Inference-Time Search in Discrete Diffusion Language Models, Guided by Internal-State Signals

Anonymous authors
Paper under double-blind review

Abstract

Inference-time search for diffusion language models assumes that a partially decoded state reveals, early and cheaply, whether a trajectory is worth continuing, and that the model's own internal state is the natural place to read it. We test that assumption on three testbeds spanning four orders of magnitude — two masked-diffusion Sudoku models (0.54M and 6.6M parameters) with an exact symbolic oracle, and LLaDA-8B on GSM8K — and find that internal-state signals track the model's *belief*, not the partial state's *determinacy*. We make this precise by splitting the variance of prefix value into a between-prompt part, which only allocation can use, and a within-prompt part, which is all that selection can use. Pooled AUROC, the standard metric, is dominated by the former and so measures difficulty: a difficulty-only signal scores 0.78 pooled and 0.50 within-prompt. Even within-prompt AUROC fails to price a selector: on the small model, confidence ranks branches above chance (0.64), yet choosing its top branch cuts accuracy from 0.69 to 0.36. We introduce a cheap estimator of within-prompt variance from branched rollouts and show it is set by the sampler — zero at temperature 0.3, and at 0.9 half or more of the remaining outcome variance on LLaDA-8B by $r = 0.5$, where a perfect selector would add 15 points. Model confidence recovers none of it at 8B; a task-structural oracle recovers most of it on Sudoku. We recommend reporting any verifier by its realized selection value, at the sampler's operating point.

1 Introduction

A masked diffusion language model writes an answer by repeatedly unmasking positions of an all-masked canvas, so generation passes through a sequence of partial states. That structure is what makes inference-time search attractive for these models: sample several trajectories, judge the partial states, and spend the remaining compute on the promising ones. A rapid 2026 line of methods does exactly this, and several propose that the judge be the model's own internal state, which is computed on every denoising step anyway and is therefore nearly free to read.

Whether that can work depends on two questions that the literature has not separated. First, do partial states of *one* prompt actually differ in their prospects? If every trajectory of a problem is equally likely to succeed, there is nothing to select. Second, when they do differ, do internal signals see the difference? Chiu & Liu (2026) show that on LLaDA-8B and Dream-7B the signal available at step 0 largely reflects prompt difficulty, and that denoising then adds 0.08–0.11 AUC. Like every figure in this literature, that AUC is pooled across prompts — and a selector never compares one prompt against another.

We separate the two questions and answer each on three testbeds, a 0.54M- and a 6.6M-parameter masked-diffusion Sudoku model with an exact symbolic oracle, and LLaDA-8B-Instruct on GSM8K. Our contributions are:

- **A decomposition with an operational reading.** The variance of prefix value splits into a between-prompt part, which only an *allocation* rule can exploit, and a

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.

within-prompt part, which is all a *selection* rule can exploit (§3). Pooled AUROC is dominated by the first and therefore measures difficulty; this reproduces on all three testbeds.

- **Realized selection value as the metric.** Within-prompt AUROC is necessary but not sufficient: a signal can rank branches above chance on average and still put wrong ones on top. On the small Sudoku model, confidence scores 0.64 within-prompt yet selecting by it costs 32 points; on a twelve-times-larger model of the same task, and at 8B, it is inert.
- **A cheap estimator of the within-prompt component.** Tails branched from a shared partial state have outcome variance $\mathrm{Var}(Y \mid x_r)$ by construction, so the quantity search can spend falls out of a branching experiment, with two conservation laws as checks.
- **The measurement.** Spendable variance is set by the sampler: it is exactly zero at temperature 0.3 and grows steadily with temperature. At 0.9 on LLaDA-8B it is half or more of the remaining outcome variance by $r = 0.5$, worth +15 points to a perfect selector — and model confidence captures none of it, while a task-structural oracle captures most of it on Sudoku.
- **Derived methods, priced honestly.** Difficulty-aware allocation gains 1.5–1.9 points on Sudoku; sharing the early phase of generation loses on LLaDA, exactly as our identity predicts from the measured variance; and with a perfect selector the efficient frontier favours few branches chosen late, not wide early pruning.

We claim no novelty for the prompt-difficulty observation (Chiu & Liu, 2026), for within-question evaluation of self-consistency confidence (Taubenfeld et al., 2025), or for the marginal-value allocation function (Damani et al., 2024).

