JUMP: Single-Pass Membership Inference on Fine-Tuned Diffusion Language Models

Yeachan Jun [1] **Albert No** [1]

Abstract

Membership inference attacks (MIAs) test whether a candidate example appeared in a model's training data. We study MIAs for fine-tuned discrete diffusion language models (dLLMs), where membership means inclusion in the target model's fine-tuning set. Unlike autoregressive language models, dLLMs allow an attacker to choose arbitrary mask sets and obtain token distributions for all masked positions in parallel. The prior dLLM attack, SAMA, follows a natural loss-mimicking strategy by averaging reconstruction signals over many randomly sampled masks, but it uses the any-order interface only as randomization and requires many target/reference queries. We propose JUMP (*Joint Uncertainty-Guided Mask Probing*), a single-pass scoring attack that exploits both distinctive properties of dLLMs: any-order decodability is used to select low-reference-confidence positions, and parallel decodability is used to score all selected positions through one joint masked query per model. JUMP masks the selected positions jointly and computes a clipped target/reference reconstruction-gap statistic. On fine-tuned LLaDA-8B-Base across six MIMIR domains, JUMP improves mean ROC-AUC from 0.82 to 0.90 over SAMA and substantially improves low-FPR detection, while requiring only one selector pass and one scoring pass through each of the target and reference models.

1. Introduction

Membership inference attacks (MIAs) are a standard tool for auditing privacy leakage in machine learning models (Shokri et al., 2017; Yeom et al., 2018). Given a trained model and a candidate example, the attacker decides whether the example was included in the model's training data. Successful attacks reveal that the model's outputs retain measurable traces of particular training examples. We study this problem for fine-tuned language models: the target model is adapted on a private corpus, and membership means inclusion in that fine-tuning set.

For autoregressive (AR) language models, MIAs naturally follow the left-to-right likelihood interface. A sequence is scored through the same prefix-conditioned predictions used in training, and stronger attacks refine this statistic with reference calibration, token filtering, perturbation-based comparisons, or information-theoretic normalization (Carlini et al., 2021; Duan et al., 2024; Shi et al., 2024b; Zhang et al., 2025; Xie et al., 2024; Mattern et al., 2023; Chang et al., 2025; Tao and Shokri, 2026). However, the conditional contexts are fixed: token x_i is always evaluated from the prefix $x_{<i}$. Thus, AR MIAs largely differ in how they aggregate a predetermined set of token scores.

Discrete diffusion language models (dLLMs) change this interface. Recent dLLMs such as LLaDA and Dream reconstruct masked tokens from bidirectional visible context rather than from a prefix alone (Nie et al., 2025; Ye et al., 2025). For MIAs, the key difference is that the attacker can choose the mask set. The same sequence can induce many reconstruction tasks depending on which tokens are hidden; we call this *any-order decodability*. At the same time, the model returns token distributions for all masked positions in one forward pass; we call this *parallel decodability*. Combined, these properties make the mask set itself a central component of the attack design.

SAMA (Chen et al., 2026), the first dLLM-specific MIA, follows a direct loss-mimicking strategy by sampling many random mask sets and averaging target/reference reconstruction gaps. This is a natural estimator of the diffusion reconstruction objective, but it uses the any-order interface mainly as randomization. Random subsets often include easy or uninformative tokens, diluting the membership signal, and the target/reference query cost grows with the number of sampled masks. This motivates our central question: *can an attack use the dLLM's any-order interface to choose a more informative mask set, and then use parallel decoding to evaluate that set in a single pass?*

We answer this question with JUMP (*Joint Uncertainty-*

[1]Department of Artificial Intelligence, Yonsei University, Seoul, Korea. Correspondence to: Albert No <albertno@yonsei.ac.kr>.

Published as a paper at the 1st FoGen workshop, ICML 2026, Seoul, South Korea, 2026.

Guided Mask Probing), a single-pass MIA for fine-tuned dLLMs. JUMP first uses the reference model to find positions where it assigns low confidence to the true token. It then masks these positions together and compares how the target and reference models reconstruct the true tokens. After the mask set is chosen, one masked query to the target model and one masked query to the reference model returns all selected token scores in parallel. Thus, JUMP makes mask-set selection the main attack design problem while keeping the final target/reference scoring cost constant.

We evaluate JUMP on fine-tuned LLaDA-8B-Base (Nie et al., 2025) across six MIMIR (Duan et al., 2024) domains. Compared with SAMA, JUMP raises mean ROC-AUC from 0.82 to 0.90 and improves detection in the strict low-FPR regime. At the same time, JUMP replaces many random target/reference masked queries with a single selected target/reference scoring query per sample. These results support the central premise of JUMP: for dLLMs, membership inference should be designed around a single informative joint mask rather than many random masks.

2. Related Work

Membership inference attacks. Given a target model M_{tgt} trained on $\mathcal{D}_{\text{train}}$ and a candidate example x, an MIA asks whether $x \in \mathcal{D}_{\text{train}}$ (Shokri et al., 2017; Yeom et al., 2018). We write a score-based MIA as a scalar statistic

$$g(x;\, M_{\text{tgt}},\, M_{\text{ref}}) \in \mathbb{R},$$

where M_{ref} may be absent for target-only attacks. The attacker predicts membership when $g(x; M_{\text{tgt}}, M_{\text{ref}}) > \eta$, and varying η yields the ROC curve. A classical example is the negative-loss statistic $g_{\text{loss}}(x; M_{\text{tgt}}) = -\ell(M_{\text{tgt}}, x)$, which connects membership leakage to the generalization gap (Yeom et al., 2018). Reference-based attacks instead compare the target with a reference, e.g., $g_{\text{ref}}(x) = -\ell(M_{\text{tgt}}, x) + \ell(M_{\text{ref}}, x)$, to reduce the confounding effect of intrinsic example difficulty (Watson et al., 2022; Carlini et al., 2022; Zarifzadeh et al., 2024). Complementary work studies training-data extraction and privacy auditing (Long et al., 2018; Carlini et al., 2019; 2021; 2023; Nasr et al., 2025; Steinke et al., 2023; Jagielski et al., 2023; Lukas et al., 2023; Mireshghallah et al., 2022).

MIA for autoregressive language models. For an autoregressive language model, the loss is a natural MIA statistic because the model is trained by next-token negative log-likelihood, yielding a sequence score from prefix-conditioned token probabilities:

$$g_{\text{AR}}(x) = \log p_{\text{AR}}(x) = \sum_{i=1}^{L} \log p(x_i \mid x_{<i}).$$

Reference-calibrated variants use $g_{\text{AR-ref}}(x) = \log p_{M_{\text{tgt}}}(x) - \log p_{M_{\text{ref}}}(x)$, while token-selection methods such as Min-K% and Min-K%++ compute g from low-probability tokens rather than all tokens (Shi et al., 2024b; Zhang et al., 2025). Other attacks use contextual perturbations, neighborhood comparisons, or information-theoretic calibrations to refine the same score-based decision rule (Xie et al., 2024; Mattern et al., 2023; Chang et al., 2025; Tao and Shokri, 2026). These methods differ in how they construct g, but the underlying conditional contexts are fixed by autoregressive decoding.

Discrete diffusion language models. Masked diffusion language models (Austin et al., 2021; Hoogeboom et al., 2021; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024a; Ou et al., 2025), scaled by recent dLLMs such as LLaDA and Dream (Nie et al., 2025; Ye et al., 2025), define reconstruction distributions over arbitrary masked subsets. Let $x = (x_1, \ldots, x_L)$ and let $x_{\setminus S}$ denote the sequence where positions in $S \subseteq \{1, \ldots, L\}$ are replaced by `[MASK]`. A dLLM models $p_\theta(x_i \mid x_{\setminus S})$ for $i \in S$ and is trained with a masked reconstruction objective of the form

$$\mathcal{L}(\theta) = -\,\mathbb{E}_{x,\lambda,S}\left[\frac{1}{|S|}\sum_{i\in S}\log p_\theta(x_i \mid x_{\setminus S})\right], \qquad \lambda \sim U(0,1).$$

The attacker-facing consequence is that S is not fixed: any-order decodability lets the attacker choose the reconstruction task, and parallel decodability returns all $|S|$ masked-token distributions in one forward pass (Uria et al., 2014; Germain et al., 2015; Yang et al., 2019; Hoogeboom et al., 2022; Ghazvininejad et al., 2019; Chang et al., 2022).

MIA for diffusion language models. SAMA (Chen et al., 2026) is the first dLLM-specific MIA and is a natural loss-mimicking baseline. It samples T random mask sets $S_1, \ldots, S_T$ and computes a reconstruction-gap statistic such as

$$g_{\text{SAMA}}(x) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|S_t|}\sum_{i\in S_t}\Big[\log p_{M_{\text{tgt}}}(x_i \mid x_{\setminus S_t}) - \log p_{M_{\text{ref}}}(x_i \mid x_{\setminus S_t})\Big]. \tag{1}$$

This statistic mirrors the dLLM reconstruction objective by averaging over randomly sampled masked subsets, and is therefore a natural first approach. However, computing Eq. (1) requires one target and one reference forward pass for each sampled subset, for a total cost of $2T$ forward evaluations per example. Our method keeps the reconstruction-gap principle but replaces random multimask averaging with a single selected mask set.

3. Threat Model

Setting. The target model M_{tgt} is a fine-tuned dLLM with the masked-reconstruction interface $p_{M_{\text{tgt}}}(x_i|x_{\setminus S})$. It is fine-tuned on a private dataset $\mathcal{D}_{\text{train}}$. The attacker receives a candidate sequence $x = (x_1, \ldots, x_L)$ and does not know whether $x \in \mathcal{D}_{\text{train}}$. The attacker also has access to a compatible reference model M_{ref}; in our experiments, this is the same base dLLM before fine-tuning.

Access. Our default setup gives the attacker score access to both models. For any mask set $S \subseteq \{1, \ldots, L\}$ chosen by the attacker, the attacker may submit the masked sequence $x_{\setminus S}$ to either model and obtain token log-probabilities at position $i \in S$:

$$\Big\{\log p_M(x_i \mid x_{\setminus S}) : i \in S\Big\}, \qquad M \in \{M_{\text{tgt}}, M_{\text{ref}}\}. \tag{2}$$

The attacker has white-box access to the reference model and may lightly train auxiliary reference-side components on generic public text. This auxiliary training uses neither member/non-member labels nor target-model weights. We also consider a stricter black-box selector setting later, in which position selection is performed without training an auxiliary component.

Derived quantities. The observable quantities in Eq. (2) are the model log-probabilities. A useful derived statistic for a chosen mask set S is the token-level target/reference reconstruction gap

$$\Delta_i(S) = \log p_{M_{\text{tgt}}}(x_i \mid x_{\setminus S}) - \log p_{M_{\text{ref}}}(x_i \mid x_{\setminus S}), \quad i \in S. \tag{3}$$

All gaps $\{\Delta_i(S) : i \in S\}$ are obtained with two model forward passes, one through M_{tgt} and one through M_{ref}, independent of $|S|$.

