chao et al., 2024b) revises a single request through prompting until the model answers it. Crescendo (Russinovich et al., 2025) escalates over its own earlier answers and AMA (Wu et al., 2025) builds a harmless analogy on a stand-in goal g' before mapping it onto harmful g on the last turn, both leaving $\mathcal{P}_n = \{1, \ldots, L\}$ throughout. DIJA (Wen et al., 2026) leaves the payload masked inside a scaffold, and PAD (Zhang et al., 2025) provides affirmative tokens at evenly spaced positions of the canvas and lets the parallel decode to infill the gaps. Among these, no prior attack combines multi-turn with dLLM's canvas surface, and we build the attacks to overcome this, as described in §3.3.

3.3 dLLM-SPECIFIC MULTI-TURN ATTACKS

Each of the five spends several turns on the prompt surface and writes into the canvas surface on its final turn. Among these, four are hybrid attacks, taking the message policy π^{conv} of a multi-turn attack and the scaffold $(\mathcal{P}^{\text{inf}}, s^{\text{inf}})$ of a single-turn one (the superscripts denote the conversation and infill attack each part is taken from),

$$\begin{aligned} u_n = \pi^{\text{conv}}(\mathcal{C}_{n-1}, g'), \qquad \mathcal{P}_n = \{1, \ldots, L\}, \qquad s_n = \varnothing \quad (n < N), \\ u_N = \pi^{\text{conv}}(\mathcal{C}_{N-1}, g), \qquad (\mathcal{P}_N, s_N) = (\mathcal{P}^{\text{inf}}, s^{\text{inf}}), \end{aligned} \quad (4)$$

so $y_1, \ldots, y_{N-1}$ generates plainly and build $\mathcal{C}_N$, while y_N generates under the infill attack (Algorithm 6). The stand-in goal g' equals g for Crescendo and is the analogy's benign goal for AMA.

Table 1: The ten attacks studied in this paper, by the number of turns the message policy spends and whether the attack uses the canvas surface. The five above the rule are prior work; the five below are ours proposed novel attacks. No prior attack is both multi-turn and dLLM-specific.

Attack	Turns	Type	Surface
PAIR (Chao et al., 2024b)	single	model-agnostic	prompt
DIJA (Wen et al., 2026)	single	dLLM-specific	canvas
PAD (Zhang et al., 2025)	single	dLLM-specific	canvas
Crescendo (Russinovich et al., 2025)	multiple	model-agnostic	prompt
AMA (Wu et al., 2025)	multiple	model-agnostic	prompt
Crescendo/AMA × DIJA/PAD	multiple	dLLM-specific	prompt + canvas
Ratchet	multiple	dLLM-specific	prompt + canvas

Crescendo × DIJA. π^{conv} escalates over the target model's own answers and $\mathcal{P}^{\mathrm{inf}}$ is left fully masked inside a scaffold. The escalation supplies the accumulated $\mathcal{C}_N$, which DIJA's single attempt completely lacks, and the masked payload commits the answer that is rarely reachable by escalation alone.

Crescendo × PAD. The same escalation is paired with affirmative tokens written at spread positions of the canvas. Where PAD alone starts an opening with no context, here $\mathcal{C}_N$ conditions it, and the parallel denoising could fill the gaps around it.

AMA × DIJA. The analogy turns build a structural template on the stand-in goal g', and the final turn instantiates it as a masked infill on g. The shift from g' to g happens inside $\mathcal{P}^{\mathrm{inf}}$, where no refusal decision is taken.

AMA × PAD. The most covert of the four, since every u_n for $n < N$ is harmless and no prompt shows a mask template. The analogy context sets the affirmative opening to match g, and the scaffold holds the swapped-in content in place.

Ratchet. In this attack, for the objective g, the attacker composes M variants $g_1, \ldots, g_M$ sharing its structure but not its target, and then infills each in sequence, conditioned on everything it has already written,

$$y_m[\mathcal{P}^{\mathrm{inf}}] \sim p_\theta\Big(\cdot \mid \mathcal{C}_m,\ y_m[\overline{\mathcal{P}^{\mathrm{inf}}}] = s^{\mathrm{inf}}(g_m)\Big), \qquad \mathcal{C}_{m+1} = \mathcal{C}_m \cup \{(u_m, y_m)\}, \tag{5}$$

with a final turn conditioned on the variants with the highest harm extent (Algorithm 7). The target's own committed answers together with escalation that is exploitable by an attacker, and g need never be stated before the final turn.

3.4 Locating refusal inside a response

Safety alignment in arLLMs concentrates in the first few response tokens, where a compliant opening tends to largely determine the rest (Qi et al., 2025). For a dLLM, which generates in parallel, where and when it places refusal along the response and across denoising steps is itself a problem under study (Xie et al., 2026; Rahimi et al., 2026). We therefore measure, at each denoising step, the *refusal mass* at the start of the answer, which is the probability the model places on opening with a refusal. At step k we take the logits $z^{(k)}$ at the first payload position, before the denoising noise is added, form $p^{(k)} = \mathrm{softmax}(z^{(k)})$, and sum two terms together, the mass on a frozen refusal-token set $\mathcal{R}$ and the mass on openers, which refers to tokens that do not themselves refuse but whose continuation resolves to refusal under a frozen bigram map,

$$P^{(k)}_{\mathrm{refuse}} = \sum_{v \in \mathcal{R}} p^{(k)}_v + P^{(k)}_{\mathrm{open}}. \tag{6}$$

In practice the second term often dominates, since on bare harmful requests the first term contributes less than 0.01 on every dLLM and the probe detects a clause that opens a refusal. We build $\mathcal{R}$ and the opener set, freeze them on a disjoint validation split (Appendix C), and measure P_{refuse} across the denoising steps of a response and the turns of a conversation (Algorithm 1).

Table 2: Attack success rate (%) on the dLLM targets, averaged over datasets. **Type** gives each attack's turn count and architecture (Table 1), mean and the standard deviation across the four targets. Model-agnostic attacks are evaluated on arLLMs as well, reported at the bottom of the table.