2 Background

A masked (absorbing-state) diffusion LM (Sahoo et al., 2024; Shi et al., 2024) decodes an answer of L positions over a sequence of steps, each committing some masked positions. We write x_r for the state in which a fraction r of answer positions is committed. Ou et al. (2025) show the concrete score factorises into a time-independent conditional on clean data, which fixes per-position marginals but not the joint, and correctness is a function of the joint; training choices also shape what partial states encode (Train for the Worst, 2025). Which positions are committed is chosen by a sampler; the standard choice, used by LLaDA (Nie et al., 2025), commits the positions the model is most confident about, with a temperature governing the randomness of the tokens it writes. Published correctness signals for LLaDA-8B on GSM8K range from output confidence (0.753 AUROC) to re-noising stability (0.893) (Li et al., 2026); HIVE (HIVE, 2026) reaches 0.9236 on Dream-7B. The cheap ones are only available once decoding ends, the informative ones cost at least a further generation (Figure 4), and all are pooled across prompts.

3 Framework

Prefix value and its decomposition. Fix a prompt Q and let $Y \in \{0, 1\}$ indicate that decoding ends correct. The *prefix value* of a state is $v(x_r) = \Pr[Y = 1 \mid x_r]$, the probability taken over the remaining stochastic steps. Because v is a conditional expectation of Y under the model's own sampling, $\{v(x_r)\}_r$ is a martingale along a trajectory, so $\mathbb{E}[v(x_r)]$ is constant in r — an identity, and the first check we apply to every estimate. By the law of total variance,

$$\underbrace{\mathrm{Var}[v(x_r)]}_{H(r)} = \underbrace{\mathrm{Var}_Q\big[\mathbb{E}[v(x_r) \mid Q]\big]}_{H_{\text{between}}} + \underbrace{\mathbb{E}_Q\big[\mathrm{Var}[v(x_r) \mid Q]\big]}_{H_{\text{within}}(r)}. \tag{1}$$

The martingale property gives $\mathbb{E}[v(x_r) \mid Q] = p_Q$, the prompt's success rate, so H_{between} does not depend on r; and at $r = 0$ all trajectories of a prompt share one state, so $H_{\text{within}}(0) = 0$ exactly — our second check.

Proposition 1 (Which heterogeneity funds which algorithm)**.** *(i) A* selection *rule — best-of-N, pruning, or any rule choosing among trajectories of one prompt — can gain over a random choice only through H_{within}; if it is zero, every selection rule attains the accuracy of a random pick. (ii) An* allocation *rule, choosing how many trajectories each prompt receives, can improve on the uniform allocation only through H_{between}; if it is zero, the uniform allocation is optimal.*

Proofs are in Appendix B. The point is operational: the two terms of (1) are consumed by different algorithms, so "is this signal useful?" is two questions, not one.

Pricing a verifier. A readout $s(x_r)$ is usually scored by pooled AUROC — correct and incorrect generations ranked against one another across the evaluation set. That rewards ranking an easy prompt above a hard one, a comparison no selector ever faces, so it measures H_{between}: a signal that is a function of the prompt alone scores high pooled and exactly 0.500 within-prompt. The within-prompt AUROC, over discordant pairs from the same prompt, is the natural correction (Taubenfeld et al., 2025). It is still not what a selector consumes. AUROC averages over *all* pairs, while an argmax selector depends only on which candidate sits at the *top*. A signal can order most pairs correctly and still place confidently wrong candidates first — which is what happens when a model grows confident about whatever it has committed, its errors included. We therefore score a readout by its **realized selection value**: pick the candidate it ranks highest at fraction r, complete it, and report the accuracy of the result against a random pick (the no-search baseline, equal to base accuracy by the martingale) and against a perfect selector. Only the last number prices a verifier for search.

A branching estimator for the spendable variance. Let $G(r) = \mathbb{E}[\mathrm{Var}(Y \mid x_r)] = \mathbb{E}[v(x_r)(1 - v(x_r))]$ be the outcome variance still available at r. Conditioning on x_r gives $\mathrm{Var}(Y) = H(r) + G(r)$, and since H_{between} is constant in r and $H_{\text{within}}(0) = 0$,

$$G(0) - G(r) \;=\; H_{\text{within}}(r). \tag{2}$$

Proposition 2 (Branching)**.** *Generating one shared prefix to $r^\star$ and then N tails costs $r^\star + N(1 - r^\star)$ full generations against N for independent samples, and the tails forgo exactly $H_{\text{within}}(r^\star)$ of the outcome variance available to independent sampling.*

Equation (2) is also a measurement device. Tails branched from a shared x_r have outcome variance $\mathrm{Var}(Y \mid x_r)$ by construction, so a branch arm estimates $G(r)$, an independent arm estimates $G(0)$, and their difference is $H_{\text{within}}(r)$ — with no re-completion from stored states. Because $H(r)$ is the variance of a martingale it is non-decreasing, so $G(r)$ must be non-increasing; we check this on every estimate. All variances are debiased for the $v(1-v)/K$ inflation of a K-sample estimate.