Goal and evaluation. An attack specifies one or more mask sets and computes a scalar MIA statistic from the resulting observations. Let $\mathcal{Q} = \{S_1, \ldots, S_m\}$ be the query collection, fixed, randomized, or chosen using x and the reference model. A general score has the form

$$g(x; M_{\text{tgt}}, M_{\text{ref}}) = G\Big(\{\log p_M(x_i \mid x_{\setminus S}) : S \in \mathcal{Q},\ i \in S,\ M \in \{M_{\text{tgt}}, M_{\text{ref}}\}\}\Big),$$

where choosing $\mathcal{Q}$ is part of the attack design. The attacker predicts membership by thresholding g. We evaluate attacks using ROC-AUC and TPR at FPR $\in \{10\%, 1\%, 0.1\%\}$, following standard MIA practice (Watson et al., 2022; Carlini et al., 2022). We also report the number of forward evaluations (NFE) per sample. Unless otherwise specified, NFE counts only target/reference passes used to compute the MIA statistic; selector passes are reported separately when they are not part of the final score.

4. Method: JUMP (*Joint Uncertainty-Guided Mask Probing*)

We first present a diagnostic experiment showing where membership signal appears in a dLLM (Section 4.1), then define the JUMP scoring rule (Section 4.2), and finally describe how we select difficult positions efficiently (Section 4.4).

4.1. Low-confidence tokens and membership signal

The any-order interface in dLLM raises a basic design question: which token positions should an attacker probe? We first study the ideal one-hole reference low-confidence score

$$q^\star(i \mid x) = \log p_{M_{\text{ref}}}(x_i \mid x_{\setminus\{i\}}), \tag{4}$$

where smaller values indicate lower reference confidence in the true token under bidirectional context. This quantity is useful for analysis because it directly measures how uncertain the reference dLLM is about the true token x_i when only that token is hidden.

To test whether this low-confidence score is related to membership signal, we run a diagnostic one-hole experiment on LLaDA-8B-Base fine-tuned on the Wikipedia (en) domain. For each candidate sequence, we use a 512-token evaluation window and, for every valid position i, mask only that position, i.e., $S = \{i\}$. We then record the one-hole reference probability $p_{M_{\text{ref}}}(x_i|x_{\setminus\{i\}})$ and the target/reference gap $\Delta_i(\{i\})$ from Eq. (3). Positions are ranked by Eq. (4) and partitioned into three groups of size $K = 64$: lowest-confidence positions, uniformly random positions, and highest-confidence positions.

Figure 2 shows that low-confidence positions carry substantially stronger membership signal. Under one-hole masking, the fine-tuned target reconstructs these positions much better on member sequences than on non-member sequences, producing a mean member/non-member gap difference of $+0.174$. Random and high-confidence positions show much weaker separation. This supports the use of reference-model confidence as a localization prior: positions where the reference model has low confidence are more informative for fine-tuning membership.

The diagnostic procedure itself is not an efficient attack. Ranking all positions by Eq. (4) requires one masked query per position to the reference model, and computing the corresponding target/reference gaps requires one masked query per position and per model. Thus exact one-hole localization and scoring costs $2L$ target/reference forward evaluations for a length-L sequence, in addition to the localization work. The goal of JUMP is to approximate the same localization principle without paying this cost: select a small set of low-confidence positions once, then exploit parallel decoding to score all of them through a single joint mask.

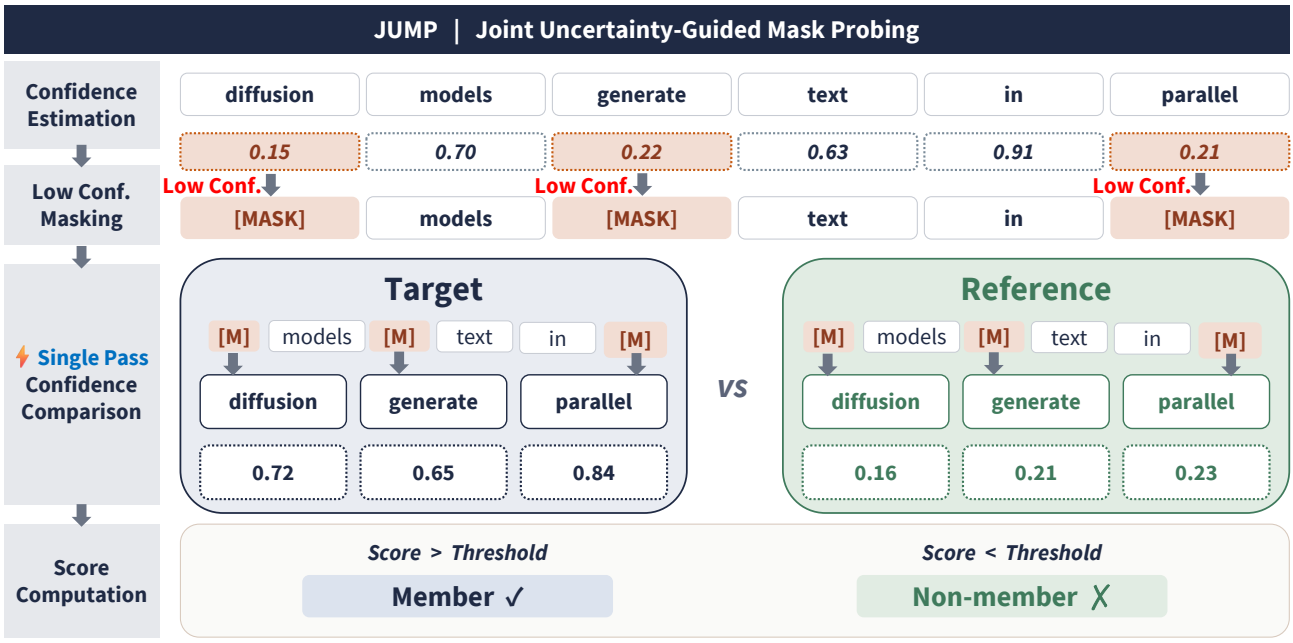


Figure 1. Overview of JUMP. The attack first selects informative positions under the reference model, then masks the selected positions jointly and scores them with the target and reference dLLMs. The final MIA statistic is computed from clipped token-level target/reference reconstruction gaps. In the visualization, general tokens are indicated by solid-line boxes, while confidence scores are denoted by dashed-line boxes.

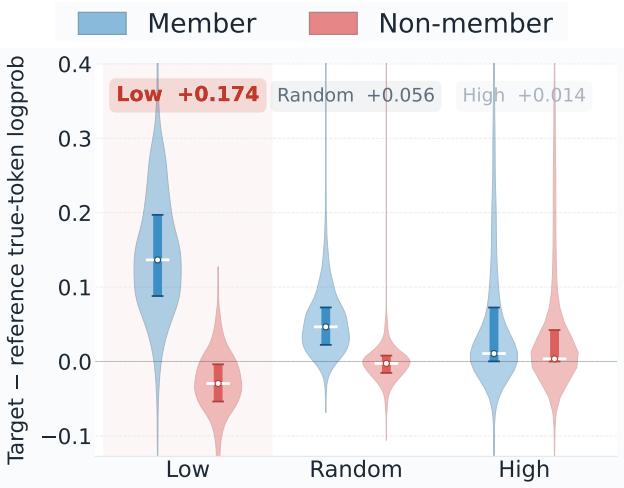


Figure 2. Distribution of the one-hole reconstruction gap $\Delta_i(\{i\})$ across reference-model confidence tiers. Low-confidence positions produce substantially stronger member/non-member separation than random or high-confidence positions.

4.2. JUMP scoring rule

Given a sequence x of length L, let $[L] \subseteq \{1, \ldots, L\}$ denote the valid positions that may be masked and scored. JUMP uses a selector score $q_\phi(i \mid x)$ for each $i \in [L]$. The score $q_\phi(i \mid x)$ is a scalar prediction of reference-model low confidence for the true token at position i; it is not a vocabulary distribution. Ideally, $q_\phi(i \mid x)$ approximates the one-hole low-confidence score $q^\star(i \mid x)$ in Eq. (4), so smaller values indicate lower reference-model reconstruction confidence.

For a probing budget K, JUMP selects the K positions of lowest confidences:

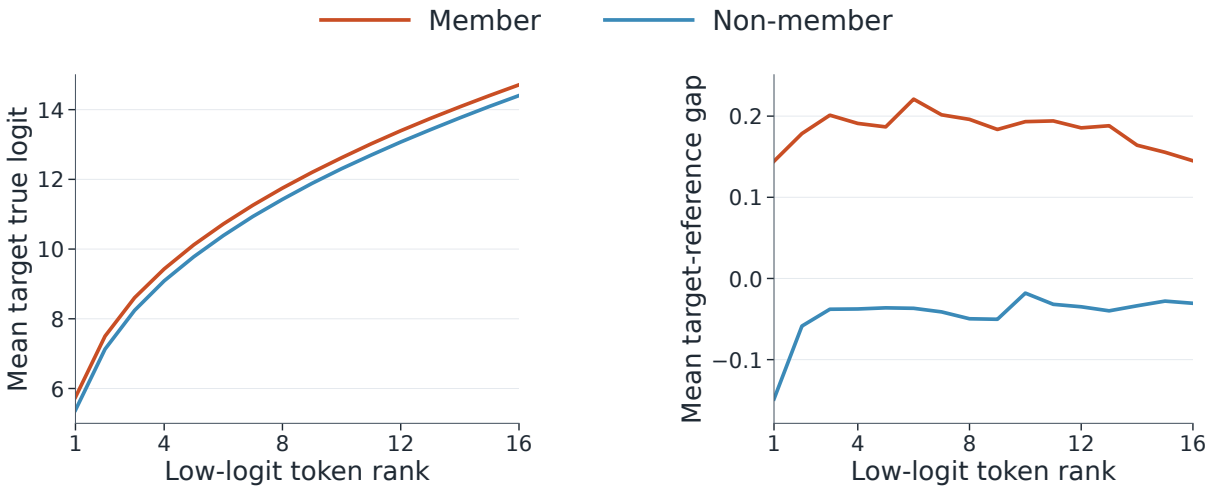
$$\mathcal{H}_K(x; \phi) = \big\{ i \in [L] \mid q_\phi(i \mid x) \text{ is one of the lowest } K \text{ values among all } i \in [L] \big\}. \tag{5}$$

In the main experiments, we use a 512-token window and set $K = 64$; this choice is examined in Section 6.2 and Appendix E.1. The selected positions are masked jointly, $S = \mathcal{H}_K(x; \phi)$, and the attacker computes $\Delta_i(S)$ for all $i \in S$ using Eq. (3). Since all positions in S are reconstructed in parallel, this joint probe returns K token-level membership signals with one target and one reference query. Thus, K affects only the number of averaged token gaps, not the target/reference scoring NFE.