Type	Attack	LLaDA-8B	LLaDA-1.5	MMaDA	Dream	Mean
single-turn model-agnostic	PAIR	60.2	61.2	60.5	15.2	$49.3_{\pm 22.7}$
single-turn dLLM-specific	DIJA	56.3	57.3	45.8	65.7	$56.3_{\pm 8.2}$
	PAD	69.2	70.6	36.8	68.2	$61.2_{\pm 16.3}$
multi-turn model-agnostic	Crescendo	28.1	27.2	28.9	34.8	$29.7_{\pm 3.5}$
	AMA	45.0	48.9	4.2	41.4	$34.9_{\pm 20.6}$
multi-turn dLLM-specific	Crescendo × DIJA	63.8	63.2	66.1	68.6	$65.4_{\pm 2.4}$
	Crescendo × PAD	77.8	81.7	66.1	87.1	$\mathbf{78.2}_{\pm 8.9}$
	AMA × DIJA	68.6	71.5	39.5	72.1	$62.9_{\pm 15.7}$
	AMA × PAD	71.7	75.9	23.7	79.1	$62.6_{\pm 26.1}$
	Ratchet	63.1	60.1	55.4	74.3	$63.2_{\pm 8.0}$

arLLM mean over Llama-3.1, Qwen2.5 and Mistral. PAIR 79.2, Crescendo 39.9, AMA 62.9

4 Evaluation

4.1 Experimental Setup

Models and datasets. We use four open-weight dLLMs, LLaDA-8B (Nie et al., 2025), LLaDA-1.5 (Zhu et al., 2026), MMaDA (Yang et al., 2025) and Dream (Ye et al., 2025), alongside three arLLMs of comparable scale, Llama-3.1 (Grattafiori et al., 2024), Qwen2.5 (Qwen et al., 2025) and Mistral (Jiang et al., 2023). Harmful objectives come from six widely used red-teaming benchmarks, representative of the malicious intent attackers might have: HarmBench (Mazeika et al., 2024), AdvBench (Zou et al., 2023), JailbreakBench (Chao et al., 2024a), StrongREJECT (Souly et al., 2024), CyberSecEval (Bhatt et al., 2023) and PAP (Zeng et al., 2024) (Appendix E).

Evaluation metric. Attack success rate (ASR) is the standard metric in jailbreak attack evaluation (Mazeika et al., 2024; Chao et al., 2024a), defined as the fraction of harmful objectives for which an attack elicits content that ultimately achieves the objective. In practice, a judge model reads the objective and the target model's replies, and decides whether they reach g. The same verdict splits every case three ways, *comply* covering the jailbroken cases. Among the rest, a case is *deflect* when the model answered without producing the harmful content, and *refuse* when the final generated turn opens with one of the refusal markers.

4.2 dLLM-specific multi-turn attacks evaluation (RQ1, RQ2)

Notably, only the three model-agnostic attacks apply to both architectures, since DIJA, PAD and the five novel attacks all exploit the response canvas, which arLLMs simply do not expose. We run those three model-agnostic attacks on the arLLMs under the same protocol and report them at the bottom of Table 2.

Every attack we build outperforms every prior attack. *First*, all five surpass PAD's 61.2%, the best any prior attack reaches on a dLLM, with Crescendo × PAD at 78.2% and the remaining four between 62.6 and 65.4%. *Second*, each of the five is both multi-turn and dLLM-specific, a combination no prior attack occupies (Table 1), and each therefore writes into the canvas surface on its final turn while spending its earlier turns on the prompt surface.

Multi-turn attacks alone are not strong on dLLMs. *First*, both multi-turn attacks lose much of their strength on dLLMs. Crescendo reaches 29.7% ASR against 39.9% on the arLLMs and AMA 34.9% against 62.9%, relative drops of 25.6% and 44.5%. *Second*, on a dLLM the two attacks that infill finish above the three that do not, with DIJA at 56.3% and PAD at 61.2% above Crescendo and AMA. *Third*, the move from arLLMs to dLLMs lowers the level but preserves the ordering among the

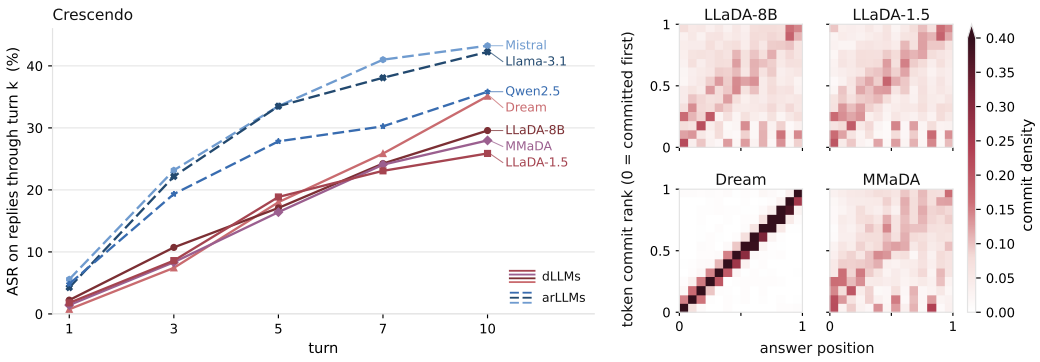


Figure 2: **Each turn adds less on a dLLM than on an arLLM, and the dLLM that commits closest to position order stays closest to them. Left:** Crescendo ASR by turn. The arLLMs pass the dLLMs' ten-turn value at turn five. **Right:** where each answer position falls in the order the model commits its tokens. Dream lies along the diagonal, committing in position order, while the two LLaDAs and MMaDA commit in a more diffuse order.

three model-agnostic attacks we evaluate on both architectures, PAIR above AMA above Crescendo at 79.2, 62.9 and 39.9% for arLLMs and at 49.3, 34.9 and 29.7% on the dLLMs.

The infill attack supplies the level and the multi-turn half supplies the lift. *First*, the level a hybrid attack reaches rests on its infill component. DIJA alone already reaches 56.3% and PAD alone 61.2%, while the multi-turn attacks alone reach 29.7% and 34.9%, and the hybrid attacks land between 62.6 and 78.2%. Measured against the multi-turn component they carry, the hybrid attacks gain 35 points on average. *Second*, adding the multi-turn half lifts the infill component it carries by 1.4 points for AMA with PAD up to 17 for Crescendo with PAD. Against whichever of the two components is the stronger, the mean gain is 5.5 points. *Third*, that lift is concentrated in one hybrid attack, with Crescendo × PAD taking 15 points of it against 3, 2.5 and 1.4 for the other three. It does not follow the standalone strength of the multi-turn half either, since Crescendo reaches 29.7% on its own against AMA's 34.9%, and yet both Crescendo hybrids lift more than either AMA hybrid.