Allocation. If prompt i has success rate p_i and receives K_i trajectories, the value of one more under a perfect end-selector is

$$\Delta_i(K) = p_i(1 - p_i)^K, \tag{3}$$

the function and greedy rule of Damani et al. (2024).

Proposition 3 (Where compute belongs)**.** *$\Delta(\cdot, K)$ is maximised at $p^\star = 1/(K+1)$: the marginal trajectory is worth most on prompts of intermediate difficulty, and a rule that directs compute to the hardest prompts is suboptimal. Greedy allocation on Δ is exactly optimal.*

The schedule is therefore fixed by a difficulty estimate available before decoding starts, not tuned. Proposition 1(ii) says what it can gain: nothing without H_{between}.

4 Testbeds and protocol

Sudoku-S: a 0.54M-parameter masked diffusion model on 6×6 Sudoku, 50 held-out puzzles, 32 branches per puzzle and 32 completions per state, with temperature $T \in \{0.3, 0.5, 0.6, 0.7, 0.9\}$ and 1, 2 or 4 cells committed per step. **Sudoku-M**: a 6.6M-parameter model on the same task (8 layers, $d = 256$, 26k steps), solving 27.6% of puzzles at 24 givens,

Table 1: Pooled against within-prompt AUROC. The first row is a signal constructed to depend on the prompt alone. On Sudoku-S the signal is the prefix value $v(x_0)$ itself — a *perfect* difficulty estimate — and on the other rows it is model confidence over masked positions. Pooled AUROC is high wherever difficulty varies; within-prompt AUROC, the part a selector can use, sits at chance. On LLaDA, 970 within-prompt pairs; no checkpoint's within-prompt value clears its permutation null (largest $|z| = 1.71$).

testbed	signal	r	pooled	within-prompt	null
constructed	prompt-only	—	0.778	**0.500**	—
Sudoku-S (0.54M)	prefix value	0	0.851	**0.492**	0.500 ± 0.027
Sudoku-M (6.6M)	confidence	0	0.768	**0.500**	exact
Sudoku-M (6.6M)	confidence	0.75	0.678	0.524	—
LLaDA-8B	confidence	0	0.702	**0.500**	0.500 ± 0.000
LLaDA-8B	confidence	mean, 8 r	0.650	0.514	max $\lvert z\rvert = 1.71$

sampled at $T = 1.0$. Both admit an exact symbolic oracle: whether a partial grid remains completable. **LLaDA-8B-Instruct** on GSM8K at $T = 0.9$ with 256 denoising steps and block length 32 (accuracy 0.768): 260 prompts × 8 independent trajectories with model confidence recorded at every step, and a branch sweep of 120 prompts with $N = 8$ tails at $r^\star \in \{0.5, 0.75\}$. Temperature must be positive: at $T = 0$ every trajectory of a prompt is identical and the within-prompt question is vacuous. In all three testbeds r is the fraction of answer positions committed. Every interval is a bootstrap over *prompts*, paired where two arms share prompts; probes use folds grouped by prompt; within-prompt AUROCs are compared against a null that permutes labels inside each prompt (Appendix E).

5 Results

5.1 Pooled AUROC measures difficulty, on every testbed

Table 1 shows the gap is not an artefact of one model. At $r = 0$ every trajectory of a prompt shares one state, so the within-prompt value is 0.500 by construction, and it is — on the 8B model with null standard deviation exactly zero. The sharpest case is Sudoku-S at $T = 0.3$, where the sampler is effectively deterministic: pooled AUROC of the prefix value is 1.000 at every r while the spendable variance is exactly 0. A verifier evaluated there would report a perfect score and be worth nothing, because there are no distinct branches to choose between. On Sudoku-M (full decomposition in Table 3) the two readings even move in opposite directions: pooled falls from 0.768 to 0.678 as decoding proceeds while within-prompt rises from 0.500 to 0.524. The pooled decline is what the literature reads as a degrading signal; it is the decay of a difficulty component that no selector could use, as the model grows confident about everything it has committed.

