DETERMINISTIC JOINT-RISK SAMPLING FOR EFFICIENT DISCRETE DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion language models repeatedly refine masked tokens, often spending forward passes on predictions that have already stabilized. Fixed transfer budgets do not adapt the number of commitments to observed token stability, creating a trade-off between unnecessary refinement and premature commitment. We introduce **Deterministic Joint-Risk Sampling (DJRS)**, which ranks masked tokens using confidence, margin, Jensen–Shannon divergence, and prediction change, then admits additional low-risk commitments under a progress-dependent budget. On LLaDA-8B-Instruct, DJRS reduces the number of forward evaluations (NFE) by 29% on GSM8K at 64 steps (64.0 to 45.6) with a 0.6-percentage-point accuracy decrease, and raises MATH-500 accuracy from 27.8% to 31.4% while reducing NFE by 22%. In a separate Dream-v0-Instruct-7B HumanEval evaluation, DJRS reaches 38.4% pass@1 at 39.8 NFE, compared with 32.9% at 64.0 NFE for Default. Trace analysis shows that margin most consistently separates stable from flip-prone masked positions, while JSD shows less monotonic associations with flips. We present DJRS as a practical adaptive-decoding heuristic rather than a calibrated risk guarantee.

1 INTRODUCTION

Masked diffusion language models (Nie et al., 2025; Sahoo et al., 2024) generate text by iteratively unmasking tokens from a fully masked canvas. At each denoising step, the model performs a forward evaluation and predicts distributions for all masked positions; a remasking strategy then commits some tokens and keeps others masked. The number of denoising steps T is a key hyperparameter: too few steps produce low-quality text, while too many waste compute on tokens that have already stabilized.

The denoising trajectory is non-uniform: some positions become stable early, while others remain sensitive to contextual updates. Nevertheless, standard low-confidence remasking applies a largely fixed transfer schedule and repeatedly evaluates positions that may already be stable. Conversely, committing uncertain positions too early can lock in incorrect predictions and degrade the final answer. The central problem is therefore to decide which masked positions can be safely committed at each step while preserving the option to revise uncertain positions.

We approach this as a *flip prediction* problem: given the model's predictions at step t, will a masked token's top-1 prediction change at step $t + 1$? We use four observable stability signals:

1. **Confidence**: the top-1 probability.
2. **Margin**: the gap between the top-1 and top-2 probabilities.
3. **JSD**: the Jensen–Shannon divergence between the token's distributions at consecutive steps.
4. **Prediction change**: whether the top-1 prediction changed relative to the preceding step.

This formulation turns token commitment into a risk-ranking problem: positions that appear stable can be committed earlier, while positions likely to flip remain masked for later refinement.

We instantiate this risk-ranking view with **Deterministic Joint-Risk Sampling (DJRS)**. DJRS combines the four signals into a per-position risk score, ranks masked positions by increasing risk,

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.

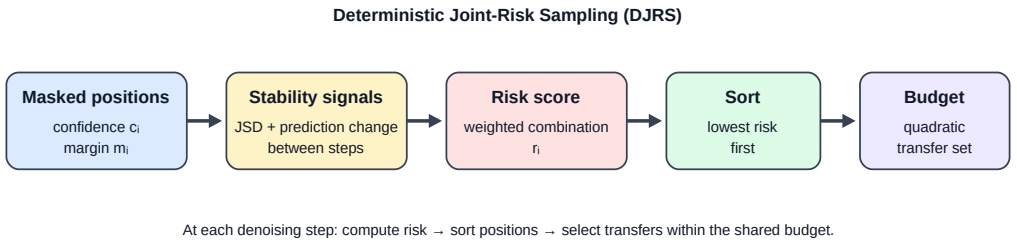

Figure 1: DJRS computes a multi-signal risk score, ranks masked positions, and adds low-risk transfers subject to a shared quadratic budget.

and selects a prefix under a progress-dependent cumulative budget. The budget is conservative early in a block and more permissive near completion. In LLaDA, DJRS is additive to the standard remasking schedule: ordinary low-confidence transfers are preserved, while additional low-risk commitments are selectively admitted. This design targets a better compute–quality operating point without changing the backbone model or training procedure.

Our contributions are:

1. We formulate adaptive token commitment in masked diffusion language models as a per-position stability and flip-prediction problem.
2. We introduce DJRS, a deterministic sampler that combines confidence, margin, distributional change, and temporal prediction change into a single risk score and applies a progress-dependent cumulative budget, with an operational specification of its integration with the decoding schedule.
3. We provide an empirical evaluation across mathematical reasoning and code-generation settings, together with separated calibration analysis and signal-composition sensitivity experiments; the results are interpreted as task- and budget-dependent compute–quality trade-offs.