The final statistic clips and averages the selected token gaps:

$$\begin{aligned} H &= \mathcal{H}_K(x; \phi), \\ g_{\text{JUMP}}(x; M_{\text{tgt}}, M_{\text{ref}}) &= \frac{1}{K} \sum_{i \in H} \text{clip}(\Delta_i(H), -\tau, \tau), \\ \tau &= \log 1.5. \end{aligned} \tag{6}$$

We clip to bound the effect of any single token, limiting heavy-tailed token gaps that can otherwise dominate the mean and create false positives at low FPR. The threshold $\tau = \log 1.5$ caps each token's contribution at a 1.5:1



(a) Target-Only Readout

(b) Reference-Calibrated Readout

Figure 3. **Why reference calibration is needed.** (a) Target-only true-token log-probabilities on the selected low-confidence positions. (b) Target–reference reconstruction gaps on the same positions. Absolute target scores remain partly confounded by intrinsic token difficulty, whereas the calibrated gap subtracts generic predictability and yields a clearer member/non-member separation.

probability ratio; we ablate this choice in Section 6.3 and Appendix E.3.

Remark (NFE and Efficiency). The scoring stage is single-pass: one target reconstruction and one reference reconstruction produce all selected token gaps, so the target/reference scoring NFE is $\mathcal{O}(1)$ in K, L, and sequence length. SAMA instead needs one target/reference pair per sampled mask subset, while exact one-hole localization scales linearly with the number of positions. The PRISM selector is trained once offline on public text and adds no target-model cost at test time.

4.3. Why Reference Calibration Is Necessary

Hard-position selection identifies where membership signal is likely to appear, but it does not by itself separate memorization from intrinsic token difficulty. Because JUMP intentionally probes positions that are hard under the reference view, absolute target scores can be ambiguous: low scores may reflect generally hard tokens, while high scores may reflect generic predictability rather than memorization. JUMP therefore uses a reference-calibrated readout: as shown in Figure 3, the target-only readout provides only modest member/non-member separation, whereas the target–reference reconstruction gap subtracts the base model's estimate of generic predictability and leaves a clearer example-specific reconstruction advantage for member sequences. Thus, hard-position selection and reference calibration play complementary roles: the selector determines which positions to probe, and the target–reference gap turns those probes into a difficulty-calibrated membership statistic for distinguishing fine-tuning memorization from generalization already present in the base model.

4.4. Generic token selector

The remaining question is how to obtain q_ϕ without running the expensive one-hole diagnostic in Section 4.1. In our default white-box reference setting, we use PRISM (Kim et al., 2025) as a generic low-confidence token selector. Given the unmasked sequence, PRISM predicts a position-wise low-confidence score from the reference-model representations:

$$q_\phi(i \mid x) \approx q^\star(i \mid x) = \log p_{M_{\text{ref}}}(x_i \mid x_{\setminus\{i\}}).$$

PRISM is a lightweight per-token quality head attached to the reference MDM. It is fine-tuned on text drawn from the reference model's pretraining distribution using a binary cross-entropy objective: for a partially masked sequence, a held-out position i is filled in by a sample from the reference MDM's one-hole posterior (with stop-gradient on the unmasking head), and $q_\phi(i \mid x)$ is trained against the binary label indicating whether the sampled token matches the ground truth at position i. Under this objective, the optimum provably recovers the reference model's per-token quality at position i given the surrounding context (Kim et al., 2025). We provide the PRISM training details in Appendix A.3. The quality head is fit only from text in the reference distribution, without any member/non-member labels or target-model information. At attack time, the unmasking probabilities and the PRISM scores $q_\phi(i \mid x)$ are produced in a *single* reference-model forward pass. JUMP then selects the set in Eq. (5) and performs the two target/reference scoring passes in Eq. (6).

Table 1. Per-domain results on fine-tuned LLaDA-8B-Base. Attack quality: ROC-AUC and TPR at fixed FPR (↑); attack cost: NFE per sample (↓). NFE is reported as target/reference scoring passes plus selector passes when applicable. Best in each column in **bold**.

Method	ArXiv					GitHub					HackerNews				
	Attack quality (↑)				Cost (↓)	Attack quality (↑)				Cost (↓)	Attack quality (↑)				Cost (↓)
	AUC	T@10	T@1	T@0.1	NFE	AUC	T@10	T@1	T@0.1	NFE	AUC	T@10	T@1	T@0.1	NFE
Loss	0.53	0.12	0.01	0.00	1	0.59	0.19	0.04	0.01	1	0.52	0.11	0.01	0.01	1
ZLIB	0.53	0.13	0.01	0.00	1	0.61	0.22	0.06	0.02	1	0.52	0.10	0.01	0.01	1
SAMA	0.82	0.54	0.28	0.07	$2T$	0.77	0.43	0.14	0.07	$2T$	0.71	0.34	0.05	0.01	$2T$
JUMP	**0.94**	**0.84**	**0.54**	**0.24**	$2{+}1$	**0.86**	**0.67**	**0.33**	**0.12**	$2{+}1$	**0.77**	**0.40**	**0.11**	**0.03**	$2{+}1$

Method	PubMed Central					Wikipedia (en)					Pile-CC				
	Attack quality (↑)				Cost (↓)	Attack quality (↑)				Cost (↓)	Attack quality (↑)				Cost (↓)
	AUC	T@10	T@1	T@0.1	NFE	AUC	T@10	T@1	T@0.1	NFE	AUC	T@10	T@1	T@0.1	NFE
Loss	0.53	0.13	0.02	0.00	1	0.52	0.10	0.01	0.00	1	0.52	0.12	0.01	0.00	1
ZLIB	0.53	0.14	0.01	0.00	1	0.52	0.10	0.01	0.00	1	0.52	0.13	0.02	0.00	1
SAMA	0.81	0.51	0.23	0.03	$2T$	0.91	0.73	0.47	0.36	$2T$	0.90	0.70	0.40	**0.20**	$2T$
JUMP	**0.92**	**0.75**	**0.44**	**0.17**	$2{+}1$	**0.98**	**0.95**	**0.80**	**0.68**	$2{+}1$	**0.96**	**0.90**	**0.56**	0.17	$2{+}1$

Black-box selector. We also evaluate a black-box selector, denoted PRISM-Free, that does not train an auxiliary head. PRISM-Free uses the reference model's clean-text true-token score as a proxy for Eq. (4), selecting positions with large

$$q_{\text{free}}(i \mid x) = \log p_{M_{\text{ref}}}(x_i \mid x),$$

where the input sequence x is completely unmasked. Theoretically, this score should be meaningless: dLLMs are trained exclusively to predict ground-truth tokens from [MASK] tokens, meaning the output distribution over an already-visible token x_i has no mathematical guarantee. Surprisingly, we empirically found that for LLaDA, evaluating the model on the clean sequence yields a reasonable "one-hole confidence" score, behaving remarkably as if x_i had actually been masked. Because this correlation is an empirical artifact of LLaDA rather than a guarantee for general dLLMs, whereas PRISM is explicitly trained to predict the parallel low-confidence score in Eq. (4), we use PRISM as the default selector and report PRISM-Free as a stricter-access variant.

5. Experiments

5.1. Experimental setup

Models. We evaluate LLaDA-8B-Base (Nie et al., 2025). For each domain, the target model M_{tgt} is obtained by fine-tuning LLaDA-8B-Base on that domain's member split. The reference model M_{ref} is the corresponding checkpoint before fine-tuning. We additionally report secondary Dream-v0-7B-Base results in Appendix C.

Data. We use six domains from MIMIR (Duan et al., 2024): ArXiv, GitHub, HackerNews, Pile-CC, PubMed Central, and Wikipedia (en). Each evaluation set contains 1,000 member examples and 1,000 non-member examples from the same domain. Members are the exact examples used for fine-tuning, and non-members are held-out examples.

Baselines. We compare against three baselines. **Loss** uses the target model's average sequence log-probability as a target-only statistic (Yeom et al., 2018). **ZLIB** normalizes the loss statistic by zlib-compressed length (Carlini et al., 2021). **SAMA** is the prior dLLM-specific MIA and averages reconstruction statistics over randomly sampled mask subsets (Chen et al., 2026). Loss and ZLIB are included as sanity checks for likelihood-style attacks; SAMA is the primary dLLM baseline.

Metrics and configuration. We report ROC-AUC and TPR at FPR $\in \{10\%, 1\%, 0.1\%\}$ (Carlini et al., 2022), along with NFE per sample (separating target/reference scoring from selector passes). Unless stated otherwise, JUMP uses a PRISM selector trained on C4 (Raffel et al., 2020), a 512-token window, $K = 64$ selected positions, clipping threshold $\tau = \log 1.5$, and batch size 8. K affects the number of aggregated token-level signals but not the target/reference scoring NFE. See Appendix A for additional details.

5.2. Performance and efficiency across domains

Table 1 reports per-domain LLaDA results, with 95% bootstrap confidence intervals in Appendix B; Dream results are in Appendix C. JUMP improves ROC-AUC over SAMA in all six domains and gives larger gains in the strict low-FPR regime. It also uses only two target/reference scoring passes per sample, plus one reference-side selector pass, whereas SAMA needs a separate target/reference pair for every random mask subset. Loss and ZLIB remain close to chance, consistent with prior evidence that uncalibrated sequence likelihood is weak for LLM-scale membership inference (Duan et al., 2024).

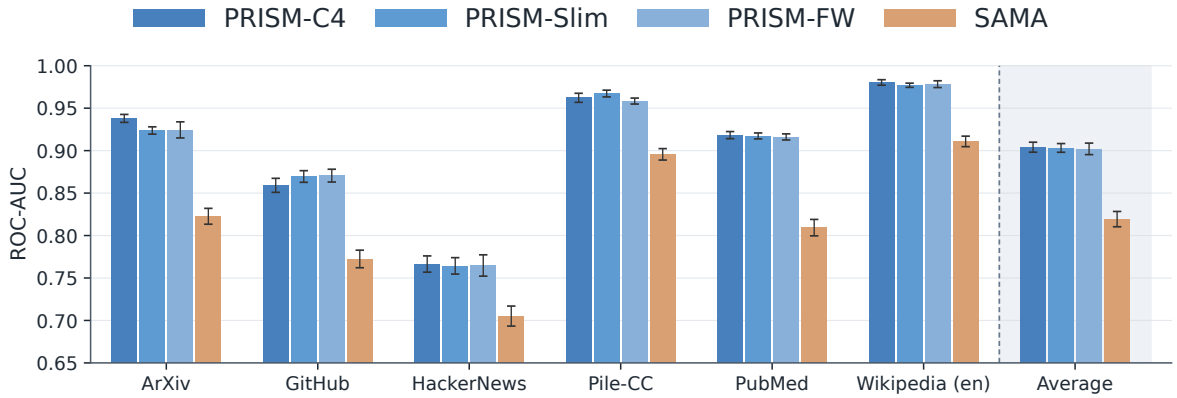


Figure 4. **Selector training corpus ablation.** ROC-AUC across MIMIR domains for PRISM selectors trained on different generic corpora. Learned selectors cluster within a narrow band, indicating that corpus identity is a second-order effect.