5.2 The spendable variance is real, set by the sampler, and large at 8B

Figure 1 shows the quantity a selector can spend is a property of the *sampler*. On Sudoku-S, H_{within} at $r = 0.95$ is 0.000, 0.003, 0.010, 0.031 and 0.094 for $T = 0.3$, 0.5, 0.6, 0.7 and 0.9, while H_{between} stays between 0.12 and 0.17; the spendable share of value variance goes from 0% to 43%. Both ends of the axis are degenerate for opposite reasons. Low temperature collapses the branches, so there is nothing to select; high temperature creates branches but leaves the outcome undetermined early. The randomness that produces branches worth selecting among is the same randomness that stops early states from settling the answer. The model matters too: at an even *higher* temperature ($T = 1.0$), Sudoku-M's spendable variance is below 10^{-3} at $r = 0.5$, against roughly 0.05 for Sudoku-S at $T = 0.9$ — so it must be measured at the operating point, never assumed or carried across models.

On LLaDA-8B we measure it with the branching estimator. At $T = 0.9$, $G(0) = 0.068$ and

$$H_{\text{within}}(0.5) = 0.035 \;\; [0.015,\, 0.054] \;\; (51\% \text{ of } G(0)),$$
$$H_{\text{within}}(0.75) = 0.050 \;\; [0.032,\, 0.068] \;\; (73\% \text{ of } G(0)),$$

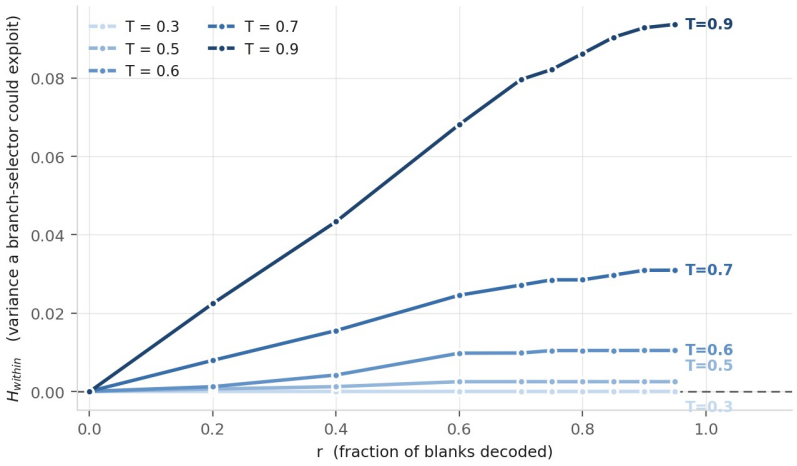


Figure 1: Spendable variance $H_{\text{within}}(r)$ on Sudoku-S against sampler temperature (one cell committed per step, 50 puzzles, 32 branches, debiased for finite-sample noise). At $T = 0.3$ the sampler is deterministic: no distinct branches exist, and $H_{\text{within}} = 0$ at every r.

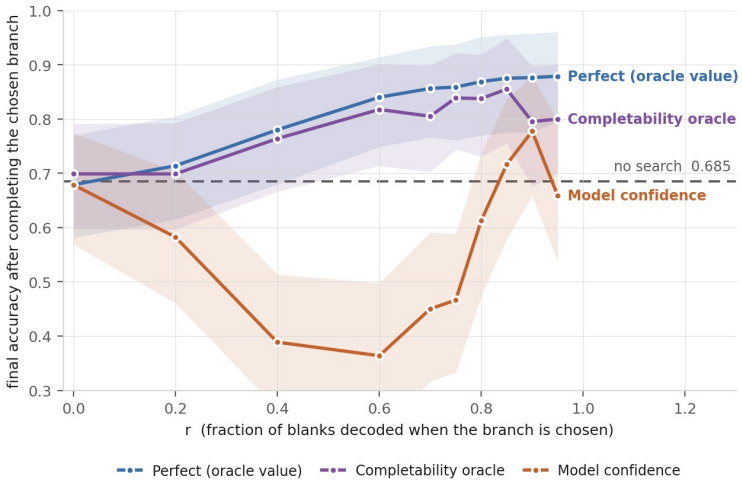


Figure 2: Realized selection value on Sudoku-S ($T = 0.9$, one cell per step, 50 puzzles, 32 branches): choose one branch at fraction r by the named rule, complete it, and score the result on a held-out half of the completions. Bands are 95% bootstrap intervals over prompts. Dashed line: no search.

both intervals clear of zero, with $G(r)$ non-increasing as the martingale requires (Figure 3a). Half or more of the remaining outcome variance is *within* prompt by the time the answer is half written: trajectories of one GSM8K problem genuinely diverge, well before decoding ends. What that is worth is direct to measure. Among eight trajectories per problem, a perfect selector reaches 0.919 against 0.768 for a random pick — a prize of 15 points.