2 Related Work

Masked diffusion language models. LLaDA (Nie et al., 2025) and MDLM (Sahoo et al., 2024) formulate text generation as iterative unmasking from a fully masked canvas. At each denoising step, the model predicts distributions for all masked positions and a remasking strategy selects which to commit. Low-confidence remasking (Nie et al., 2025) commits the highest-confidence tokens first, leaving uncertain positions masked for later steps. FastdLLM (Wu et al., 2025) accelerates generation via KV-cache reuse and block-wise unmasking, where the sequence is divided into blocks that are denoised sequentially. Our fixed-quota baseline configuration uses a predetermined transfer schedule. DJRS differs by adapting the commitment schedule *within* the denoising loop—it commits stable tokens earlier and defers uncertain ones—without changing the nominal step budget or the backbone model.

Non-autoregressive and iterative text generation. Non-autoregressive generation has a long history in NLP. Insertion-based and edit-based approaches (Stern et al., 2019; Gu et al., 2019) generate tokens in a non-left-to-right order by predicting insertion positions or refining partially decoded sequences. Masked diffusion models inherit this non-left-to-right generation order but add an explicit denoising process: starting from all-masked tokens, the model iteratively commits positions over multiple steps. Discrete diffusion (Austin et al., 2021) formalizes this as a Markov chain over token states. Our work builds on this framework but addresses a complementary question: given a fixed denoising budget, how should tokens be *prioritized* for commitment at each step? Prior non-autoregressive methods focus on *what* to generate and in *what order*; we focus on *when* within the denoising trajectory a token is stable enough to commit.

Early exit and adaptive computation. Early exit has been studied for autoregressive transformers (Schuster et al., 2022), where the model can exit intermediate layers when a token prediction is sufficiently confident. In the diffusion image domain, accelerated samplers such as DDIM (Song et al., 2021) and DPM-Solver (Lu et al., 2022) are examples of continuous-domain methods that reduce sampling steps through modified trajectories or solvers. Classifier-free guidance (Ho & Salimans, 2022) is another continuous-domain control mechanism, rather than a direct baseline for token commitment. These methods modify continuous-state sampling rather than discrete token commitment. Our setting differs in two ways: (i) we adapt the *commitment schedule* within the denoising loop, not the depth of a single forward pass or the step size of a continuous ODE; and (ii) discrete diffusion has a combinatorial commitment structure—each position is either masked or committed—so the analogue of "early exit" is deciding *which* tokens to commit early, not merely *when* to stop.

Uncertainty estimation in language models. For autoregressive models, nucleus sampling truncates the low-probability tail of the predictive distribution (Holtzman et al., 2020), while uncertainty estimation has been studied for token- and sequence-level error detection (Malinin & Gales, 2021). In masked diffusion, the per-token confidence at each step is a natural stability signal, but as we show in our controlled analysis (Section 5), confidence alone does not fully predict whether a token will flip in subsequent steps. Margin and Jensen–Shannon divergence capture complementary distribution-level information, and our temporal prediction-change indicator captures whether the model's top prediction has already shifted. DJRS combines all four into a single risk score; the reimplemented baselines instead use their respective method-specific stability or convergence criteria.

Confidence-based stopping baselines. We reimplemented three published, training-free stopping or commitment policies in the same FastdLLM/LLaDA evaluation harness: LATCH (Lee et al., 2026), Prophet (Li et al., 2026), and SureLock (Oba et al., 2026). The implementations preserve each method's decision rule while adapting only the model wrapper, cache interface, and block-level bookkeeping needed by our evaluation. Exact operating points are LATCH confidence 0.70, patience 2, maximum holes 1; Prophet thresholds $(7.5, 5.0, 2.5)$ for early, middle, and late progress; and SureLock uncertainty percentile 20 with KL threshold 5×10^{-4}. All methods use greedy decoding, block size 32, low-confidence remasking, and the same model and denoising budget:

- **LATCH** (Localized Acceleration with Tracked-Candidate Halting): separates two decisions that are often conflated in diffusion decoding. It commits locally stable positions while retaining a global answer-level stopping criterion, so a locally complete block does not by itself terminate the whole sequence.
- **Prophet**: uses the confidence gap between the top-two prediction candidates to decide whether to continue refinement or commit all remaining tokens in one step, before the nominal schedule ends.
- **SureLock**: detects positions whose posterior predictions have converged and locks them for later iterations, allowing their computation to be skipped while the remaining positions continue to denoise. It targets computational reuse after convergence rather than ranking all candidate transfers by risk.

Because these are reimplementations in a shared harness, they are controlled operating-point comparisons and need not reproduce the original systems' exact runtime or accuracy. These baselines cover local commitment, answer-aware prediction, and token locking. DJRS differs in three respects: (i) it combines four stability signals—confidence, margin, JSD, and a temporal prediction-change indicator—into a single risk score; (ii) it commits tokens individually in risk order rather than applying a single global stop decision; and (iii) it uses a shared per-step risk budget that grows with denoising progress.