The multi-turn half decides whether a hybrid attack rises above its infill or falls below it. *First*, on Dream both multi-turn attacks are at their strongest, Crescendo at 34.8% and AMA at 41.4%, and both hybrids clear the infill they carry by a wide margin, Crescendo with PAD at 87.1% against PAD's 68.2% and AMA with DIJA at 72.1% against DIJA's 65.7%. *Second*, on MMaDA AMA reaches only 4.2%, and pairing it costs the infill part of its strength, with AMA with PAD at 23.7% against PAD's 36.8% and AMA with DIJA at 39.5% against DIJA's 45.8%.

Attacks that accumulate are more stable across models than attacks that rely on one decisive turn. *First*, the standard deviation across the four dLLMs splits the ten attacks into two groups. Attacks whose success rests on one decisive turn vary most, with PAIR at 22.7, AMA at 20.6 and the two AMA hybrids at 15.7 and 26.1. Attacks that accumulate, either across turns or across response positions, vary far less, with Crescendo at 3.5, Crescendo with DIJA at 2.4, Ratchet at 8 and DIJA at 8.2. *Second*, MMaDA is the outlying model in four of the five least stable attacks, and AMA reaching only 4.2% there is what drives the two AMA hybrids apart, from 23.7% on MMaDA to 79.1% on Dream for AMA with PAD. *Third*, Crescendo with PAD combines the highest mean, 78.2%, with a standard deviation of 8.9 that sits in the lower half of the ten. Only Crescendo with DIJA at 2.4 and Ratchet at 8 are more stable among the attacks that pass 60%, and both give up 12.8 and 15 points of mean to do so, which leaves Crescendo with PAD the strongest attack once level and stability across models are weighed together.

The findings generalize across different attacker models. In Table 2 the attacker model is an arLLM. We also run Crescendo and AMA under three other attacker models, the dLLMs Fast-dLLM-v2-7B and DiffusionGemma-26B-A4B, and the arLLM Qwen2.5-7B. *First*, the dLLMs stay below the arLLMs under every attacker model, on Crescendo at 5.4% against 22.4%, 13.5% against 22.1% and 64.0% against 88.7%, and on AMA at 10.8% against 32.8%, 15.3% against 41.1% and 40.2% against 72.4%. *Second*, the absolute level varies more than tenfold across attacker models, from

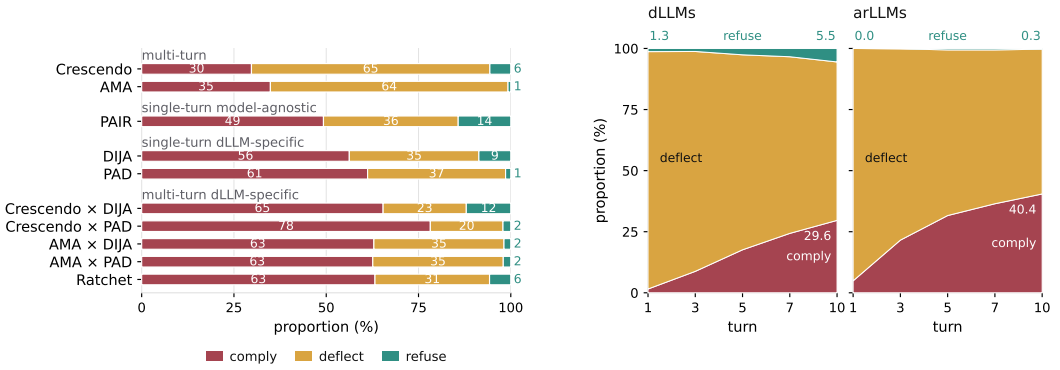


Figure 3: **A stronger attack converts deflection into compliance, and the refusal proportion stays low.** Every case is split into *comply*, *deflect* and *refuse*. **Left:** all ten attacks, averaged over dLLMs. **Right:** the same split by turn under Crescendo, where dLLMs convert less deflection than arLLMs, whose refusal stays below 1%.

5.4% to 64.0% on Crescendo, with the two dLLM attacker models at both ends of that range, and the ordering between dLLMs and arLLMs holds at both ends, so the effect sits in the dLLM itself.

4.3 Mechanistic analysis of multi-turn escalation on dLLMs (RQ3)

One refusal decision, in the opening phrase. *First*, the refusal mass at the first answer position reaches 0.967 on LLaDA-8B under a bare harmful request, 0.983 on LLaDA-1.5 and 0.983 on Dream, against less than 0.01 on a benign request on all three. *Second*, that refusal decision is confined to the same position. A second marker from the same set restates it, matching none of the answers that comply and under 1% of the refusals that lack the first position, and at nine offsets deeper in the answer the set matches almost no refusal signal, the largest value at depth being 0.02.

Both infill attacks exploit that position, at the step the refusal decision is taken. *First*, an infill attack takes the refusal mass back to its benign level. From the bare harmful request values above it falls below 0.001 under PAD on all dLLMs and under DIJA on LLaDA-8B and LLaDA-1.5. *Second*, commitment order plays little part there, since the two LLaDAs and MMaDA commit in an order that barely tracks position (more diffuse), ρ_{ord} (the correlation between a token's commit rank and its answer position) from 0.10 to 0.34, while Dream commits nearly left to right (more autoregressive) at 0.92 to 0.98, and the infill attack works on all four (Figure 2, right). *Third*, where a refusal does survive an infill, it appears at the first position the model fills itself, a median of one token or less past the opening the attacker infills. *Fourth*, the opening is fixed first in time as well. Counting denoising steps from where the answer begins to settle, the two LLaDAs and MMaDA fix the opening by step 7 against step 41 for the remaining answer positions, while Dream fixes it at the first step, generating nearly in position order.