Table 2. NFE comparison across MIA strategies. L is sequence length and T is the number of SAMA random mask subsets. Scoring NFE counts target/reference passes used to compute the final statistic. For JUMP, the parenthesized term denotes the additional reference-model pass used by the PRISM selector.

Model	Strategy / Access	Context	Scoring NFE
AR	Target / Reference loss	left-to-right	$2L$
dLLM	One-hole probing	bidirectional	$2L$
dLLM	SAMA	bidirectional	$2T$
dLLM	JUMP (K selected tokens)	bidirectional	$2\,(+1)$

5.3. NFE analysis

Table 2 compares representative query costs, using L for sequence length and T for the number of SAMA mask subsets. Under a per-position score-access accounting, AR target/reference loss and exact dLLM one-hole probing both require $2L$ passes. SAMA reduces the number of reconstruction tasks to T random subsets but still costs $2T$ target/reference passes. In contrast, JUMP scores K selected positions through one joint mask, so the target/reference statistic costs exactly two scoring passes independent of K, L, and T; the default PRISM selector adds one reference pass for position selection.

6. Ablation Study

We ablate the main components of JUMP: selector training corpus, selected-token budget, clipping, and the training-free PRISM-Free selector.

6.1. Selector training corpus

The first ablation tests whether the attack relies on the PRISM selector's training corpus. Three separate PRISM heads are trained on C4 (Raffel et al., 2020), SlimPajama (Soboleva et al., 2023), and FineWeb (Penedo et al., 2024), and evaluated using the standard JUMP pipeline.

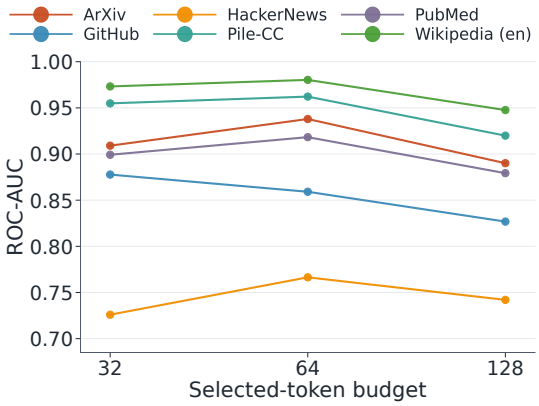


Figure 5. **Selected-token budget ablation.** ROC-AUC remains stable around the default $K = 64$.

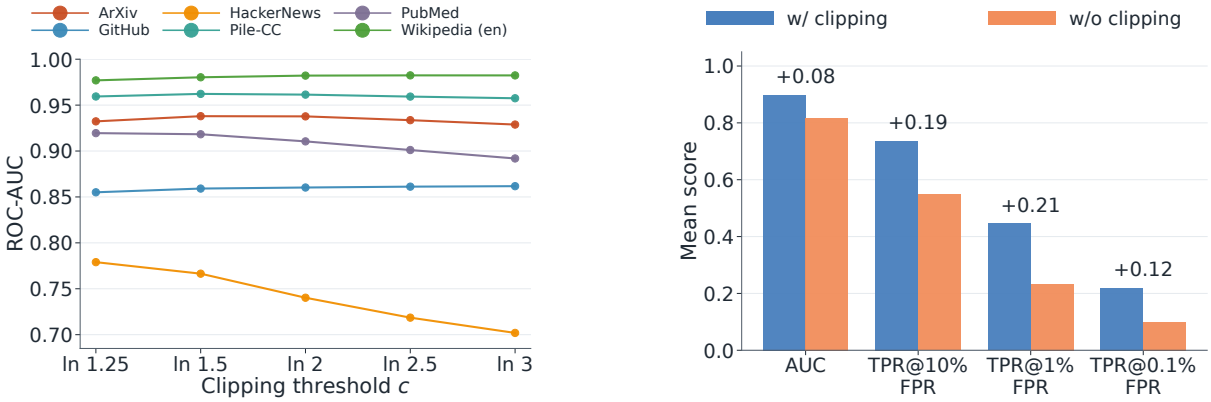
The learned selectors exhibit nearly identical performance across domains. PRISM-C4 is marginally best, but the small differences suggest that PRISM captures generic reconstruction difficulty rather than corpus-specific artifacts.

6.2. Selected-token budget

We next vary the selected-token budget K. Because selected positions are reconstructed in parallel, K changes only the number of aggregated token gaps, not the target/reference scoring NFE. *Figure* 5 shows that JUMP is robust: too few positions weaken the aggregate signal, while increasing beyond $K = 64$ adds less-informative positions without consistent gains. Full details are in Appendix E.1.

6.3. Clipping and threshold sensitivity

We next examine clipping. *Figure* 6a shows that performance is stable over moderate thresholds; we use $c = \log 1.5$ because the full sweep in Appendix E.3 gives the best aggregate low-FPR operating point. *Figure* 6b shows that clipping improves all metrics by suppressing isolated extreme token gaps and favoring consistent target–reference advantages.



(a) Threshold sweep

(b) Effect of clipping

Figure 6. **Clipping ablation.** (a) Sweeping the clipping threshold constant c shows that JUMP is not overly sensitive around the default setting. (b) Clipping improves detection performance and robustness, with the largest gains in the strict low-FPR regime.

6.4. PRISM-Free

Table 3. PRISM-Free per-domain results. Metrics and formatting follow Table 1. Because PRISM-Free requires no training, NFE reflects only target/reference scoring passes.

Domain	AUC	TPR@10%	TPR@1%	TPR@0.1%
ArXiv	0.89	0.67	0.32	0.12
GitHub	0.86	0.65	0.22	0.13
HackerNews	0.76	0.36	0.07	0.01
Pile-CC	0.94	0.83	0.51	0.18
PubMed	0.82	0.54	0.17	0.01
Wikipedia (en)	0.96	0.89	0.63	0.36
Mean	0.87	0.66	0.32	0.14

PRISM-Free replaces the learned PRISM head with the reference model's clean-text true-token score and requires no auxiliary training. It remains effective, reaching mean ROC-AUC 0.87 and outperforming SAMA (0.82), although PRISM-C4 is more stable at extreme FPRs (*Table* 3).

7. Conclusion

JUMP leverages any-order decodability to identify vulnerable positions and parallel decodability to evaluate all probes simultaneously. Across fine-tuned LLaDA and Dream models, this yields a membership inference attack that is both stronger and more computationally efficient than SAMA. Our DP-LoRA results (Appendix G) further show that parameter-efficient fine-tuning with differential privacy can mitigate uncertainty-guided membership inference (Abadi et al., 2016; Yu et al., 2022; Li et al., 2022). Ultimately, JUMP establishes a rigorous baseline for future dLLM privacy evaluations.

Limitations. Our evaluation focuses on fine-tuned dLLMs to isolate architectural vulnerabilities from the ambiguities of pretraining-scale membership. Extending the analysis to pretrained models remains important. In addition, our primary attack uses a matched reference model, although PRISM-Free shows that effective single-pass probing does not strictly require target internals.

Implications for dLLM auditing. Our results suggest that the mask set is not merely an implementation detail in dLLM privacy auditing. Because the same sequence can be queried through many conditional reconstruction tasks, different masks can expose different degrees of membership signal. This helps explain why averaging over random masks, as in SAMA, can dilute the signal, while a single carefully chosen joint mask can reveal a stronger fine-tuning trace. More broadly, JUMP suggests that privacy leakage in dLLMs is partly localized: it may concentrate on positions where the base model is uncertain but the fine-tuned model has acquired a reconstruction advantage. Rather than treating a sequence as uniformly memorized or non-memorized, this view suggests that fine-tuning may leave sparse, position-level traces that become visible only under the right reconstruction query and mask context, rather than under average loss alone.

Broader Impacts. This work has both positive and negative societal implications. JUMP provides an efficient tool for auditing membership leakage in fine-tuned diffusion language models, helping developers identify privacy risks before deployment. However, stronger MIAs are dual-use and could be misapplied to test whether sensitive text was included in a private fine-tuning corpus. To reduce this risk, we frame JUMP as an auditing method, evaluate it on benchmark datasets, and include DP-LoRA as a concrete mitigation with formal (ϵ, δ)-DP guarantees. We do not release private fine-tuning data or target checkpoints; reproducibility is supported through code, public assets, and detailed configurations.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM CCS*, 2016.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, 2021.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security*, 2019.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *USENIX Security*, 2021.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy (S&P)*, 2022.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *ICLR*, 2023.

Hongyan Chang, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Reza Shokri. Context-aware membership inference attacks against pre-trained large language models. In *EMNLP*, 2025.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *CVPR*, 2022.

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*, 2026.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? In *COLM*, 2024.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *ICML*, 2015.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP-IJCNLP*, 2019.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *NeurIPS*, 2021.

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *ICLR*, 2022.

Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Guha Thakurta, Nicolas Papernot, and Chiyuan Zhang. Measuring forgetting of memorized training examples. In *ICLR*, 2023.

Jaeyeon Kim, Seunggeun Kim, Taekyun Lee, David Z. Pan, Hyeji Kim, Sham Kakade, and Sitan Chen. Fine-tuning masked diffusion for provable self-correction. *arXiv preprint arXiv:2510.01384*, 2025.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *ICLR*, 2022.

Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *ICML*, 2024.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. In *IEEE Symposium on Security and Privacy (S&P)*, 2023.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. Membership inference attacks against language models via neighbourhood comparison. In *ACL Findings*, 2023.

Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. Quantifying privacy risks of masked language models using membership inference attacks. In *EMNLP*, 2022.

Milad Nasr, Javier Rando, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne

Ippolito, Christopher A. Choquette-Choo, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from aligned, production language models. In *ICLR*, 2025.

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*, 2025.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *ICLR*, 2025.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *NeurIPS*, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

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*, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *NeurIPS*, 2024a.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In *ICLR*, 2024b.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy (S&P)*, 2017.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627b token cleaned and deduplicated version of RedPajama, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Thomas Steinke, Milad Nasr, and Matthew Jagielski. Privacy auditing with one (1) training run. In *NeurIPS*, 2023.

Jiashu Tao and Reza Shokri. Information-theoretic membership inference for granular quantification of memorization. In *ICLR*, 2026.

Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *ICML*, 2014.

Lauren Watson, Chuan Guo, Graham Cormode, and Alexandre Sablayrolles. On the importance of difficulty calibration in membership inference attacks. In *ICLR*, 2022.

Roy Xie, Junlin Wang, Ruomin Huang, Minxing Zhang, Rong Ge, Jian Pei, Neil Zhenqiang Gong, and Bhuwan Dhingra. Recall: Membership inference via relative conditional log-likelihoods. In *EMNLP*, 2024.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

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

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE Computer Security Foundations Symposium (CSF)*, 2018.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models. In *ICLR*, 2022.

Sajjad Zarifzadeh, Philippe Liu, and Reza Shokri. Low-cost high-power membership inference attacks. In *ICML*, 2024.

Jingyang Zhang, Jingwei Sun, Eric Yeats, Yang Ouyang, Martin Kuo, Jianyi Zhang, Hao Frank Yang, and Hai Li. Min-k%++: Improved baseline for pre-training data detection from large language models. In *ICLR*, 2025.