3 METHOD

3.1 BACKGROUND: MASKED DIFFUSION DECODING

LLaDA starts from a fully masked sequence of length L and unmasks tokens over T steps. At step t, the model predicts a distribution $p^{(t)}(v \mid i)$ for each masked position i. A remasking strategy selects which positions to commit (replace mask with argmax) and which to keep masked. Low-confidence remasking (Nie et al., 2025) commits the highest-confidence tokens first, leaving uncertain positions masked for later steps. In the block-wise setting (Arriola et al., 2025), the sequence is divided into blocks of size B. Let T_b denote the denoising iterations allocated to active block b; the total decoding budget is the sum of the per-block allocations.

3.2 CONTEXT-CONDITIONED STABILITY SIGNALS

For each masked position i at step t, we compute four signals from the model's predictions.

Confidence. $c_i^{(t)} = p^{(t)}(v_1 \mid i)$, the top-1 probability.

Margin. $m_i^{(t)} = p^{(t)}(v_1 \mid i) - p^{(t)}(v_2 \mid i)$, the gap between the top two probabilities.

Jensen–Shannon divergence. $\mathrm{JSD}_i^{(t)} = \mathrm{JSD}\big(p^{(t-1)}(\cdot \mid i),\ p^{(t)}(\cdot \mid i)\big)$, measuring how much the distribution changed since the previous step.

Prediction change. $\pi_i^{(t)} = \mathbb{1}[\arg\max_v p^{(t)}(v \mid i) \neq \arg\max_v p^{(t-1)}(v \mid i)]$, indicating whether the top-1 prediction changed since the preceding step. This term assigns higher risk to positions whose predicted token has just changed.

3.3 FLIP PREDICTION ANALYSIS

A *flip* occurs when the top-1 prediction at position i changes between steps t and $t+1$:

$$\mathrm{flip}_i^{(t)} = \mathbb{1}\big[\arg\max_v p^{(t)}(v \mid i) \neq \arg\max_v p^{(t+1)}(v \mid i)\big]. \tag{1}$$

We examine associations with $\mathrm{flip}_i^{(t)}$ using a confidence-stratified trace design (Section 5.3). The saved diagnostic uses confidence and margin from $t-1$, JSD between $t-1$ and t, and only positions that remain masked at both t and $t+1$. It therefore differs from the decoder, which scores current-step commitment candidates. In these traces, even high-confidence tokens can flip. JSD provides complementary next-step information beyond confidence in some strata, motivating its inclusion in the DJRS risk score. We include the prediction-change indicator as a temporal heuristic that penalizes positions whose top-1 token just changed.

3.4 DETERMINISTIC JOINT-RISK SAMPLING

We describe the additive LLaDA policy here; the Dream adaptation is specified in Appendix F. At each iteration, DJRS preserves the standard low-confidence transfer set and selects additional positions using three operations:

1. **Score** every masked position in the active block:
$$r_i = w_c(1 - c_i) + w_m(1 - m_i) + w_j \mathrm{JSD}_i + w_p \pi_i. \tag{2}$$
Lower scores indicate greater estimated stability, not calibrated error probabilities.
2. **Rank and budget**: sort positions by increasing risk and retain the longest prefix whose cumulative score does not exceed
$$b(t) = b_{\min} + (b_{\max} - b_{\min}) \left(\frac{t}{\max(T_b - 1, 1)} \right)^2. \tag{3}$$
Here $t = 0, \ldots, T_b - 1$ is block-local and T_b is the effective block horizon. The budget is recomputed at each iteration.

3. **Filter and merge**: retain budget-selected positions satisfying $c_i \geq \theta_c$ and $m_i \geq \theta_m$, then commit their union with the standard transfer set.

Filtering follows budget selection. Standard-selected positions are not excluded from the risk ranking, so overlap and subsequently filtered positions can consume budget. The budget therefore constrains the selected risk prefix, not the complete union of commitments.

Design rationale. Confidence and margin describe the current prediction: confidence measures probability mass on the leading token, while margin measures its separation from the strongest alternative. JSD and prediction change describe temporal behavior. The distribution can shift without changing its most likely token, whereas a small shift near a tie can change the top prediction; the two temporal signals therefore capture different events. The cumulative budget couples commitment decisions: several low-risk positions may fit where fewer higher-risk positions would. For fixed endpoint budgets, quadratic growth allocates less budget than a linear schedule at intermediate progress, permitting fewer additional commitments early and more near block completion. This is a design choice, not evidence that quadratic growth is optimal.

Default hyperparameters. We use $w_c = 0.35$, $w_m = 0.35$, $w_j = 0.20$, $w_p = 0.10$, $b_{\min} = 0.50$, $b_{\max} = 4.0$, $\theta_c = 0.60$, and $\theta_m = 0.10$. JSD is computed over the full vocabulary; compressed top-5 traces are used only for diagnostics.