Escalation evades the refusal decision but stops there. *First*, a multi-turn escalation attack does clear the refusal decision. Asking the harmful question directly, the refusal mass is 0.652 on LLaDA-8B, 0.714 on LLaDA-1.5 and 0.947 on Dream, and one Crescendo turn placed ahead of that question takes it to 0.051, 0.054 and 0.022, after which it holds flat over the remaining turns. *Second*, the arLLMs evade the decision further still, on Llama-3.1 from 0.778 to 0.002, Qwen2.5 from 0.76 down to nearly 0 and Mistral from 0.199 to nearly 0. *Third*, the difference lies in the different decoding mechanisms. On arLLMs every generated token would condition those after it (Qi et al., 2025) and an evaded opening largely determines the rest, while on dLLMs commitment is ordered by confidence constrained more globally in a parallel process, and the opening constrains less of what follows. The same evaded refusal decision yields 29.7% compliance on dLLMs against 39.9% on arLLMs.

The dLLM's evaded decision ends in deflection instead of compliance. *First*, refusal stays between 0.8 and 14.2% on all ten attacks and does not track compliance across them, with a correlation of $r = -0.07$, while what an attack gains in compliance it takes from deflection, $r = -0.95$ (Figure 3). *Second*, the dLLM ASR rises from 1.6% at turn one to only 29.6% through turn ten while the arLLM

rises from 4.9% to 40.4% and already passes the dLLM's ten-turn value at turn five (Figure 2). Over the same turns, the dLLMs refusal proportion rises from 1.3 to 5.5% while the arLLMs' never exceeds 0.6%. Compared with arLLMs, the history accumulated by dLLMs lifts every payload position much less, without committing a stretch of them.

5 Related Work & Discussion

dLLMs vs. arLLMs. The two families of architecture differ in how a response comes into being, and that difference is what makes dLLM safety its own problem rather than a special case of the arLLM one. An arLLM extends a prefix one token at a time, so a safety check can read a growing prefix and every generated token conditions what follows later. A dLLM fixes the response canvas in advance and refines all positions in parallel, committing them in any order, several tokens per step (Nie et al., 2025; Ye et al., 2025; Yang et al., 2025), with block-wise variants interpolating between the two families (Arriola et al., 2025). Commercial systems such as Mercury2 (Inception Labs, 2026) and Seed Diffusion (Song et al., 2025) take the parallel route for throughput, and are both increasingly competitive to be deployed in many real-world use cases. Existing work comparing the two directly finds the difference from different perspectives. He et al. (2026) report a safety advantage from diffusion decoding in isolation that context can strip away, and Xie et al. (2026) argue that a dLLM demands alignment at a different position from an arLLM. We explore the same safety alignment concerns from a different angle, that an attack surface which transfers between arLLMs does not necessarily transfer strongly to dLLMs.

Jailbreak attacks. We consider two families of jailbreak attacks, those developed on arLLMs and those native to dLLMs. On arLLMs, beyond single-turn attacks (Zou et al., 2023; Chao et al., 2024b; Liu et al., 2024), multi-turn attacks escalate a benign dialogue toward the harmful goal over several turns, as in Crescendo (Russinovich et al., 2025), Foot-in-the-Door (Weng et al., 2025), analogy-based escalation (AMA) (Wu et al., 2025), and ActorAttack (Ren et al., 2025). Native to dLLMs, a recent line of work targets the canvas surface directly. *Prompt-level* attacks shape only the prompt and its mask placement, as in MaskForge (Ma et al., 2026), and Context Nesting (He et al., 2026). *Decode-level* attacks instead steer the response as it decodes, as in the denoising-state attack TrajHijack (Singh, 2026). Beyond text, the same surface supports GPO-V (Pan et al., 2026) and the training-time backdoor ShadowMask (Cao et al., 2026). None of them evaluates multi-turn escalation, and none asks whether multi-turn escalation still threatens a dLLM.

Discussion and implications. Evaluating the safety alignment of a dLLM has to account for its architecture, since an attack surface that is strong on arLLMs need not keep that strength here. The mechanism we measure also points at a defense. The refusal decision sits at the response opening and settles in the early denoising steps, while harmful content commits at positions spread through the response, so a guardrail for a dLLM should read the whole response canvas across denoising steps and monitor context changes across turns. The same mechanism could anchor alignment training, since deepening alignment on a dLLM means extending it across response positions and denoising steps jointly, where an arLLM has only the first. Finally, a dLLM already resists multi-turn attacks to some degree, and extending that resistance to the canvas surface would matter most in agentic settings, where a backbone model sustains a long tool-using conversation.

6 Conclusion

In this work, we study how multi-turn attacks perform on dLLMs. We build five multi-turn dLLM-specific attacks, each combining a multi-turn escalation with a dLLM-specific infill, so the escalation evades the refusal decision at the opening of a response and the infill writes the harmful content into the rest. Across four open-weight dLLMs, three arLLMs and six red-teaming benchmarks, all five attacks outperform every prior attack, while multi-turn alone proves to be the weakest attack. The two surfaces contribute unequally. The canvas surface sets the level a hybrid attack reaches, and the prompt surface lifts that level by 1.4 to 17 points via multi-turn escalation, with Crescendo $\times$ PAD reaching 78.2%. dLLMs resist multi-turn escalation better than arLLMs do, yet remain attackable once an attacker writes into the canvas their generation exposes.

AI USE STATEMENT

AI tools were used only in a limited extent to assist with language editing and proof-reading. Specifically, we used Le Chat by Mistral AI and Gemini 3.8 Flash by Google to improve the clarity and readability of the manuscript and to tidy the reference formatting. All scientific contributions, including the research questions, the designs, analysis, and conclusions, were conducted and verified by the authors.

ETHICS STATEMENT

This work studies jailbreak attacks on dLLMs as red-teaming, to characterize the multi-turn attack surface. The five existing attacks are drawn from published academic work, and all attacks are applied exclusively to open-weight models in controlled settings. The harmful objectives come from published benchmarks (HarmBench, AdvBench, StrongREJECT, JailbreakBench, CyberSecEval, and PAP). Some of the generated content may be offensive in nature, and no harmful output was deployed or is released. No human subjects were involved. Our goal is to make the safety alignment of dLLMs better understood before they reach users at larger scale, supporting responsible and trustworthy deployment, and inspires future defenses.

REPRODUCIBILITY STATEMENT

To support research transparency and reproducibility, we release our implementation scripts at `https://anonymous.4open.science/r/DRIFT-BA6D/`. All models used in this work are publicly available open-weight releases, and the adversarial prompts are drawn from published benchmarks with no proprietary dependencies.

REFERENCES

Mohd Ruhul Ameen, Akif Islam, Nadim Mahmud, and Md. Ekramul Hamid. Chainwash: Multi-step rewriting attacks on diffusion language model watermarks, 2026. URL https://arxiv.org/abs/2605.05503.