5.3 Model confidence does not capture it

Figures 2 and 3 and Table 2 give the central result. On Sudoku-S, selecting the branch the model is most confident about is *worse than not selecting at all*: at $r = 0.6$ it yields 0.364 against a no-search 0.685, although the same confidence has a within-prompt AUROC of 0.640, comfortably above chance. The two are not in tension. AUROC averages over

Table 2: Realized selection value: accuracy after choosing one candidate by each rule and completing it. Sudoku-S and Sudoku-M select among 32 branches at $T = 0.9$; LLaDA-8B among 8 independent trajectories at $T = 0.9$. "Perfect" selects by the true prefix value on Sudoku-S; on the other two it picks a correct completed candidate whenever one exists, an upper bound on selection at any r. Confidence is the mean over masked positions. Sudoku-M and LLaDA-8B stay within noise of no search at every r we test (worst -4.3, CI $[-8.9, +0.4]$, and -1.8, CI $[-4.9, +1.1]$).

selector	Sudoku-S 0.54M, $r = 0.6$	Sudoku-M 6.6M, $r = 0.62$	LLaDA-8B 8B, $r = 0.5$
no search (random pick)	0.685	0.633	0.768
perfect	0.840 (+0.155)	0.975 (+0.342)	0.919 (+0.151)
task oracle	0.826 (+0.141)	—	n/a
model confidence	**0.364** (**−0.321**)	**0.615** (−0.018)	**0.785** (+0.016)

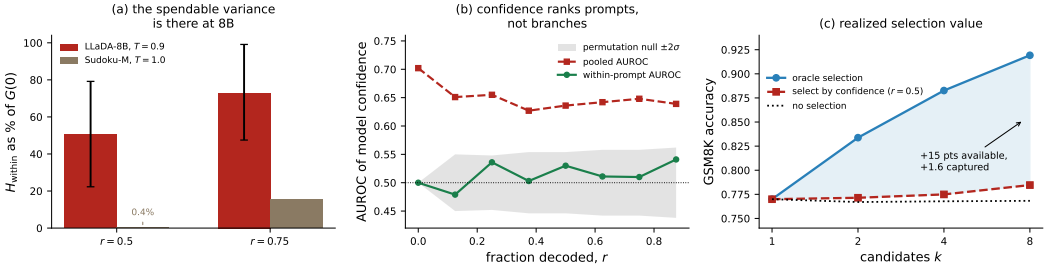


Figure 3: LLaDA-8B on GSM8K at $T = 0.9$. **(a)** Spendable variance from the branching estimator, as a share of the outcome variance still available; on Sudoku-M at $r = 0.5$ it is 0.4%. **(b)** Model confidence ranks prompts, not branches: pooled AUROC stays well above chance while the within-prompt AUROC stays inside its permutation null (260 prompts × 8 trajectories). **(c)** Realized selection value: a perfect selector gains 15 points over eight candidates, selection by confidence at $r = 0.5$ about 1.6.

every pair, while the argmax depends only on the top of the ranking; the collapse means that on this model the most confident branches are disproportionately wrong, even though confidence orders the typical pair correctly.

At 8B the failure is quieter but just as complete. Selecting the most confident of eight LLaDA trajectories lands within ±2 points of a random pick at every r we examine, capturing about 1.6 of the 15 available points (Figure 3c). The harm does not appear with more candidates either: going from 2 to 4 to 8 candidates moves the effect by at most two points in either direction, nothing like the small model's collapse. Is the small model's collapse a property of Sudoku or of that model? Sudoku-M separates the two: the same task, the same temperature and branch count, a model twelve times larger. It does not collapse. Selecting its most confident branch stays within noise of a random pick at every r (worst -4.3, CI $[-8.9, +0.4]$), under both mean confidence and the cumulative log-likelihood of what has been committed, and its within-prompt AUROC is 0.50 throughout, against a headroom of 34 points. The collapse is therefore specific to the 0.54M model. What holds on all three is the part that matters for search: model confidence recovers essentially none of a large, measured headroom. Linear probes on the residual stream fare no better on a 30-prompt run that stores hidden states at every checkpoint: their within-prompt AUROC stays inside the permutation null throughout, including a probe fit on prompt-demeaned features so that it can *only* use within-prompt variation (Appendix E).