Interaction with standard remasking. At iteration t, standard remasking selects the highest-confidence $\min(q_t, |M_t|)$ masked positions, where q_t is the precomputed transfer quota and M_t is the remaining masked set in the active block. DJRS adds its risk-selected positions without removing standard transfers. If no additional position fits the budget or survives filtering, the standard schedule proceeds unchanged; there is no forced DJRS fallback. Once the block contains no masked positions, its remaining iterations are skipped, reducing NFE without example-level early exit. Quota construction, initialization, and tie handling are specified in Appendix F.

4 Experiments

4.1 Setup

Model. We use LLaDA-8B-Instruct (Nie et al., 2025) via the FastdLLM pipeline (Wu et al., 2025) with block-wise decoding (block size 32), prefix caching, and low-confidence remasking. For coding, we evaluate Dream-v0-Instruct-7B (Dream Team, 2025) on HumanEval. All experiments were run on a single NVIDIA A6000 GPU with 48 GB of memory.

Benchmarks. We evaluate on GSM8K (Cobbe et al., 2021) (1,319 test examples) and MATH-500 (Lightman et al., 2023) (500 test examples) under both 64- and 128-step denoising budgets. GSM8K tests grade-school arithmetic with short multi-step solutions, while MATH-500 covers competition-level problems spanning algebra, geometry, number theory, and precalculus. Together they stress different failure modes: GSM8K rewards precise step-by-step arithmetic, whereas MATH-500 rewards global planning over longer, more structurally diverse solutions where early errors are especially costly. For coding, we evaluate Dream-v0-Instruct-7B (Dream Team, 2025) on the 164-example HumanEval benchmark (Chen et al., 2021). The model differs between coding and mathematics, so these evaluations do not isolate task from model effects.

Baselines. For the primary LLaDA evaluation, we compare DJRS against three baselines (Section 2) and the Default LLaDA decoding at the same 64- or 128-step budget, without early exit. All methods use the same model, block size, and remasking strategy; only the commitment/stopping criterion differs.

Metrics. We report accuracy and the average number of backbone forward evaluations (NFE) per example, measured by a forward-pass hook during evaluation. In block-wise decoding, a forward evaluation is one backbone call for the active block; lower NFE therefore reflects fewer denoising calls, not merely fewer tokens committed.

4.2 GSM8K RESULTS

Table 1 shows the results on GSM8K at both budgets. At 64 steps, DJRS obtains 63.8% accuracy at 45.6 NFE, a 29% reduction in NFE relative to the Default 64-step setting (64.4% at 64.0 NFE) and a 0.6-percentage-point decrease in accuracy. DJRS has no example-level early exits. Its NFE reduction instead comes from completing active blocks once all their positions have been selected for transfer. This mechanism differs from methods which reduce NFE by terminating decoding early.

Table 1: GSM8K results with LLaDA-8B-Instruct at two denoising budgets. NFE is average forward evaluations; ΔAcc is relative to Default at the same step budget.

Steps	Method	Accuracy (%)	NFE	ΔAcc	Early Exit (%)
64	Default	64.4	64.0	—	0.0
64	DJRS (ours)	63.8	45.6	−0.6	0.0
64	LATCH	63.2	46.9	−1.2	1.2
64	Prophet	64.4	64.0	0.0	0.1
64	SureLock	49.8	60.4	−14.6	0.0
128	Default	68.8	128.0	—	0.0
128	DJRS (ours)	68.5	57.0	−0.3	0.0
128	LATCH	69.4	66.8	+0.6	0.0
128	Prophet	69.3	128.0	+0.5	0.0
128	SureLock	43.1	105.3	−25.8	0.0

Comparison. At 64 steps, DJRS changes accuracy from 64.4% to 63.8% while reducing NFE from 64.0 to 45.6 (about 29%). It and LATCH have similar compute use, with DJRS at the higher accuracy point estimate; Prophet retains Default accuracy but does not reduce NFE, while SureLock has much lower accuracy at 60.4 NFE. At 128 steps, DJRS changes accuracy from 68.8% to 68.5% and reduces NFE from 128.0 to 57.0 (about 55%). LATCH and Prophet have slightly higher reported accuracy, at 66.8 and 128.0 NFE, respectively. The GSM8K signal-composition sensitivity analysis is reported in Appendix D.

4.3 MATH-500 RESULTS

Table 2 shows the results on MATH-500 at both budgets. In the 64-step condition, DJRS increases accuracy by 3.6 percentage points while reducing NFE by 22%. At 128 steps, it reduces NFE by 46% with a 1.8-percentage-point accuracy decrease.

Table 2: MATH-500 results with LLaDA-8B-Instruct at two denoising budgets. ΔAcc is relative to Default at the same step budget.