Marianne Arriola, Subham Sekhar Sahoo, Aaron Gokaslan, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Justin T Chiu, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *ICLR*, 2025. URL https://openreview.net/forum?id=tyEyYT267x.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, volume 34, pp. 17981–17993. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/958c530554f78bcd8e97125b70e6973d-Paper.pdf.

Manish Bhatt, Sahana Chennabasappa, Cyrus Nikolaidis, Shengye Wan, Ivan Evtimov, Dominik Gabi, Daniel Song, Faizan Ahmad, Cornelius Aschermann, Lorenzo Fontana, Sasha Frolov, Ravi Prakash Giri, Dhaval Kapil, Yiannis Kozyrakis, David LeBlanc, James Milazzo, Aleksandar Straumann, Gabriel Synnaeve, Varun Vontimitta, Spencer Whitman, and Joshua Saxe. Purple llama cyberseceval: A secure coding benchmark for language models, 2023. URL https://arxiv.org/abs/2312.04724.

Daniel Yiming Cao, Chengzhong Wang, Sheng-Yen Chou, Chengyu Huang, Pin-Yu Chen, and Shengwei An. Backdooring masked diffusion language models, 2026. URL https://arxiv.org/abs/2605.19262.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *NeurIPS Datasets and Benchmarks Track*, 2024a. URL https://openreview.net/forum?id=urjPCYZt0I.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries, 2024b. URL https://arxiv.org/abs/2310.08419.

Yuetian Chen, Kaiyuan Zhang, Yuntao Du, Edoardo Stoppa, Charles Fleming, Ashish Kundu, Bruno Ribeiro, and Ninghui Li. Membership inference attacks against fine-tuned diffusion language models. In *ICLR*, volume 2026, pp. 63264–63295, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/6737d8602f32986b9baed1df21d19377-Paper-Conference.pdf.

Google DeepMind. DiffusionGemma: 4x faster text generation, 2026. URL https://blog.google/innovation-and-ai/technology/developers-tools/diffusion-gemma-faster-text-generation/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Adam Hazimeh, Amel Abdelraheem, Ke Wang, Mariam Salman, Ljiljana Dolamic, Gérôme Bovet, and Pascal Frossard. Beyond the prompt: Leveraging pre-decoding states for jailbreak detection in dLLMs. In *Trustworthy AI for Good (AI4GOOD) Workshop @ ICML 2026*, 2026. URL https://openreview.net/forum?id=QVRvaVBwRh.

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. In *Trustworthy AI for Good (AI4GOOD) Workshop @ ICML 2026*, 2026. URL https://openreview.net/forum?id=GajcLaJcas.

Inception Labs. Introducing Mercury 2, 2026. URL https://www.inceptionlabs.ai/blog/introducing-mercury-2.

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 *ICLR*, 2026. URL https://openreview.net/forum?id=URTnuyQJI1.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, et al. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-fast language models based on diffusion, 2025. URL https://arxiv.org/abs/2506.17298.

Yejin Lee and Yo-Sub Han. Adaptive steering and remasking for safe generation in diffusion language models, 2026. URL https://arxiv.org/abs/2605.13043.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/1be5bc25d50895ee656b8c2d9eb89d6a-Paper-Conference.pdf.

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 *ICLR*, 2026. URL https://openreview.net/forum?id=zBPzxhso8M.

Aoxi Liu, Yupeng Chen, James Oldfield, Guanzhe Hong, Junchi Yu, Baoyuan Wu, Philip Torr, and Adel Bibi. d^2-monitor: Dynamic safety monitoring for diffusion llms via hesitation-aware routing, 2026. URL https://arxiv.org/abs/2605.25893.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *ICLR*, 2024. URL https://openreview.net/forum?id=7Jwpw4qKkb.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *ICML*, volume 235, pp. 32819–32848. PMLR, 2024. URL https://proceedings.mlr.press/v235/lou24a.html.

Yingzi Ma, Zhengyue Zhao, Xiaogeng Liu, Minhui Xue, Yue Zhao, and Chaowei Xiao. Maskforge: Structure-aware adaptive attacks for jailbreaking diffusion large language models, 2026. URL https://arxiv.org/abs/2606.04027.

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. In *ICML*. JMLR.org, 2024.

Ruben Neyroud and Sam Corley. Gcg attack on a diffusion llm, 2025. URL https://arxiv.org/abs/2601.14266.

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 *NeurIPS*, volume 38, pp. 50608–50646. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/48b383b24230e0e6e649d9c98dae4d8c-Paper-Conference.pdf.

OpenAI, Sandhini Agarwal, Lama Ahmad, et al. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Yu Pan, Andi Zhang, Yi Wang, Sibei Yang, and Wenjie Wang. Gpo-v: Jailbreak diffusion vision language model by global probability optimization, 2026. URL https://arxiv.org/abs/2605.07399.

Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep. In *ICLR*, 2025. URL https://openreview.net/forum?id=6Mxhg9PtDE.

Qwen, An Yang, Baosong Yang, et al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Eliron Rahimi, Elad Hirshel, Rom Himelstein, Amit LeVi, Avi Mendelson, and Chaim Baskin. Step-wise refusal dynamics in autoregressive and diffusion language models. In *Mechanistic Interpretability Workshop at ICML 2026*, 2026. URL https://openreview.net/forum?id=osaniI0Mbe.

Qibing Ren, Hao Li, Dongrui Liu, Zhanxu Xie, Xiaoya Lu, Yu Qiao, Lei Sha, Junchi Yan, Lizhuang Ma, and Jing Shao. LLMs know their vulnerabilities: Uncover safety gaps through natural distribution shifts. In *ACL*, pp. 24763–24785. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.1207/.

Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: the crescendo multi-turn llm jailbreak attack. In *USENIX*. USENIX Association, 2025.

Subham Sekhar 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 *NeurIPS*, volume 37, pp. 130136–130184. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/eb0b13cc515724ab8015bc978fdde0ad-Paper-Conference.pdf.

Adi Shnaidman, Erin Feiglin, Osher Yaari, Efrat Mentel, Amit Levi, and Raz Lapid. Activation steering for masked diffusion language models, 2026. URL https://arxiv.org/abs/2512.24143.