A. Experimental Details

A.1. Threat Model and Access Assumptions

The attacker has score access to the target model M_{tgt}: for a chosen mask set, the attacker can query masked inputs and observe true-token reconstruction log-probabilities at the masked positions. The attacker also has access to a compatible reference model M_{ref}. The learned PRISM selector is trained using reference-model representations and generic public text only; it does not use target-model representations, member/non-member labels, or the evaluation examples. PRISM-Free removes this auxiliary training step and uses only the reference model's clean true-token score as a selector proxy.

A.2. Target Model Training Setup

For the LLaDA experiments, each target model is initialized from `GSAI-ML/ LLaDA-8B-Base` and fine-tuned separately for each MIMIR domain. For each domain, we use 1,000 member sequences for target-model fine-tuning and evaluate membership inference on a disjoint set of 1,000 held-out member examples and 1,000 non-member examples. Unless otherwise noted, the same fine-tuning recipe is used for all six domains.

We fine-tune with AdamW for 4 epochs using a learning rate of 5×10^{-5}, weight decay 0.1, and a linear learning-rate schedule with 500 warmup steps. The maximum sequence length is 512 tokens. Training is performed on 4 NVIDIA L40S GPUs in `bf16` with DeepSpeed ZeRO-3 and gradient checkpointing enabled. We use per-device batch size 1 and 12 gradient-accumulation steps, giving an effective batch size of 48 sequences per optimizer step.

Checkpoint selection is based on validation loss with early stopping. We monitor `eval_loss` and retain the best checkpoint under patience 3 and threshold 0.0. In practice, the retained checkpoints are typically the best saved checkpoints around `checkpoint-80`, and these checkpoints are used for all reported MIA evaluations.

A.3. PRISM Selector Training

Our default LLaDA PRISM selector is trained on a materialized 200k-sample English C4 subset. We split the materialized data into 190k training sequences and 10k held-out sequences; after preprocessing, the matched C4 run retains 189,919 training sequences. The selector is initialized from the base `GSAI-ML/LLaDA-8B-Base` backbone and trained for one epoch on 4 NVIDIA L40S GPUs in fp16 precision. The per-GPU batch size is 4, with 2 gradient-accumulation steps, giving an effective global batch size of 32. We use maximum sequence length 256, AdamW with learning rate 10^{-4}, weight decay 0, and random seed 42.

We train only the PRISM quality head and LoRA adapters, leaving the reference backbone otherwise fixed. This updates 29.38M trainable parameters out of 8.04B total parameters, corresponding to 0.37% of the model. LoRA is applied only to the query, key, and value projection modules, with rank 16, alpha 16, and dropout 0.1.

The training objective follows the PRISM self-correction formulation rather than direct regression to the one-hole score. For each clean sequence x_0, we sample a masking ratio $t \sim \mathrm{Uniform}(10^{-3}, 1)$ and independently mask each valid token with probability t to obtain x_t. The backbone predicts masked-token reconstruction logits on x_t. We then choose $K_{\mathrm{upd}} = 8$ masked positions, fill them with their argmax predictions to form a one-step updated sequence, and train the quality head to predict whether each updated token matches the ground-truth token. Thus, the quality label is generated internally from reconstruction correctness, not from external annotations, member/non-member labels, or target-model information.

The retained objective is

$$\mathcal{L}_{\mathrm{PRISM}} = \mathcal{L}_{\mathrm{BCE}} + 0.1\,\mathcal{L}_{\mathrm{CE}},$$

where $\mathcal{L}_{\mathrm{BCE}}$ is the self-correction binary cross-entropy loss on the one-step updated positions, and $\mathcal{L}_{\mathrm{CE}}$ is an auxiliary masked-token reconstruction cross-entropy loss over masked positions. Following the PRISM recipe, the reconstruction CE term is reweighted by the inverse masking ratio. No member/non-member labels, target-model weights, or target-model hidden states are used during selector training.

For the retained C4-200k checkpoint, training terminates after 2,968 optimizer steps with mean training loss 0.6535. The total loss decreases from 0.7617 at step 50 to 0.6433 at step 2950, while the BCE component decreases from 0.5493 to 0.4309. The checkpoint is saved after approximately 1,613 seconds, or 26.9 minutes. We do not use validation correlation for model selection in this retained run; instead, the selector is the epoch-1 checkpoint from this fixed one-epoch recipe, and its utility is evaluated through downstream MIA performance.

A.4. Baseline Configurations

For the SAMA baseline, we use our retained evaluation configuration with 16 progressive masking steps per example. At each step, SAMA enlarges the cumulative masked context according to its masking schedule and then samples 128 random local mask subsets from the valid token positions. Each subset contains 10 valid token positions. For every sampled subset, the attack evaluates the target model and the reference model under the corresponding masked input and records whether the target/ reference loss comparison favors membership. These subset-level signals are then aggregated over all 128 subsets and all 16 masking steps to produce the final membership score. Accordingly, the cost of SAMA is driven by both the multi-step masking schedule and the large number of repeated random subset evaluations at each step.

All attacks, including SAMA, are evaluated on the same 512-token windows with batch size 8. When reporting runtime, we normalize wall-clock measurements to seconds per sample, even though batching is used internally for efficiency. This normalization reflects the per-example auditing cost faced by the attacker and prevents repeated- query attacks such as SAMA from appearing artificially cheap due to implementation- level batching.

B. Bootstrap Confidence Intervals for Main Results

Table 4 reports the same per-domain JUMP results as *Table* 1 (main paper Table 1), augmented with 95% bootstrap confidence intervals to quantify uncertainty due to the finite evaluation split.

We use stratified bootstrap over evaluation examples. For each domain, the 1,000 sequences used to fine-tune the target model serve as member examples for MIA evaluation, and we pair them with 1,000 held-out non-member examples from the same domain. We do not assume Normality of the resulting bootstrap distributions.

We note that confidence intervals at the strictest operating point (TPR@0.1%FPR) are inherently wide: this metric is determined by only the few most extreme non-member scores out of 1,000, making it highly sensitive to bootstrap resampling. ROC-AUC and TPR at less strict FPRs, which depend on the full score distribution, exhibit substantially tighter intervals.

Table 4. Per-domain JUMP results on fine-tuned LLaDA-8B-Base, with 95% bootstrap CIs in brackets.

Domain	AUC	T@10	T@1	T@0.1
ArXiv	0.94 [.93,.95]	0.84 [.80,.86]	0.54 [.45,.60]	0.24 [.10,.47]
GitHub	0.86 [.84,.87]	0.67 [.62,.70]	0.33 [.24,.43]	0.12 [.00,.26]
HackerNews	0.77 [.74,.78]	0.40 [.34,.45]	0.11 [.07,.16]	0.03 [.00,.08]
PubMed Central	0.92 [.90,.93]	0.75 [.71,.80]	0.44 [.31,.52]	0.17 [.04,.32]
Wikipedia (en)	0.98 [.97,.99]	0.95 [.94,.97]	0.80 [.76,.85]	0.68 [.61,.78]
Pile-CC	0.96 [.95,.97]	0.90 [.88,.93]	0.56 [.52,.67]	0.17 [.01,.53]

C. Dream result

We additionally evaluate `Dream-v0-Base-7B` as a secondary dLLM check. These experiments are not used as the main evidence for JUMP, but test whether joint multimask probing remains meaningful beyond LLaDA. For Dream, the strongest retained PRISM configuration uses the same joint probing budget, $K = 64$, with clipping threshold $c = \log 2.5$.

Table 5. Dream aggregate results averaged over six MIMIR domains.

Method	ROC-AUC	TPR@10%	TPR@1%	TPR@0.1%
Loss	0.5461	0.1422	0.0205	0.0055
ZLIB	0.5508	0.1502	0.0275	0.0082
Dream PRISM	0.8250	0.5673	0.1704	0.0503
Dream SAMA	**0.8513**	**0.6122**	**0.2862**	**0.1660**

Table 5 shows that the broad qualitative pattern transfers to Dream. Vanilla likelihood baselines remain weak, while joint multimask probing produces a much stronger membership signal. Dream PRISM improves substantially over Loss and ZLIB across all metrics, indicating that selector-guided joint probing is not specific to LLaDA. In the current Dream setup, however, Dream SAMA remains stronger than Dream PRISM, especially at stricter FPRs. We therefore use Dream as a secondary-model analysis rather than as the main evidence for JUMP.

Table 6. Domain-wise Dream PRISM and Dream SAMA results. Each entry reports ROC-AUC / TPR@10%FPR / TPR@1%FPR / TPR@0.1%FPR.

Domain	Dream PRISM	Dream SAMA
ArXiv	0.7830 / 0.4812 / 0.1518 / 0.0372	0.8435 / 0.5680 / 0.2920 / 0.2230
GitHub	0.8147 / 0.5500 / 0.0841 / 0.0062	0.8332 / 0.6180 / 0.2590 / 0.1190
HackerNews	0.8216 / 0.5096 / 0.1473 / 0.0491	0.8295 / 0.5670 / 0.1710 / 0.0540
Pile-CC	0.8716 / 0.6821 / 0.2846 / 0.1709	0.8972 / 0.7160 / 0.3470 / 0.1910
PubMed Central	0.8149 / 0.5571 / 0.1193 / 0.0257	0.8425 / 0.5850 / 0.3030 / 0.1630
Wikipedia (en)	0.8443 / 0.6239 / 0.2353 / 0.0127	0.8621 / 0.6190 / 0.3450 / 0.2460

Table 6 reports the corresponding domain-wise results. The relative ordering is consistent across most domains: Dream PRISM clearly improves over likelihood-style baselines, while Dream SAMA remains the stronger Dream attack in the current configuration. This suggests that the single-pass joint probing idea transfers beyond LLaDA, but the best localization strategy can depend on the dLLM family.

Table 7. Dream selector-free and hand-designed controls averaged over six MIMIR domains. Runtime is reported in seconds per sample.

Method	ROC-AUC	Runtime
Random Joint	0.6127	0.1753
Entropy Top	0.6455	0.1396
PRISM-Free (Ref-LowProb)	0.5968	0.1297
Dream PRISM	0.8250	0.1339
Dream SAMA	0.8513	1.1699

We further evaluate selector-free and hand-designed Dream controls to separate attack quality from efficiency. As shown in Table 7, these controls are weaker than both Dream PRISM and Dream SAMA, but they show that non-random position choice can still provide non-trivial signal without a learned selector. They are also efficient: Random Joint, Entropy Top, Ref-LowProb, and Dream PRISM are all substantially cheaper than Dream SAMA. Thus, although Dream SAMA gives the strongest attack quality in this setup, Dream PRISM and the selector-free variants remain much cheaper while preserving a meaningful membership signal.

Overall, the Dream results should be read as a robustness check. They support the broader claim that joint masked probing is a useful dLLM-specific MIA primitive beyond LLaDA, while also showing that the relative advantage of PRISM-based localization can depend on the dLLM family. For this reason, we keep LLaDA as the primary model for the main claim and report Dream as appendix-only supporting evidence.