Steps	Method	Accuracy (%)	NFE	ΔAcc	Early Exit (%)
64	Default	27.8	64.0	—	0.0
64	DJRS (ours)	31.4	49.7	+3.6	0.0
64	LATCH	29.2	51.2	+1.4	0.6
64	Prophet	13.4	46.3	−14.4	37.6
64	SureLock	22.4	61.0	−5.4	0.0
128	Default	35.4	128.0	—	0.0
128	DJRS (ours)	33.6	69.6	−1.8	0.0
128	LATCH	36.8	80.1	+1.4	0.0
128	Prophet	10.6	68.3	−24.8	37.6
128	SureLock	20.0	110.5	−15.4	0.0

Comparison. At 64 steps, DJRS reaches 31.4% accuracy at 49.7 NFE, compared with LATCH's 29.2% at 51.2 NFE. At 128 steps, DJRS remains below Default (33.6% versus 35.4%) and LATCH (36.8% at 80.1 NFE), while using fewer evaluations.

4.4 HUMANEVAL RESULTS

Table 3 reports execution-based HumanEval (Chen et al., 2021) results on 164 examples using Dream-v0-Instruct-7B (Dream Team, 2025) at both denoising budgets. For Dream, we adapt DJRS to its native scheduler: the risk budget can increase the scheduled transfer count, and positions are ranked by risk. This preserves the scheduler's minimum transfer count while allowing risk-based acceleration; details appear in Appendix F. Our Dream–HumanEval integration does not apply Prophet's early-commit trigger because it lacks a task-specific answer-span rule; this configuration therefore follows the default schedule. Its unchanged NFE reflects this integration choice, not a demonstrated limitation of the original method.

Table 3: HumanEval results with Dream-v0-Instruct-7B on 164 examples. NFE is average forward evaluations; ΔAcc is relative to Default at the same step budget. Early-exit accounting was not recorded. Differences are computed before rounding the displayed scores.

Steps	Method	pass@1 (%)	NFE	ΔAcc	Early Exit (%)
64	Default	32.9	64.0	—	—
64	DJRS	38.4	39.8	+5.5	—
64	LATCH	35.4	41.9	+2.5	—
64	Prophet	32.9	64.0	0.0	—
64	SureLock	32.3	45.5	-0.6	—
128	Default	53.7	128.0	—	—
128	DJRS	52.4	49.6	-1.2	—
128	LATCH	52.4	77.9	-1.2	—
128	Prophet	53.7	128.0	0.0	—
128	SureLock	42.7	70.9	-11.0	—

Comparison. At 64 steps, the reported DJRS run changes pass@1 from 32.9% to 38.4% while reducing NFE from 64.0 to 39.8; LATCH reports 35.4% at 41.9 NFE and SureLock reports 32.3% at 45.5 NFE. At 128 steps, DJRS reduces NFE from 128.0 to 49.6, but pass@1 decreases by 1.2 percentage points to 52.4%; LATCH reaches the same reported accuracy at 77.9 NFE, while Default decoding is 53.7%.

5 ANALYSIS

5.1 CROSS-DATASET METHOD COMPARISON

Across datasets, DJRS occupies a task- and budget-dependent point on the accuracy–compute trade-off. Its most favorable results occur at 64 steps, where it reduces NFE with comparable or better quality. At 128 steps, Default or LATCH can be more accurate; for example, LATCH reaches 36.8% on MATH-500 versus DJRS at 33.6%.

One possible explanation for accuracy gains is earlier access to useful context. Correct, informative commitments can help remaining masked positions resolve dependencies in subsequent forward passes. Incorrect commitments can instead propagate errors; committed positions are fixed, so only positions that remain masked can be refined. The MATH-500 and HumanEval gains at 64 steps and decreases relative to Default at 128 steps are consistent with this trade-off, but do not establish its cause. More committed context alone does not guarantee better predictions (Appendix C).

5.2 WHERE THE NFE REDUCTION COMES FROM

Figure 2 shows how the compute reduction is distributed across examples. Default uses the full 128-step budget, whereas DJRS completes blocks at variable and generally lower NFE. This is consistent with early block completion rather than cheaper forward passes.

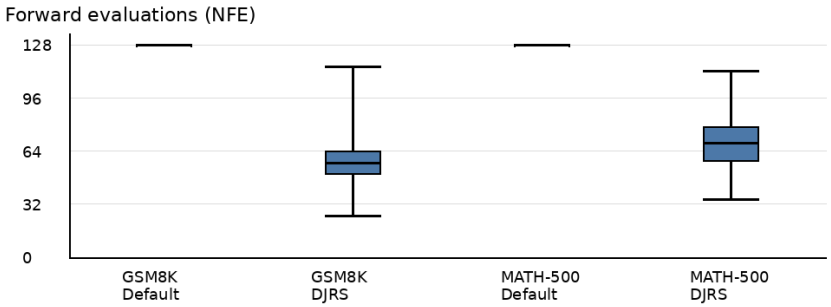

Figure 2: Per-example NFE distributions in the logged 128-step runs. Default uses the full nominal budget, whereas DJRS completes blocks at different and generally lower NFE values. Boxes show the interquartile range and the center line marks the median.