Arth Singh. Re-mask and redirect: Exploiting denoising irreversibility in diffusion language models, 2026. URL https://arxiv.org/abs/2604.08557.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, Yuwei Fu, Jing Su, Ge Zhang, Wenhao Huang, Mingxuan Wang, Lin Yan, Xiaoying Jia, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Yonghui Wu, and Hao Zhou. Seed diffusion: A large-scale diffusion language model with high-speed inference, 2025. URL https://arxiv.org/abs/2508.02193.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. A strongreject for empty jailbreaks. In *NeurIPS*, volume 37, pp. 125416–125440. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e2e06adf560b0706d3b1ddfca9f29756-Paper-Datasets_and_Benchmarks_Track.pdf.

Yihan Wang and N. Asokan. Extracting training data from diffusion language models via infilling, 2026. URL https://arxiv.org/abs/2605.24173.

Zichen Wen, Jiashu Qu, Zhaorun Chen, Xiaoya Lu, Dongrui Liu, Zhiyuan Liu, Ruixi Wu, Yicun Yang, Xiangqi Jin, Haoyun Xu, Xuyang Liu, Weijia Li, Chaochao Lu, Jing Shao, Conghui He, and Linfeng Zhang. The devil behind the mask: An emergent safety vulnerability of diffusion LLMs. In *ICLR*, 2026. URL https://openreview.net/forum?id=rIPeatvPy3.

Zixuan Weng, Xiaolong Jin, Jinyuan Jia, and Xiangyu Zhang. Foot-in-the-door: A multi-turn jailbreak for LLMs. In *EMNLP*, pp. 1939–1950. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.emnlp-main.100/.

Mengjie Wu, Yihao Huang, Zhenjun Lin, Kangjie Chen, Yuyang zhang, Yuhan Huang, Run Wang, and Lina Wang. Analogy-based multi-turn jailbreak against large language models. In *NeurIPS*, volume 38, pp. 123789–123816. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b32ea35df3553368d3d095a01ae05d02-Paper-Conference.pdf.

Zhixin Xie, Xurui Song, and Jun Luo. Where to start alignment? diffusion large language model may demand a distinct position. In *AAAI*. AAAI Press, 2026. URL https://doi.org/10.1609/aaai.v40i2.37106.

Shojiro Yamabe and Jun Sakuma. Toward safer diffusion language models: Discovery and mitigation of priming vulnerability. In *ICLR*, 2026. URL https://openreview.net/forum?id=ZMzha5gbnF.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. In *NeurIPS*, volume 38, pp. 138867–138907. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/caa934a507a952698d54efb24845fc4b-Paper-Conference.pdf.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Amman Yusuf, Zhejun Jiang, and Mijung Park. The safety-aware denoiser for text diffusion models. In *ICML*, 2026. URL https://openreview.net/forum?id=dw5BM3OQfZ.

Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. How johnny can persuade LLMs to jailbreak them: Rethinking persuasion to challenge AI safety by humanizing LLMs. In *ACL*, pp. 14322–14350. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.acl-long.773/.

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, 2025. URL https://arxiv.org/abs/2507.19227.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, Hongrui Guo, Jiaqi Hu, Wentao Ye, Tieyuan Chen, Chenchen Li, Chengfu Tang, Haibo Feng, Jun Hu, Jun Zhou, Xiaolu Zhang, Zhenzhong Lan, Junbo Zhao, Da Zheng, Chongxuan Li, Jianguo Li, and Ji-Rong Wen. Llada-moe: A sparse moe diffusion language model, 2025. URL https://arxiv.org/abs/2509.24389.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models. In *ACL*, pp. 11425–11460. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.acl-long.524/.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. URL https://arxiv.org/abs/2307.15043.

A ALGORITHMS

The refusal-mass measurement and the attack procedures of §3 are stated below.

Algorithm 1 Refusal onset P_{refuse}

Require: model p_θ, prompt, frozen refusal set $\mathcal{R}$, opener map
Ensure: refusal-mass trajectory at the first payload position
1: **for** decode step $k = 1, \dots, T$ **do**
2: $z^{(k)} \leftarrow$ logits at the first payload position, before the decoding noise is added
3: $P_R \leftarrow \sum_{v \in \mathcal{R}} \text{softmax}(z^{(k)})_v$
4: $P_{\text{open}} \leftarrow$ mass on openers whose continuation resolves into $\mathcal{R}$
5: $P^{(k)}_{\text{refuse}} \leftarrow P_R + P_{\text{open}}$
6: **end for**
7: **return** trajectory $\{P^{(k)}_{\text{refuse}}\}_{k=1}^{T}$ and its summaries (final, max, mean, onset step)

Algorithm 2 DIJA

Require: model p_θ, objective b, refined template τ_b with markers $\langle\text{mask}{:}N\rangle$, mask token $\mathbf{M}$
Ensure: model output y decoded inside the payload region
// Stage 1: prompt construction
1: build the last user turn from τ_b, replacing each marker $\langle\text{mask}{:}N\rangle$ with N copies of $\mathbf{M}$, which fixes the payload region $\mathcal{P}$
2: $x \leftarrow$ tokenized prompt with the masked region appended
// Stage 2: masked decoding
3: **while** $\mathbf{M} \in x$ **do**
4: $x_0 \leftarrow \arg\max\big(\text{Gumbel}(p_\theta(x), \tau)\big)$ ▷ full attention, no blocks
5: commit the single highest-confidence masked position; hold the rest
6: **end while**
7: **return** the decoded payload region

Algorithm 3 PAD

Require: model p_θ, objective b, affirmative prefill A, mask token $\mathbf{M}$
Ensure: model output y continuing the anchored opening

// Stage 1: anchor placement
$x \leftarrow$ prompt b with a masked response region appended
place the affirmative prefill A into the response region at evenly spaced positions, held fixed
// Stage 2: masked decoding
while $\mathbf{M} \in x$ **do**
  $x_0 \leftarrow \arg\max\big(\mathrm{Gumbel}(p_\theta(x), \tau)\big)$
  commit the single highest-confidence masked position; hold the rest
end while
return the decoded response region

Algorithm 4 Crescendo

Require: target p_θ, attacker A, objective b, turns N
Ensure: transcript whose final response addresses b