D. Additional Selector and Scoring Ablations

D.1. Selector-Controlled Baselines

We include selector-controlled baselines to separate the contribution of joint multimask probing, hard-position selection, reference calibration, and the learned PRISM selector. All methods in *Table* 8 use the same six MIMIR domains and the same selected-token budget. They differ only in how positions are selected or whether the final score uses a reference model.

Table 8. Selector-controlled LLaDA baselines averaged over six MIMIR domains.

Method	ROC-AUC	TPR@10%	TPR@1%	TPR@0.1%
Random Joint	0.8153	0.5284	0.2168	0.0776
Entropy Top	0.8788	0.6857	0.3532	0.1112
PRISM-Free (Ref-LowProb)	0.8617	0.6374	0.3045	0.1167
Target-Only	0.5551	0.1406	0.0234	0.0064
JUMP	**0.9041**	**0.7518**	**0.4636**	**0.2337**

Random joint masking. Random Joint masks K valid token positions uniformly at random and then applies the same target–reference clipped aggregation used by JUMP. Its non-trivial performance shows that joint multimask probing alone is a meaningful DLM attack primitive, but it is clearly weaker than hard-position selection.

Entropy and reference-low-probability selection. Entropy Top selects uncertain positions according to the selector distribution, while PRISM-Free selects positions with low clean-text true-token probability under the reference model. Both

are strong controls, confirming that membership signal is concentrated on difficult positions. Their gap to JUMP, especially at low FPR, suggests that the learned hard-position selector provides a better localization prior than these simpler heuristics.

Target-only control. Target-Only uses PRISM-selected positions but removes the reference-model discrepancy from the final score. Its weak performance indicates that the selector is not directly acting as a membership classifier. The main gain comes from combining hard-position localization with target–reference reconstruction advantage.

D.2. Selector Diagnostics

We further diagnose whether JUMP is simply reproducing token rarity or clean reference-model confidence. To avoid using statistics from the evaluation split, we define the RareToken control using an external unigram frequency table computed from C4. RareToken (C4-freq) selects the K valid positions with the lowest C4 token frequencies, and then applies the same joint masking and clipped target–reference scoring rule as JUMP. All diagnostics use the same selected-token budget, $K = 64$, and the same six MIMIR domains as the main LLaDA experiments.

Table 9. Selector diagnostic summary averaged over six MIMIR domains. Non-member P99 and P99.9 denote upper-tail score quantiles of the non-member distribution. RareToken uses an external C4 unigram frequency table.

Method	TPR@10%	TPR@1%	TPR@0.1%	NM P99	NM P99.9
JUMP	**0.7518**	**0.4636**	**0.2337**	**0.1032**	**0.1402**
RareToken (C4-freq)	0.5262	0.1899	0.0283	0.2585	0.3652
PRISM-Free (Ref-LowProb)	0.6374	0.3045	0.1167	0.1128	0.1523

Table 9 shows that token rarity alone is not sufficient to explain the gains of JUMP. RareToken (C4-freq) is weaker than JUMP at every operating point and is especially poor in the strict low-FPR regime. This gap is consistent with the non-member tail statistics: RareToken has much larger average NM P99/P99.9 scores than JUMP. Thus, external-corpus rarity yields a noisy score with many high-scoring non-member outliers, whereas JUMP produces a cleaner high-confidence tail.

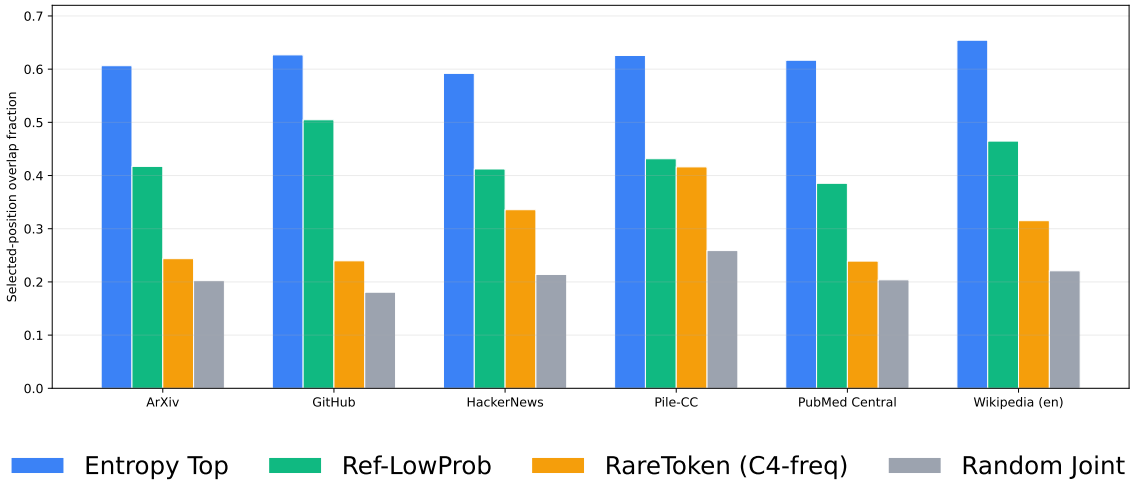


Figure 7. Overlap between JUMP-selected positions and alternative selector rules across domains. The legend is shared across domains. Entropy Top has the largest overlap with JUMP, but substantial disagreement remains. RareToken (C4-freq) and Random Joint have much smaller overlap, showing that JUMP is not simply selecting rare tokens or random positions.

Figure 7 further shows that JUMP is not equivalent to simple selector heuristics. Averaged over domains, Entropy Top overlaps with JUMP on 62.0% of selected positions, Ref-LowProb on 43.6%, RareToken (C4-freq) on 29.8%, and Random Joint on 21.3%. Thus, JUMP captures part of the clean-uncertainty signal reflected by entropy, but is much less aligned with simple rarity or clean reference probability. This supports the interpretation that the learned selector captures a broader token-level uncertainty signal than any single hand-designed heuristic.

Figure 8 visualizes the same effect at the score-distribution level. With external C4 frequencies, RareToken no longer behaves as a strong broad-ranking surrogate. In several domains, its non-member scores extend much farther into the upper

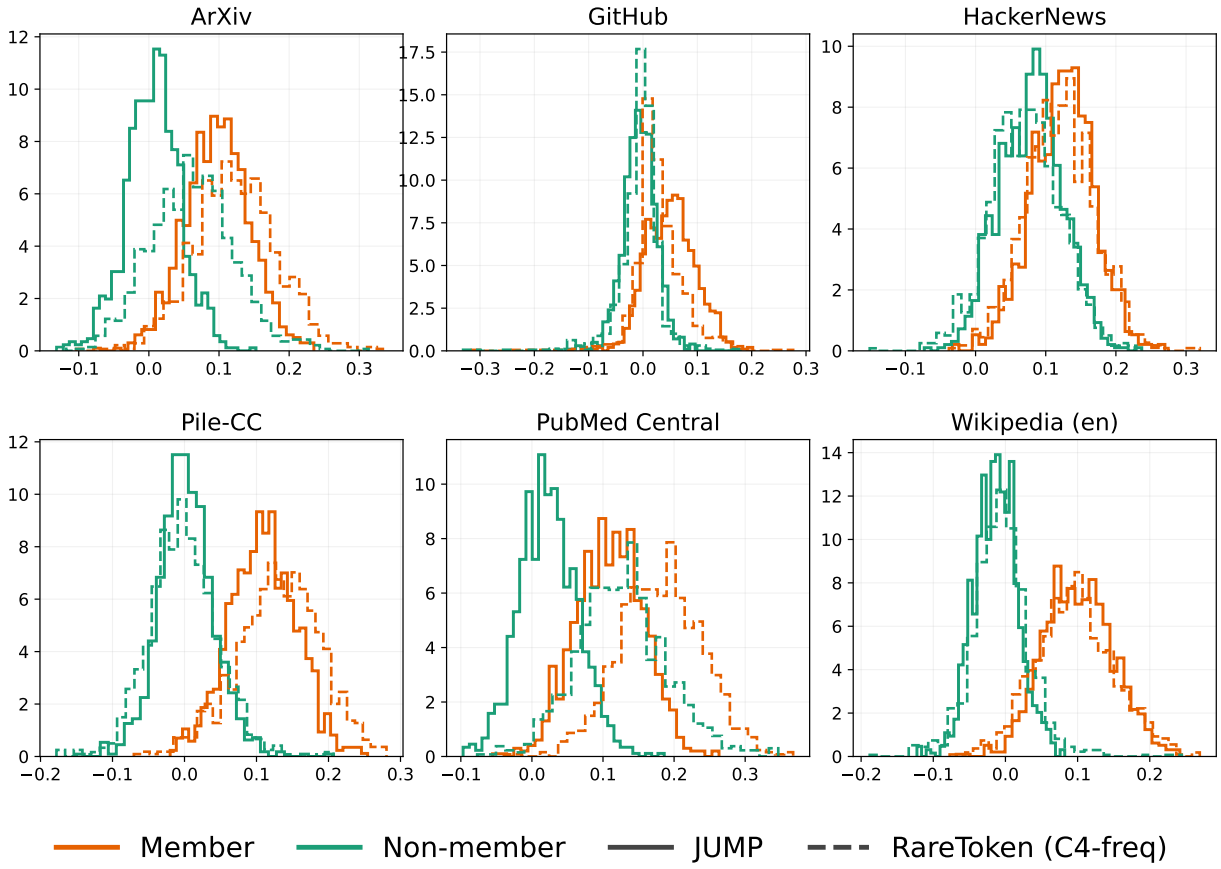


Figure 8. Score distributions for JUMP and RareToken across domains, where RareToken uses external C4 token frequencies. RareToken produces a substantially heavier non-member upper tail, which explains its weak low-FPR performance relative to JUMP.

tail than those of JUMP, directly weakening TPR@1%FPR and TPR@0.1%FPR. JUMP is therefore better understood as the combination of learned low-confidence localization, target–reference discrepancy scoring, and clipped aggregation, rather than as a rarity-based heuristic.

D.3. Selector Training Sample-Size Ablation

We further ablate the amount of selector-training data used for the PRISM head. Using the same LLaDA backbone and matched selector-training hyperparameters, we train PRISM heads on C4 subsets of size 20k, 50k, 100k, 200k, 400k, and 500k. All selectors are evaluated with the same downstream JUMP pipeline: $K = 64$ selected positions, a single joint multimask query, the target–reference true-token log-probability gap, and clipped aggregation with threshold $[-\log 1.5, \log 1.5]$. The evaluation averages over the same six MIMIR domains used in the main LLaDA experiments.

Table 10. C4 selector-training sample-size ablation for the LLaDA PRISM head. All rows use the same JUMP scoring rule and report averages over six MIMIR domains.