5.3 Flip Prediction Analysis

To motivate the DJRS signal design, we analyze flip prediction using a confidence-stratified trace design. We collect full denoising trajectories on 150 GSM8K examples across block sizes $\{16, 32, 64, 128\}$, using the first 75 examples to construct the calibration reservoirs and the next 75 examples for evaluation. This defines the separated calibration/evaluation protocol used below. Additional logging details appear in Appendix A, and the separated analysis design is described in Appendix B.

Method. The saved analysis pairs confidence and margin at $t-1$ and JSD between $t-1$ and t with flips from t to $t+1$. It includes only positions masked at both t and $t+1$, excluding positions committed during that transition. These are lagged-signal associations in a selected population, not a direct validation of current-step commitment decisions. For each (confidence bin $\times$ signal quintile) cell, we measure the flip rate. This design examines the signal's association with flips within confidence strata: we compare tokens with similar confidence but different signal values.

Results. Table 4 reports evaluation flip rates in the lowest and highest signal quintiles within five fixed-width confidence bins for block size 64. Signal quintiles are ordered by increasing value using calibration ranks. These comparisons describe within-bin associations, not isolated effects.

Table 4: Evaluation flip rates (%) for block size 64. Each cell gives the lowest/highest signal-quintile rates (Q1/Q5), not their ratio. Confidence bins have width 0.2. Diagnostic JSD uses logged top-5 probabilities plus residual mass.

Signal	$[0, .2)$	$[.2, .4)$	$[.4, .6)$	$[.6, .8)$	$[.8, 1]$
Margin	39.71/16.21	30.93/7.74	16.96/2.22	5.39/0.50	0.79/0.00
JSD	22.70/35.00	11.64/22.80	4.77/12.49	0.87/5.69	0.00/0.63

The trace analysis asks whether individual signals rank near-term commitment instability. Across all four schedules, safer margin-conditioned strata have lower future-flip frequency; for block size 64, rates decrease from 39.71% to 16.21% across margin quintiles in the lowest confidence bin, and from 0.79% to zero observed flips in the highest. This makes margin a useful component of a commitment-risk score. JSD-conditioned rates are less monotonic in isolation, so JSD is best interpreted as complementary information rather than a standalone calibrated estimator. Because positions within a trajectory are related, these rates are descriptive rather than independent-sample effect sizes.

Together, the diagnostics motivate component-level risk ranking; they do not calibrate the complete weighted DJRS score or guarantee final-answer accuracy. An example of per-token scores, eligibility filtering, and risk ranking is provided in Appendix E.

5.4 MATH-500 SIGNAL ABLATION

Table 5 reports a 64-step MATH-500 ablation with cumulative signals. The full score achieves the highest observed accuracy among these runs, but adding a signal also changes NFE and token selection. These results therefore describe operating points rather than isolated causal contributions.

Table 5: MATH-500 DJRS signal ablation at 64 denoising steps.

Signals used	Accuracy (%)	NFE
Confidence only	30.6	45.3
Confidence + margin	31.0	46.1
Confidence + margin + JSD	29.0	49.9
All four signals (DJRS)	31.4	49.7

6 LIMITATIONS

DJRS requires prediction tracking for JSD and temporal change, which adds memory and bookkeeping overhead. Its weights and budget may need task- and model-specific calibration. It is designed for sequential block-wise decoding, and extending it to fully parallel unmasking requires a different schedule. The LLaDA mathematics and Dream HumanEval evaluations do not isolate model from task effects or establish robustness across domains and hardware. Finally, NFE may differ from wall-clock latency once bookkeeping is included.

7 CONCLUSION

We proposed Deterministic Joint-Risk Sampling (DJRS), which ranks masked positions using a multi-signal risk score and a progress-dependent budget. Across LLaDA GSM8K and MATH-500 and separate Dream HumanEval evaluations, DJRS reduces NFE at useful operating points, including a 29% reduction on GSM8K at 64 steps with a 0.6-percentage-point accuracy decrease and a 22% reduction with a 3.6-percentage-point MATH-500 gain. The 128-step results show that this trade-off depends on task and budget: Default or LATCH can achieve higher accuracy when further refinement is useful. The flip analysis supports risk ranking as a practical heuristic, not an unconditional accuracy guarantee.

AI USE STATEMENT

Generative AI tools assisted with code generation, experiment scripting, language editing, and LaTeX formatting. The authors reviewed the resulting work and take full responsibility for the final manuscript, code, and reported results.