5.4 Task structure does capture it

What does work is the task, not the network. On Sudoku-S, the exact completability oracle used as a selector gains +0.048 at $r = 0.2$ rising to +0.146 at $r = 0.85$, against a

perfect-selector ceiling of +0.190; at $r = 0.6$ it recovers 0.141 of the available 0.155 (Table 2). Sudoku-M shows the same split from the other side. There, the same oracle is at chance early — AUROC 0.501 at $r = 0.25$ and 0.537 at $r = 0.5$, because almost every quarter-filled grid is still completable — and becomes informative (0.841) only late. Over the same interval the model's confidence *degrades* as a predictor (0.762 → 0.592) while task-structural features improve (0.543 → 0.841). The model's certainty and the state's actual determinacy come apart, and every internal signal we test follows the former.

5.5 Pricing the derived methods

Allocation. Estimating difficulty from the $r = 0$ readout alone and allocating greedily by Proposition 3 beats the uniform allocation at matched budget on Sudoku-M by +1.67, +1.51, +1.67, +1.65 and +1.92 points at budgets 2 to 6 ($n = 2000$ prompts, $p = 0.035$–0.002, paired). On LLaDA ($H_{\text{between}} = 0.108$) it gains +0.7 and +0.8 at budgets 2 and 3, neither separable from zero; calibrating the bootstrap on the Sudoku runs, a gain of the predicted size would need about 2600 prompts. **Sharing the early phase.** Branching $N = 8$ tails from a shared prefix loses to independent sampling at matched compute on LLaDA: −4.0 points at $r^\star = 0.5$ (4.5 generations; CI $[-9.1, +0.8]$) and −4.7 at $r^\star = 0.75$ (2.75 generations; CI $[-9.0, -0.7]$). This is Proposition 2 working as intended: at $r^\star = 0.5$ the tails forgo 51% of the available outcome variance, and a factor 1.78 in compute does not buy that back. With Sudoku-M's value the same identity predicts sharing would be nearly free; the method must be priced per model. **Buying selectability.** Committing more cells per step makes early states more selectable — within-prompt AUROC at $r = 0.2$ goes 0.571 → 0.633 → 0.774 for 1, 2 and 4 cells per step — but costs base accuracy (0.685 → 0.615 → 0.439), and fine steps retain a higher perfect-selector ceiling where it is resolvable (+0.166, CI $[+0.039, +0.291]$ at $r = 0.6$; +0.135, $[+0.005, +0.271]$ at $r = 0.8$). Coarse steps occupy the compute frontier only below about 25 forward passes per prompt, where a single fine-step sample is unaffordable. **Where to select.** Sweeping branch count jointly with the selection point, the frontier on Sudoku-S prefers *two* branches selected at $r \approx 0.8$: 0.779 accuracy for 45 forward passes, +0.094 over one sample for 1.8× the compute, while reaching 0.879 takes 32 branches and 761 passes. Wide, early pruning — the usual framing of inference-time search, and the structure of successive halving (Jamieson & Talwalkar, 2016) — is the least efficient corner of the frontier (Figure 5). These frontier numbers assume a perfect selector; on LLaDA no internal-state readout we test comes close to one.

6 Related work

Correctness signals inside diffusion LMs. The closest work is Chiu & Liu (2026), who probe LLaDA-8B and Dream-7B at seven denoising checkpoints on four tasks. They report that the step-0 readout already carries substantial signal — an autoregressive probe on the prompt alone matches it on three of four tasks, so it "reflects prompt difficulty rather than diffusion-specific computation" — and that denoising then adds 0.08–0.11 AUC on reasoning tasks. We engage the second finding: every AUC they report is pooled across instances (their design draws one generation per prompt, so a within-prompt number is not computable from it), and we show the pooled and within-prompt curves can move in opposite directions. Their observation that selecting among 5 seeds by the step-1 probe yields "+0.0% improvement" — on one task, at one checkpoint — is the first evidence for what we measure systematically. They list "probe-guided adaptive compute allocation" as future work, which §5.5 supplies.

Per-query versus pooled evaluation. That a confidence score useful across questions need not be useful within one is not new. Taubenfeld et al. (2025) make this argument for self-consistency and introduce Within-Question Discrimination, noting that the best-calibrated measure was the least useful for selection. In information retrieval the point is structural rather than a finding (Järvelin & Kekäläinen, 2002), and Zhou et al. (2025) import IR metrics into reward-model evaluation for the same reason, finding that more responses *per prompt* improves correlation with policy regret more than more prompts. What is new here is applying the criterion to *partial* states in a diffusion model, and decomposing the variance the two criteria measure.