C4 samples	Mean ROC-AUC	Mean TPR@1%	Mean TPR@0.1%	Train time
20k	0.8933	0.3916	0.1679	190s
50k	0.8965	0.3789	0.1931	421s
100k	0.8914	0.3584	0.1895	813s
200k	0.8987	**0.4453**	**0.2185**	1,613s
400k	**0.9012**	0.4246	0.2153	3,158s
500k	0.8974	0.4270	0.2053	3,922s

Table 10 and *Figure* 9 show that selector performance largely saturates by 200k training samples. Moving from 200k to 400k gives only a small increase in mean ROC-AUC, from 0.8987 to 0.9012, but does not improve the low-FPR metrics that

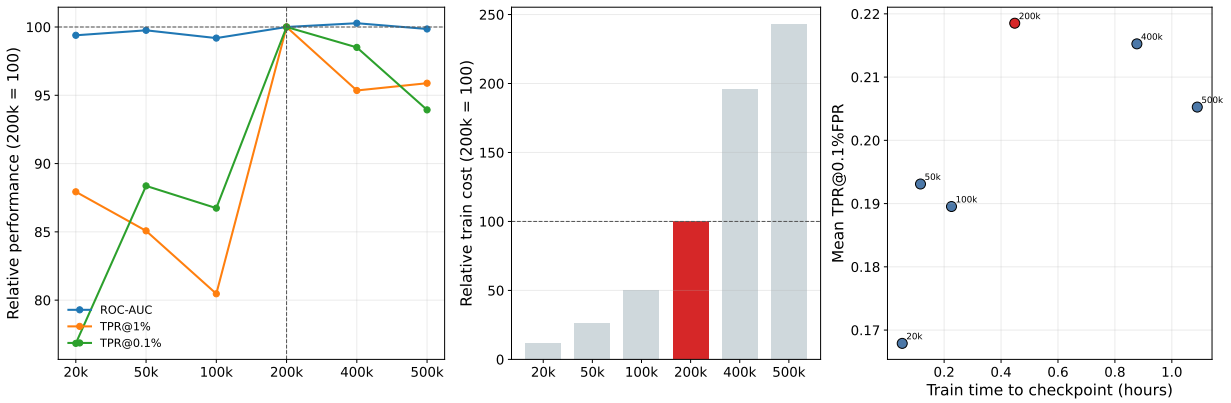


Figure 9. Effect of selector-training set size for the C4-trained PRISM head. We train LLaDA PRISM heads on matched C4 subsets from 20k to 500k samples and evaluate each selector with the same JUMP pipeline across six MIMIR domains. Performance improves rapidly at small scales and largely saturates by 200k. Although 400k gives a slightly higher mean ROC-AUC, 200k gives the best average low-FPR performance while requiring much less training time.

are most important for high-confidence membership inference. The 200k selector achieves the best average TPR@1%FPR and TPR@0.1%FPR among the tested budgets. The matched 500k rerun also does not improve over 200k, despite requiring substantially more training time.

The cost-performance tradeoff therefore supports using 200k C4 samples as the main selector-training budget. Relative to 200k, the 400k and 500k selectors require approximately $1.96\times$ and $2.43\times$ more training time, respectively, without improving low-FPR behavior. Thus, 200k is not necessarily uniformly optimal on every metric, but it is the most practical operating point: it is already near the ROC-AUC plateau, gives the strongest average low-FPR performance, and avoids the substantially larger selector-training cost of larger C4 subsets.

E. Budget and Clipping Details

E.1. Selected-Token Budget Sweep

Figure 10 reports the selected-token budget sweep for the low-FPR metrics. Across domains, $K = 64$ provides the most stable tradeoff and is therefore used as the default setting. Using only $K = 32$ positions reduces the number of aggregated token-level membership signals, while increasing to $K = 128$ does not consistently improve low-FPR detection and can introduce less informative positions.

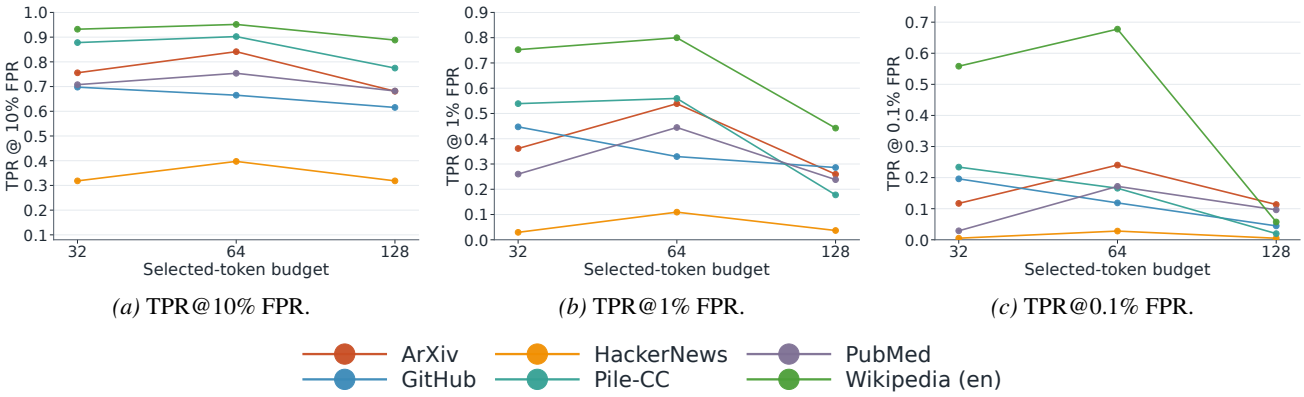
Importantly, increasing K does not materially increase attack cost in our implementation. All selected positions are scored within the same joint masked-query pipeline, so the dominant cost remains the selector, target-model, and reference-model forward passes rather than the number of retained positions itself. *Table* 11 reports the measured runtime for the same sweep. Averaged over the six LLaDA domains with clipping threshold $\tau = \log 1.5$, the total wall-clock time is 392.1 s for $K = 32$, 385.7 s for $K = 64$, and 365.0 s for $K = 128$, corresponding to 0.196, 0.193, and 0.183 seconds per sample, respectively. Thus, the practical runtime remains essentially flat across the K sweep, supporting the NFE-based claim that increasing K does not change the target/reference scoring cost.

Table 11. Runtime of the selected-token budget sweep, averaged over the six LLaDA domains with clipping threshold $\tau = \log 1.5$. Increasing K does not materially increase wall-clock cost because all selected positions are scored in the same joint masked-query pipeline.

Selected-token budget K	Total runtime (s)	Runtime / sample (s)
32	392.1	0.196
64	385.7	0.193
128	365.0	0.183

E.2. Clipping Operation Ablation

Table 12 reports the aggregate clipping ablation corresponding to Section 6.3. The selector is fixed to the PRISM-C4 head, and both readouts use the same selected token positions. The unclipped variant averages raw target/reference token



(a) TPR@10% FPR. *(b)* TPR@1% FPR. *(c)* TPR@0.1% FPR.

Figure 10. Detailed selected-token budget sweep across low-FPR operating points. The legend is shared across all panels. The default $K = 64$ gives a stable operating point across domains: smaller budgets can weaken the aggregate signal, while larger budgets do not consistently improve low-FPR detection.

log-probability gaps, while the clipped variant caps each token contribution before aggregation.

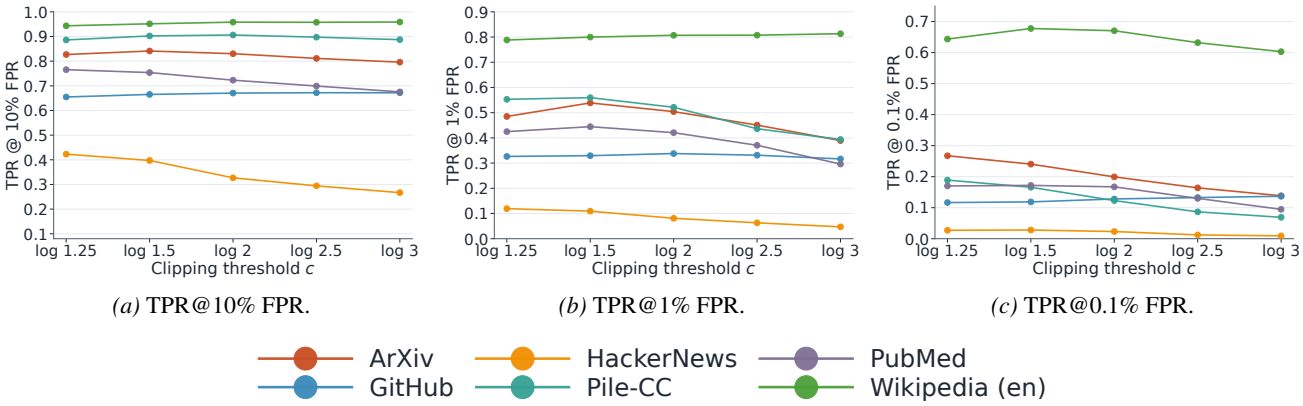
Table 12. Clipping ablation averaged over six MIMIR domains.

Method	ROC-AUC	TPR@10%	TPR@1%	TPR@0.1%
No clip mean	0.8166	0.5478	0.2323	0.1003
Clipped mean	**0.8987**	**0.7358**	**0.4453**	**0.2185**

Clipping improves every metric, with the largest relative gain at the strictest operating point. In particular, TPR@0.1%FPR more than doubles after clipping. This behavior is consistent with the role of clipping as a robustness correction: selected hard positions can produce extreme target/reference gaps, and an unclipped mean can be dominated by a small number of unstable token-level contributions. Clipping limits these isolated spikes and instead rewards sequences where the target has a stable reconstruction advantage across multiple selected positions.

E.3. Clipping-threshold sweep

We provide the full clipping-threshold sweep for the low-FPR metrics omitted from the main text. The threshold c controls the per-token cap in Eq. (6); smaller values are more aggressive and suppress token-level outliers more strongly, while larger values approach the unclipped mean. Across the full domain–metric grid, $c = \log 1.5$ provides the best aggregate operating point. Some individual domain–metric pairs can prefer a nearby threshold, but $c = \log 1.5$ gives the most stable tradeoff once TPR at low FPR is considered. We therefore use $c = \log 1.5$ as the default in all main experiments.



(a) TPR@10% FPR. *(b)* TPR@1% FPR. *(c)* TPR@0.1% FPR.

Figure 11. Full clipping-threshold sweep for low-FPR metrics. The legend is shared across all panels. The default $c = \log 1.5$ gives the best aggregate operating point across domains and metrics, even though individual domain–metric pairs may prefer nearby thresholds.

F. Implementation and Runtime Details

F.1. Selector Training Cost

For the selector-corpus ablation in Section 6.1, all PRISM heads are trained with the same budget of 200k samples. This matched budget isolates the effect of the selector training corpus rather than the amount of selector supervision. The downstream attack pipeline is otherwise identical across PRISM-C4, PRISM-Slim, and PRISM-FW: each selector chooses $K = 64$ positions, after which the same joint multimask query and clipped target–reference scoring rule are applied.

The selector-free variant removes the learned PRISM head. Instead, it ranks valid positions by the reference model's clean-text true-token logit and selects the K positions with the lowest values. Thus, the selector-free variant uses a reference-side difficulty heuristic without any auxiliary selector training.