REPRODUCIBILITY STATEMENT

Code and trace data are not currently distributable. To support reproducibility, we document the public models and datasets, evaluation settings, and key method hyperparameters; selected aggregate results and analysis summaries are included in the paper. We use the public LLaDA-8B-Instruct model via the FastdLLM pipeline. Key hyperparameters: 64 and 128 denoising steps (GSM8K, MATH-500), block size 32, low-confidence remasking, prefix caching. DJRS weights: $w_c = 0.35$, $w_m = 0.35$, $w_j = 0.20$, $w_p = 0.10$; budget $[0.5, 4.0]$; confidence threshold 0.60; margin threshold 0.10. Trace collection uses 150 GSM8K examples across block sizes $\{16, 32, 64, 128\}$ with top-5 token logging for storage; the decoder computes JSD from the full vocabulary softmax. The separated flip-prediction analysis uses the first 75 examples for calibration and the next 75 examples for evaluation; calibration observations are not reused for evaluation. See Sections 3–4 for full details.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *arXiv preprint arXiv:2107.03006*, 2021.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374. Dataset: https://huggingface.co/datasets/openai/openai_humaneval.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dream Team. Dream-v0-Instruct-7B. https://huggingface.co/Dream-org/Dream-v0-Instruct-7B, 2025.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. *Advances in Neural Information Processing Systems*, 2019.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1904.09751.

Chia-Ming Lee, Ming-Ching Chang, Xin Li, Yu-Lun Liu, and Chih-Chung Hsu. Where and when to commit: Candidate-aware decoding for diffusion language models. *arXiv preprint arXiv:2607.28166v1*, 2026. URL https://arxiv.org/abs/2607.28166v1.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Soroush Vosoughi, and Shiwei Liu. Diffusion language models know the answer before decoding. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2508.19982.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. *arXiv preprint arXiv:2206.00927*, 2022.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2502.09992.

Daisuke Oba, Danushka Bollegala, Masahiro Kaneko, and Naoaki Okazaki. Stopping computation for converged tokens in masked Diffusion-LM decoding. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.06412.

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. *Advances in Neural Information Processing Systems*, 2024.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2207.07061.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.02502.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, 2019. URL https://arxiv.org/abs/1902.03249.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025. URL https://arxiv.org/abs/2505.22618.

A Trace Collection Details

We collect full denoising trajectories using an isolated trace collector that wraps the MDLMSampler without modifying the evaluation path. For each example and block size, we log:

- At each step: the full token canvas, mask status, top-5 token IDs and probabilities for each masked position, and the model's predictions.
- Per-position: confidence, margin, JSD (vs. previous step), and spatial fill.
- Per-trajectory: example ID, block size, final answer, ground-truth answer, and correctness.

The trace collector runs on a single A6000 GPU with 150 examples taking approximately 30 minutes per block size.

B Controlled Analysis Design

The separated analysis uses a two-pass development/evaluation design: the first 75 examples construct confidence-matched signal reservoirs, while the next 75 examples are evaluated without recalibration. For each evaluation observation, signal ranks are computed using only the development reservoir. Under this separated protocol, margin-conditioned strata show a stable reduction in future-flip frequency as risk decreases across schedules, whereas JSD is less monotonic when considered alone. We report these results as next-step flip prediction: a flip is a change in top-1 prediction between t and $t+1$, not a direct measure of eventual token correctness or downstream answer accuracy. These results motivate stability-based ranking for the analyzed population, but do not establish an unconditional accuracy guarantee. Confidence and margin are taken from the preceding step, whereas JSD spans the preceding and current steps. Only positions masked at both the current and next steps are included, so the diagnostic is not an evaluation of all commitment candidates.

C Conditional Risk Interpretation

We state a conditional bound for an idealized calibrated extension of DJRS. Let $\mathcal{F}_t$ contain the decoding history and all information used to select positions at iteration t, including scores and the current canvas. The selected set S_t and budget B_t must be measurable with respect to $\mathcal{F}_t$. For each candidate, define $U_i^{(t)}$ as an indicator of a specified future instability event: disagreement between its current top-1 prediction and a later prediction under a fixed reference continuation that keeps this position available for prediction. The continuation and comparison horizon must be specified before calibration. Such disagreement does not by itself imply an incorrect token or an incorrect final answer.

Assumption and bound. Suppose nonnegative, $\mathcal{F}_t$-measurable bounds $q_i^{(t)}$ satisfy

$$\mathbb{E}[U_i^{(t)} \mid \mathcal{F}_t] \le q_i^{(t)}, \qquad \sum_{i \in S_t} q_i^{(t)} \le B_t \quad \text{almost surely.}$$

Then

$$\mathbb{E}\left[\sum_{i \in S_t} U_i^{(t)} \mid \mathcal{F}_t\right] = \sum_i \mathbb{1}\{i \in S_t\}\mathbb{E}[U_i^{(t)} \mid \mathcal{F}_t] \le \sum_{i \in S_t} q_i^{(t)} \le B_t.$$

The equality follows because selection is known given $\mathcal{F}_t$. Taking expectations and summing over a finite decoding horizon gives

$$\mathbb{E}\left[\sum_t \sum_{i \in S_t} U_i^{(t)}\right] \le \mathbb{E}\left[\sum_t B_t\right].$$

No independence between positions is required. For deterministic budgets the right-hand side is $\sum_t B_t$; after stopping, take $S_t = \emptyset$ and $B_t = 0$. A score-only calibration condition such as $\Pr(U_i^{(t)} = 1 \mid r_i^{(t)}) \le g(r_i^{(t)})$ is insufficient unless it also remains valid conditional on the information used for adaptive selection.