Variance decompositions in evaluation. Miller (2024) applies the law of total variance to the eval-score estimator, splitting question-selection variance from within-question resampling variance, and warns that pooling across repeated answers to one question violates independence. Schaeffer et al. (2025) decompose aggregate pass@k over the *distribution* of per-prompt success rates, showing a heavy tail reconciles per-problem exponential with aggregate polynomial scaling. Neither decomposes the value of a partially generated state, nor maps the components onto distinct inference-time algorithms.

Allocating test-time compute. Damani et al. (2024) derive the marginal value $\lambda(1-\lambda)^b$ of an extra sample and show greedy allocation is optimal by a matroid argument; their difficulty predictor, like our r=0 readout, runs without generating any output. They report empirically that small budgets favour easy and medium queries while large budgets shift to hard ones, which is what the interior optimum $p^\star = 1/(K+1)$ states in closed form. Snell et al. (2024) select a strategy per difficulty bin and observe that search helps most at intermediate difficulty and not at all at the extremes. We claim no novelty for the marginal-value function; we use it because the decomposition identifies allocation as the algorithm the measurable component of $\mathrm{Var}[v]$ can actually fund.

Inference-time search for dLLMs. A rapid 2026 line proposes search structures over the denoising trajectory: branching and deterministic forking (UnMaskFork, 2026), hierarchical and stratified search (Prism, 2026; S^3, 2026), particle filtering and twisted sequential Monte Carlo (SR-SMC, 2026; TDS, 2023; SVDD, 2024), policy-gradient and reward-guided unmasking (PG-DLM, 2026; RFG, 2025), and learned or confidence-driven remasking (KLASS, 2025; TACG, 2026; PSP, 2026). Related measurement work reports that most answers are settled well before decoding ends (Prophet, 2026), that reported confidence and internal representation come apart (Unsure but Certain, 2026), and that partial-masking parameterisations change what a partial state even encodes (Chao et al., 2025). Our controlled testbed follows the Sudoku diffusion setup of MGDM (2024), and our allocation comparison uses the difficulty-conditioned framing of Manvi et al. (2024). Our results are not a claim that these methods fail; they are a statement about what their selection signals can be worth, and a warning that a pooled AUROC does not price them. We follow CaRE (2026) in reporting NFE on every row, since NFE-mismatched comparisons reverse rankings in this literature.

7 Limitations

Sample size and run-to-run noise. Sudoku-S uses 50 puzzles, so every selection-value interval is wide (the perfect-selector ceiling at $r = 0.95$ is 0.879, CI $[0.780, 0.960]$). Seeding did not hold on the MPS backend: two nominally identical runs of one cell returned ceilings of 0.958 and 0.879. Our intervals are over prompts and do not include this component, so single cells should not be read as point estimates. **One large model, one task.** The 8B results are LLaDA-8B on GSM8K at $T = 0.9$; the temperature sweep was run only on Sudoku-S, so whether LLaDA's spendable variance falls with temperature as Sudoku's does is untested, as is any second diffusion LM. **The readouts we test are not every readout.** We test confidence, entropy and linear probes, the probes on only 30 prompts at 8B. A trained non-linear verifier could do better; the exact oracle bounds that concern on Sudoku, where the information is absent from early states, but not on GSM8K. **Across models.** The two Sudoku models differ sharply in spendable variance at similar temperature, and only the smaller one collapses under confidence selection; we do not isolate why. **Prior work.** We do not reproduce the 0.08–0.11 pooled denoising gain of Chiu & Liu (2026) at our operating point, which differs from theirs, and so do not claim to test whether that gain survives within-prompt. **The 8B ceiling** is selection among completed trajectories, an upper bound on selection at any r rather than the ceiling at a given r.

8 Conclusion

Reading the half-written answer is the appealing premise of inference-time search in diffusion language models. On three testbeds spanning four orders of magnitude, the signals a model

exposes about its own partial states track what it believes, not what the state has settled. Pooled AUROC credits them for ranking problems by difficulty, which no selector can use; within-prompt AUROC can credit them for a correlation whose top end is wrong. The variance search needs is real — set by the sampler, and at $T = 0.9$ on LLaDA-8B worth 15 points — and model confidence recovers essentially none of it, while a task-structural oracle recovers most of it where one exists. What the heterogeneity visible before decoding *does* fund is allocation, not selection. The discipline we would ask of any proposed verifier is narrow: report the value its intended algorithm can actually realize, at the sampler setting it will actually run.