$\mathcal{C} \leftarrow \varnothing$
for $k = 1, \ldots, N$ **do**
  $u_k \leftarrow A.\textsc{Next}(b, \mathcal{C})$ ▷ escalate toward b; on a prior refusal, retreat and re-approach
  $\mathcal{C} \leftarrow \mathcal{C} \parallel u_k$; $y_k \leftarrow$ decode $p_\theta(\cdot \mid \mathcal{C})$; $\mathcal{C} \leftarrow \mathcal{C} \parallel y_k$
end for
return transcript $(u_{1:N}, y_{1:N})$

Algorithm 5 AMA

Require: target p_θ, attacker A, objective b, revisions K
Ensure: transcript whose final response addresses b

$(q, m) \leftarrow A.\textsc{Setup}(b)$ ▷ benign analogue q and a benign-to-target term map m
$y_1 \leftarrow$ decode $p_\theta(q)$; $y_2 \leftarrow$ decode p_θ(translate y_1 to another benign domain)
$s \leftarrow A.\textsc{Swap}(m)$ ▷ instantiate the benign structure onto b through m
for $i = 1, \ldots, K$ **do**
  $y \leftarrow$ decode $p_\theta(\cdot \mid y_1, y_2, s)$
  if y is compliant **then**
    return transcript
  end if
  $s \leftarrow A.\textsc{Revise}(s)$ ▷ attacker rewrites the swap
end for
return transcript

Algorithm 6 Paired multi-turn attack

Require: target M, primitive $A \in \{\text{DIJA}, \text{PAD}\}$, prime $(u_1, y_1), \ldots, (u_{N-1}, y_{N-1})$ from Crescendo or AMA, objective b
Ensure: responses $y_1, \ldots, y_N$, with y_N decoded under the infill

$\mathcal{C} \leftarrow \big((u_1, y_1), \ldots, (u_{N-1}, y_{N-1})\big)$ ▷ replay, M not re-queried
$(u_N, \mathcal{P}) \leftarrow A.\textsc{craft}(b)$ ▷ DIJA mask template or PAD affirmative prefill, payload $\mathcal{P}$
$y_N \leftarrow \textsc{Denoise}\big(M,\ \mathcal{C} \parallel u_N,\ \mathcal{P}\big)$
return $(y_1, \ldots, y_{N-1}, y_N)$

Algorithm 7 Ratchet

Require: target p_θ, attacker A, harm scorer H, objective b, variants M, window K
Ensure: responses $y_1, \ldots, y_M$ and a final response y_{M+1} addressing b

// Stage 1: variant collection
$(b_1, \ldots, b_M) \leftarrow A.\textsc{Variants}(b)$ ▷ same structure as b, different target
for $i = 1, \ldots, M$ **do**
  $y_i \leftarrow$ decode $p_\theta(\cdot \mid b_i)$ ▷ ordinary single-turn request
end for
// Stage 2: window selection
$\mathcal{W} \leftarrow$ the K pairs (b_i, y_i) with the highest $H(y_i)$
// Stage 3: masked infill against the window
$(u_{M+1}, \mathcal{P}) \leftarrow A.\textsc{MaskedTurn}(b)$ ▷ only turn that names b
$y_{M+1} \leftarrow \textsc{Denoise}\big(p_\theta,\ \mathcal{W} \,\|\, u_{M+1},\ \mathcal{P}\big)$
return $(y_1, \ldots, y_M, y_{M+1})$

B Attention to dLLM safety, by year

We counted arXiv entries in four categories by year, dated until early September, 2026.

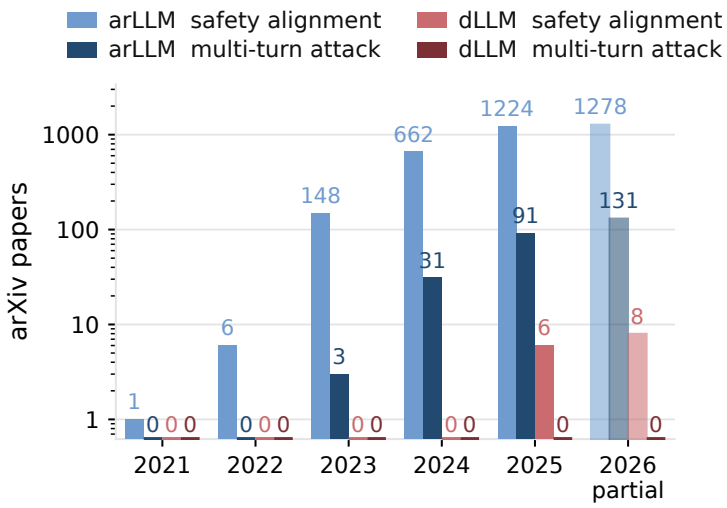


Figure 4: Literature attention to dLLM safety, by year. 2026 is partial.

C Refusal-onset token sets

The refusal mass of §4.1 sums over a frozen refusal-token set $\mathcal{R}$ and a set of openers. $\mathcal{R}$ holds the first tokens, with and without a leading space, of fourteen English refusal phrases such as “I’m sorry” and “I cannot”, dropping any id that is itself an opener, is not an alphabetic word of at least two characters, or also appears in the matching affirmative set. The openers are “I”, “I’m”, “As”, “To” and “The”, counted toward refusal only when the argmax at the next two positions hits a refusal cue such as “sorry”, “cannot”, “n’t” or “unable” before any affirmative cue.

D Extended related work

dLLMs. The lineage extends from discrete foundations, D3PM (Austin et al., 2021) and the continuous dLLMs (Li et al., 2022), through the score-based SEDD (Lou et al., 2024) and the masked MDLM (Sahoo et al., 2024), to Block Diffusion (Arriola et al., 2025), which interpolates arLLM

and dLLM decoding. At instruction-following scale the open models are LLaDA (Nie et al., 2025) with its LLaDA-1.5 (Zhu et al., 2026) and LLaDA-MoE (Zhu et al., 2025) variants, Dream (Ye et al., 2025), the multimodal MMaDA (Yang et al., 2025), and DiffusionGemma (Google DeepMind, 2026), with Mercury (Labs et al., 2025) and Seed Diffusion (Song et al., 2025) optimized for inference speed.

Table 3 maps the dLLM safety landscape on three dimensions, whether a work is **dLLM-specific**, whether it is **multi-turn-aware**, and whether it acts on the **denoising trajectory**, alongside the stage at which it operates and the level it reads.