Scope. This bounds expected instability events, not answer error. Only with the additional assumption that an answer-change event A relative to a specified reference decoder implies at least one counted instability event can one infer $\Pr(A) \le \min\{1, \mathbb{E}[\sum_t B_t]\}$ by the union bound. This implication is not established here. The implemented DJRS budgets raw weighted scores and retains standard remasking transfers in LLaDA. Consequently, the bound does not apply to the reported decoder: its assumptions would need to cover every transfer being bounded, including standard transfers if they are included in S_t.

Committed context and accuracy. Correct, informative commitments can supply context that improves predictions at remaining masked positions. Incorrect commitments can propagate errors. Neither increased confidence nor a larger number of committed tokens guarantees correctness. This explanation is consistent with the observed differences across tasks and budgets, but is not a consequence of the bound. The trace analysis concerns next-step flips and does not establish either the conditional calibration assumption or an improvement in final-answer accuracy.

D GSM8K Operating-Point Sensitivity

The GSM8K signal ablation is included as a sensitivity analysis rather than as evidence that signals are uniformly additive. All runs use the 64-step budget and the same evaluation setup. Confidence-only selection is Pareto-superior at this operating point, while adding signals makes the transfer policy more conservative and increases NFE. The result motivates task- and schedule-specific calibration of the signal weights.

Table 6: GSM8K signal-composition sensitivity at 64 nominal steps.

Configuration	Accuracy (%)	NFE
Default	64.4	64.0
Confidence only	65.0	39.0
Confidence + margin	64.1	39.6
Confidence + margin + JSD	64.1	45.5
All four signals	63.8	45.6

E Token-Level DJRS Example

This illustration shows two eligible positions (79 and 87) and two filtered positions (92 and 93) from a single GSM8K trace. Positions index the sequence; token IDs identify vocabulary entries.

Table 7: Token-level DJRS example from a GSM8K trace.

Position	Token ID	Confidence	Margin	JSD	Change	Risk	Status
79	19149	0.781	0.675	0.309	1	0.352	Eligible
87	1261	0.746	0.695	0.227	0	0.241	Eligible
92	16	0.531	0.348	0.200	1	0.532	Filtered
93	13	0.590	0.440	0.237	0	0.387	Filtered

F Decoder Implementation Details

LLaDA quotas and selection. The total nominal budget T is divided across blocks, with $\lceil T/N_b \rceil$ iterations initially allocated to each of N_b blocks. The standard scheduler precomputes integer transfer quotas from the initial mask count using rounded expected transfers, removes zero-quota iterations, and defines the effective horizon T_b. At iteration t, standard remasking selects the highest-confidence $\min(q_t, |M_t|)$ positions, where q_t is the precomputed quota and M_t is the remaining masked set. DJRS independently sorts all positions in M_t by risk, takes the cumulative-budget prefix, filters it by confidence and margin, and unions the survivors with the standard set. Rejected or overlapping positions do not release budget for additional candidates. An empty prefix or an empty filtered set leaves standard decoding unchanged. Remaining masked positions continue under subsequent quotas until block completion.

Initialization and ties. Prediction history resets at each block. When previous distributions or predictions are unavailable or have incompatible tensor shapes, the corresponding JSD-risk or prediction-change term is initialized to one. Otherwise JSD uses full-vocabulary probabilities and natural logarithms, with its risk term clipped to $[0, 1]$. Sorting uses the framework's default ordering; no stable positional tie-break is explicitly enforced, so tied scores need not have identical ordering across platforms.

Dream adaptation and rationale. The prefix-cached Dream decoder first performs its native cache-building pass and commits the first position of the active block. Subsequent iterations retain its scheduled minimum transfer count k_{sched}. DJRS computes the same weighted risk signals and the number k_{budget} admitted by their sorted cumulative sum, then selects the lowest-risk positions using

$$k = \min\{|M|, \max(k_{\text{sched}}, k_{\text{budget}})\}.$$

The native quota is $\lfloor |M|(1 - s/t) \rfloor$ for consecutive diffusion times t, s, with all remaining positions selected at the final scheduled transfer. The quadratic budget uses progress $(j-1)/\max(H-1, 1)$, where j is the inner-loop index starting at one and H is the allocated block horizon. Prediction history resets per block and missing-history risk terms are initialized to one. Unlike LLaDA, this adapter does not apply confidence/margin eligibility thresholds or preserve a separate confidence-selected transfer set. The rationale is to retain Dream's native minimum transfer schedule while using risk to determine selection and additional capacity. This is a backbone-specific implementation choice, not evidence that one integration is superior to the other.