PRISM-head training is a one-time preprocessing cost. In our implementation, PRISM-C4 takes 26.9 minutes of wall time, corresponding to approximately 1.8 GPU-hours on 4×L40S GPUs. PRISM-Slim takes 23.9 minutes, or approximately 1.6 GPU-hours, and PRISM-FW takes 26.8 minutes, or approximately 1.8 GPU-hours. When amortized over the evaluation suite of six datasets with 2,000 samples each, these costs correspond to 0.134 seconds per sample for PRISM-C4, 0.120 seconds per sample for PRISM-Slim, and 0.134 seconds per sample for PRISM-FW.

F.2. Per-Sample Runtime Accounting

All reported runtimes are normalized as seconds per sample. Although attacks are executed in batches for efficiency, per-sample runtime is the most direct measure of attack cost from the auditor's perspective. This is particularly important for DLM MIAs because attacks that repeatedly query many masked variants can become expensive even when each individual forward pass is parallelized.

Table 13. Runtime per sample in seconds. "Head" denotes amortized one-time selector training.

Method	Inference	Head	Total
PRISM-C4	0.184	0.134	0.318
PRISM-Slim	0.185	0.120	0.305
PRISM-FW	0.183	0.134	0.317
PRISM-Free	0.173	–	0.173
SAMA	1.863	–	1.863

Even after amortizing selector-head training, the learned-selector variants remain substantially faster than SAMA. The computational advantage comes from the single-pass design: once positions are selected, all selected tokens are reconstructed jointly rather than through many separate masked subsets.

G. Defense via Differentially Private Fine-Tuning

We evaluate differentially private fine-tuning as a principled defense against JUMP. Whereas output-level redaction defenses depend on hiding a particular signal from the API, DP fine-tuning targets the underlying cause of membership leakage: the target model's tendency to over-fit individual fine-tuning examples. Because JUMP relies on subtle target–reference reconstruction-gap differences aggregated over a small set of selected positions, even modest suppression of per-example memorization should translate into a large reduction in attack signal. We test this hypothesis with DP-LoRA at $(\varepsilon, \delta) = (8.0, 10^{-5})$, a relatively weak privacy regime by formal standards but a standard configuration for DP fine-tuning of large language models (Yu et al., 2022; Li et al., 2022).

G.1. Defense Setup

We fine-tune each LLaDA-8B-Base target with DP-SGD applied only to LoRA adapter parameters. LoRA adapters are attached to the query and value projections (`q_proj`, `v_proj`) with rank $r = 16$, $\alpha = 16$, and dropout 0.1. This results in 8.39M trainable parameters out of 8.02B total, corresponding to 0.10% of the full model. The frozen backbone keeps the base reconstruction interface used by JUMP, so attack-side scoring remains directly comparable to the undefended setting.

Training uses $4\times$ NVIDIA L40S GPUs in bf16 with DeepSpeed ZeRO-3, per-device batch size 1, gradient accumulation 12 (effective batch size 48), learning rate 5×10^{-5}, weight decay 0.1, 500 warmup steps, and 4 epochs at maximum sequence

length 512. For privacy, we set the target privacy budget to $(\varepsilon, \delta) = (8.0, 10^{-5})$, use flat per-sample gradient clipping with norm 1.0, and the PRV accountant. Under this configuration, Opacus selects a noise multiplier of $\sigma = 0.4211$. The full DP-LoRA fine-tuning runtime is approximately 57 minutes per domain on 4 GPUs (3,418.7 seconds averaged over the six domains).

G.2. Attack Evaluation Protocol

We evaluate the defended targets using the same JUMP configuration as in the main experiments: the matched C4-200k PRISM selector, `quality_bot` selection, selected-token budget $K = 64$, and clipped mean target/reference true-token log-probability gap aggregation with $\tau = \log 1.5$. Each evaluation uses 1,000 member and 1,000 non-member examples per domain, with 10 bootstrap resamples (seed 42) for stability. The reference model and PRISM selector are unchanged from the main experiments; only the target model is replaced by its DP-LoRA-fine-tuned counterpart.

G.3. Utility Metric

We measure utility as the held-out masked reconstruction loss on the 1,000 non-member examples from each domain, and exponentiate it to obtain a *reconstruction perplexity*. Because LLaDA is a masked diffusion language model, this quantity is the exponentiated masked reconstruction loss under the same diffusion-style objective the model was trained on, not autoregressive perplexity. Reporting the metric on non-members ensures that the utility number reflects domain adaptation rather than memorization of the fine-tuning set.

G.4. Results

JUMP suppression. Table 14 reports JUMP ROC-AUC and low-FPR TPR for the undefended target and the DP-LoRA-defended target across all six MIMIR domains. DP-LoRA at $\varepsilon = 8$ reduces mean ROC-AUC from 0.9041 to 0.5020, bringing the attack close to chance (0.50) on every domain. The largest defended AUC across domains is 0.5228 (GitHub) and the smallest is 0.4673 (Pile-CC). Low-FPR detection is also essentially eliminated: mean TPR at 1% FPR drops from 0.4636 in the undefended setting (Table 1 of the main paper) to 0.0131 under DP-LoRA, and TPR at 0.1% FPR drops to 0.0025. These low-FPR values are at or near the FPR levels themselves, indicating no usable high-confidence membership signal remains.

Table 14. JUMP performance against the DP-LoRA-defended LLaDA target at $(\varepsilon, \delta) = (8.0, 10^{-5})$. Undefended (non-DP) AUC is included for reference. Across all six domains, DP-LoRA drives JUMP near chance and eliminates low-FPR detection.

Domain	non-DP AUC	DP-LoRA AUC	TPR@10%	TPR@1%	TPR@0.1%
ArXiv	0.9380	0.5133	0.1178	0.0136	0.0022
GitHub	0.8591	0.5228	0.1206	0.0169	0.0044
HackerNews	0.7664	0.5198	0.1276	0.0145	0.0028
Pile-CC	0.9622	0.4673	0.0777	0.0096	0.0030
PubMed Central	0.9183	0.4956	0.0952	0.0107	0.0003
Wikipedia (en)	0.9803	0.4934	0.1200	0.0130	0.0026
Average	**0.9041**	**0.5020**	**0.1098**	**0.0131**	**0.0025**

Utility cost. Table 15 reports held-out non-member reconstruction loss and reconstruction perplexity for the same targets. Averaged across the six domains, DP-LoRA increases non-member reconstruction perplexity by a factor of 1.115, corresponding to roughly +11.5%. Domain-level cost varies: Pile-CC, GitHub, and Wikipedia (en) show small overheads (+3.9% to +6.4%), while HackerNews and PubMed Central show larger overheads (+17.5% and +19.9%). In all cases, the defended perplexity remains in the same quality regime as the undefended fine-tuned target rather than collapsing toward base-model perplexity, indicating that domain adaptation is preserved with bounded degradation.

G.5. Discussion

The DP-LoRA results support a simple interpretation of why JUMP is effective and how to defend against it. JUMP localizes membership signal at low-reference-confidence positions and aggregates target/reference reconstruction gaps clipped at $\tau = \log 1.5 \approx 0.405$. The per-token signal is small in absolute terms (Section Section 4.1 reports a member/non-member gap difference of +0.174 in nats under one-hole probing), and the attack's strength comes from consistently combining many such small contributions. DP-LoRA suppresses the per-example memorization that produces this consistency: by

Table 15. Utility cost of DP-LoRA on held-out non-member data. Loss is the masked reconstruction loss, and reconstruction perplexity is its exponential. The rightmost column shows the multiplicative perplexity increase (DP-LoRA / non-DP).

Domain	non-DP loss	DP-LoRA loss	non-DP ppl	DP-LoRA ppl	ppl ratio
ArXiv	1.5551	1.6998	4.7356	5.4729	1.1557
GitHub	1.0197	1.0577	2.7725	2.8796	1.0386
HackerNews	1.7454	1.9068	5.7280	6.7316	1.1752
Pile-CC	1.4491	1.5115	4.2593	4.5336	1.0644
PubMed Central	1.4347	1.6158	4.1982	5.0321	1.1986
Wikipedia (en)	1.1989	1.2559	3.3163	3.5109	1.0587
Average	**1.4005**	**1.5079**	**4.1683**	**4.6935**	**1.1152**

adding calibrated noise during fine-tuning, it bounds the influence of any single training example on the resulting model. Because JUMP depends on aggregating fine-grained per-example reconstruction advantages, even relatively weak DP ($\varepsilon = 8$) is sufficient to bring the attack to chance.

In contrast to output-redaction approaches that act only at the score interface, DP-LoRA provides a formal (ε, δ)-DP guarantee on the fine-tuning procedure. This guarantee bounds the success of *any* membership inference attack that operates on the fine-tuned model, including adaptive variants of JUMP, distillation-based attacks, and future attacks that exploit other aspects of the dLLM interface. The price is a moderate $+11.5\%$ increase in held-out reconstruction perplexity, consistent with reported overheads in DP fine-tuning of similarly-sized language models (Yu et al., 2022). We view this trade-off as favorable: a single DP-LoRA fine-tuning run, taking under one hour on four L40S GPUs per domain, neutralizes JUMP across all six MIMIR domains.

H. Broader Impacts

This work has both positive and negative societal implications. On the positive side, JUMP provides a stronger and more efficient privacy-auditing tool for identifying membership leakage in fine-tuned diffusion language models, helping model developers evaluate and mitigate privacy risks before deployment. On the negative side, stronger membership inference attacks are dual-use: if misapplied, they could be used to test whether sensitive text was included in a private fine-tuning corpus. To reduce this risk, we frame JUMP as an auditing tool, evaluate it on benchmark datasets, and provide a DP-LoRA fine-tuning defense as a concrete mitigation strategy with formal (ε, δ)-DP guarantees.

I. Existing assets and licenses

We use existing publicly released models, datasets, and baseline resources. Table 16 summarizes the assets used in this work, their roles, and the corresponding license or terms-of-use information. We do not redistribute the original datasets or pretrained model weights as part of this submission; users should obtain each asset from its original source and comply with the corresponding license and terms of use.

Table 16. Existing assets used in this paper.

Asset	Role in this paper	Citation / source	License / terms
LLaDA-8B-Base	Main base/reference dLLM and fine-tuning target	Nie et al. (2025)	MIT
Dream-v0-Base-7B	Secondary dLLM evaluation in appendix	Ye et al. (2025)	Apache-2.0
MIMIR	Member/non-member evaluation benchmark	Duan et al. (2024)	MIT; gated access terms on Hugging Face
C4	PRISM selector training corpus	Raffel et al. (2020)	ODC-BY; Common Crawl Terms of Use
SlimPajama	Selector-corpus ablation	Soboleva et al. (2023)	Apache-2.0 for the original Cerebras release; subset-specific terms if using a split or reupload
FineWeb	Selector-corpus ablation	Penedo et al. (2024)	ODC-By v1.0; Common Crawl Terms of Use
SAMA	Prior dLLM MIA baseline	Chen et al. (2026)	MIT if using the official code repository