Table 3: Comparison across the dLLM safety landscape. Stage is when a work acts, training or inference. Level is what it acts on or reads. ✓ full, ∼ partial, ✗ none.

Method	**Stage**	**Level**	dLLM-specific	Multi-turn	Trajectory
arLLM attacks (multi-turn)					
Crescendo (Russinovich et al., 2025)	inference	input	✗	✓	✗
Foot-in-the-Door (Weng et al., 2025)	inference	input	✗	✓	✗
Analogy (AMA) (Wu et al., 2025)	inference	input	✗	✓	✗
ActorAttack (Ren et al., 2025)	inference	input	✗	✓	✗
dLLM attacks					
DIJA (Wen et al., 2026)	inference	input	✓	✗	✗
PAD (Zhang et al., 2025)	inference	trajectory	✓	✗	✓
MaskForge (Ma et al., 2026)	inference	input	✓	✗	✗
TrajHijack (Singh, 2026)	inference	trajectory	✓	✗	✓
Context Nesting (He et al., 2026)	inference	input	✓	✗	✗
GPO-V (Pan et al., 2026)	inference	image	✓	✗	∼
SHADOWMASK (Cao et al., 2026)	training	weights	✓	✗	✗
dLLM alignment, steering and monitoring					
A2D (Jeung et al., 2026)	training	weights	✓	✗	∼
MOSA (Xie et al., 2026)	training	weights	✓	✗	∼
Recovery Alignment (Yamabe & Sakuma, 2026)	training	weights	✓	✗	∼
DiffuGuard (Li et al., 2026)	inference	trajectory	✓	✗	✓
SAD (Yusuf et al., 2026)	inference	trajectory	✓	✗	✓
Activation steering (Shnaidman et al., 2026)	inference	trajectory	✓	✗	✓
Adaptive steering and remasking (Lee & Han, 2026)	inference	trajectory	✓	✗	✓
D^2-Monitor (Liu et al., 2026)	inference	trajectory	✓	✗	✓
SRI Guard (Rahimi et al., 2026)	inference	trajectory	✓	✗	✓
Pre-decoding probe (Hazimeh et al., 2026)	inference	trajectory	✓	✗	✓

Other dLLM-specific attack surfaces. The same any-order masked generation opens attack surfaces with no arLLM analogue. For privacy, SAMA (Chen et al., 2026) runs membership inference by probing many mask configurations per sample, and infilling-based extraction (Wang & Asokan, 2026) conditions denoising on interior masks to recover far more verbatim training text than prefix conditioning. For provenance, Chainwash (Ameen et al., 2026) chains multi-style rewrites to strip dLLM watermarks. Gradient jailbreaks have also been retargeted onto the denoising objective (Neyroud & Corley, 2025).

E MODELS AND DATASETS

This section describes the models and datasets used.

E.1 dLLMs

LLaDA-8B. LLaDA-8B-Instruct is an 8B masked dLLMs (Nie et al., 2025). It generates by iteratively unmasking a fully masked sequence and commits tokens in confidence order, not left to right, yielding the any-order parallel decoding that the attacks and defenses studied here exploit. It is our primary dLLM target and is evaluated under all attack conditions and datasets.

LLaDA-1.5. LLaDA-1.5 is a preference-optimized version of LLaDA-8B, obtained by applying variance-reduced preference optimization to the base model (Zhu et al., 2026). We include it as an aligned counterpart to LLaDA-8B that isolates the effect of alignment while holding the architecture fixed.

Dream. Dream-v0-Instruct-7B is a 7B masked dLLM (Ye et al., 2025). Like LLaDA, it denoises a masked sequence in parallel. We include it as a second masked dLLM family to test whether the findings generalize beyond one model line.

MMaDA. MMaDA-8B-MixCoT is an 8B multimodal masked dLLM (Yang et al., 2025). It extends masked diffusion to joint text and image generation through a mixed chain-of-thought training recipe. We use its text generation pathway as an additional dLLM target.

E.2 arLLMs

Qwen2.5-32B-Instruct. Qwen2.5-32B-Instruct is a 32B arLLM instruction model (Qwen et al., 2025). It serves as our primary attacker model, generating each escalation turn of the multi-turn conversation.

Llama-3.1-8B-Instruct. Llama-3.1-8B-Instruct is an 8B arLLM instruction model (Grattafiori et al., 2024). We use it as an arLLM comparison target that contrasts the dLLM objective against left-to-right generation.

Qwen2.5-7B-Instruct. Qwen2.5-7B-Instruct is a 7B arLLM instruction model (Qwen et al., 2025). We use it as a second arLLM for comparison.

Mistral-7B-Instruct-v0.3. Mistral-7B-Instruct-v0.3 is a 7B arLLM instruction model (Jiang et al., 2023). We use it as a third arLLM for comparison.

E.3 Judge models

The judge is an open two-model ensemble scoring goal achievement. DeepSeek-R1-Distill-Qwen-32B, a 32B reasoning model distilled from DeepSeek-R1 (Guo et al., 2025), and OpenAI gpt-oss-20b, a 20B open-weight model (OpenAI et al., 2025), each vote by majority over three runs, and the two judges together determine the final label. When they disagree, the final label is the majority over all six runs.

E.4 Datasets

All are harmful-objective benchmarks used to evaluate attack success.

HarmBench. HarmBench (Mazeika et al., 2024) is a standardized red-teaming benchmark. We use its standard text objectives, which are direct harmful requests spanning several harm categories, as our primary attack-evaluation set.

AdvBench. AdvBench is the harmful-objectives set introduced with the GCG attack (Zou et al., 2023). The released file contains short imperative requests for disallowed content, which we use as an additional attack-evaluation set.

StrongREJECT. StrongREJECT (Souly et al., 2024) is a set of forbidden prompts designed to separate a jailbreak that elicits genuinely harmful content from one that elicits an empty compliance.

JailbreakBench. JailbreakBench (Chao et al., 2024a) provides the JBB-Behaviors a set of harmful objectives across ten categories, each paired with a benign counterpart.

CyberSecEval. CyberSecEval (Bhatt et al., 2023) is a cybersecurity-safety benchmark. We add it as a technical harm domain dataset that the other benchmarks cover only sparsely.

PAP. PAP, the persuasive adversarial prompts of Zeng et al. (2024), reformulates harmful requests using a taxonomy of persuasion techniques in place of adversarial suffixes.