Use of AI assistance

The results come from two independent experimental pipelines, one producing the LLaDA-8B and Sudoku-M measurements and one producing the Sudoku-S measurements. AI assistants were used substantially in both. In the first, an AI coding agent (Claude Code) was used to search and extract the literature into structured notes, write the experiment harness and analysis code, run experiments, develop the decomposition and the branching estimator, and draft and edit this manuscript. In the second, an AI assistant was used to write the model, experiment and analysis code, run the experiments, and draft the description of the results. In both, the authors directed the research questions and are responsible for the claims.

Every reported number is produced by code and checked against the conservation identities of §3 — the martingale $\mathbb{E}[v(x_r)] = \text{const}$, the identity $H_{\text{within}}(0) = 0$, the requirement that $G(r)$ be non-increasing, and a constructed signal that must score exactly 0.500 within-prompt. Several errors were caught this way and are reported rather than silently corrected, including a prompt-masking bug that broke the martingale, an undebiased variance that manufactured spurious within-prompt heterogeneity, a degenerate end-selector that produced a spurious negative result about best-of-N, and an extrapolation of spendable variance from a small model to a large one that the measurement later refuted. Claims attributed to prior work were verified against the primary sources.

References

Guan-Ming Chiu and Jeng-Yue Liu. Probing Functional Correctness in Diffusion Language Models. In *Proceedings of ACL 2026, Volume 4: Student Research Workshop*, 2026. doi:10.18653/v1/2026.acl-srw.15.

Taubenfeld et al. Confidence Improves Self-Consistency in LLMs. Findings of ACL 2025, arXiv:2502.06233.

Järvelin and Kekäläinen. Cumulated Gain-Based Evaluation of IR Techniques. ACM TOIS 20(4):422–446, 2002.

Damani et al. Learning How Hard to Think: Input-Adaptive Allocation of LM Computation. arXiv:2410.04707, 2024.

Zhou et al. Rethinking Reward Model Evaluation: Are We Barking up the Wrong Tree? ICLR 2025, arXiv:2410.05584.

Miller. Adding Error Bars to Evals. arXiv:2411.00640, 2024.

Snell et al. Scaling LLM Test-Time Compute Optimally. arXiv:2408.03314, 2024.

Schaeffer et al. How Do Large Language Monkeys Get Their Power (Laws)? ICML 2025, arXiv:2502.17578.

Chao et al. Beyond Masked and Unmasked: Discrete Diffusion Models via Partial Masking. arXiv:2505.18495, 2025.

CaRE: an audit of compute-matched evaluation for diffusion LLMs. arXiv:2607.24763, 2026.

HIVE: Hidden-evidence verification of diffusion language model trajectories. arXiv:2604.26139, 2026.

Jamieson and Talwalkar. Non-stochastic Best Arm Identification and Hyperparameter Optimization. arXiv:1502.07943, 2016.

KLASS: KL-Adaptive Stability Sampling. arXiv:2511.05664, 2025.

Backward-Mapping Consistency for diffusion LLM uncertainty. arXiv:2604.16565, ICML 2026.

Manvi et al. Adaptive Inference-Time Compute. arXiv:2410.02725, 2024.

Multi-Granularity Diffusion Modeling for Planning. arXiv:2410.14157, 2024.

Nie et al. Large Language Diffusion Models. arXiv:2502.09992, 2025.

Ou et al. Your Absorbing Discrete Diffusion Secretly Models the Conditional Distributions of Clean Data. ICLR 2025, arXiv:2406.03736.

Inference-Time Scaling of Diffusion Language Models via Trajectory Refinement. arXiv:2507.08390, COLM 2026.

Prism: Hierarchical Search and Self-Verification for Discrete Diffusion LMs. arXiv:2602.01842, ICML 2026.

Prophet: Early-commit decoding for diffusion LLMs. arXiv:2508.19982, 2026.

Successive halving for diffusion seed selection. arXiv:2607.21591, 2026.

RFG: Reward-Free Guidance for diffusion LLM reasoning. arXiv:2509.25604, 2025.

S^3: Stratified Scaling Search for Test-Time in Diffusion Language Models. arXiv:2604.06260, 2026.

Sahoo et al. Simple and Effective Masked Diffusion Language Models. NeurIPS 2024.

Shi et al. Simplified and Generalized Masked Diffusion for Discrete Data. NeurIPS 2024.

Self-Refining Sequential Monte Carlo for diffusion LLMs. arXiv:2602.01849, 2026.

Derivative-free guidance / value-based importance sampling for discrete diffusion. arXiv:2408.08252, 2024.

Trajectory-Aware Confidence Guidance. arXiv:2607.03236, 2026.

Wu et al. Practical and Asymptotically Exact Conditional Sampling in Diffusion Models. arXiv:2306.17775, 2023.

Time Is a Feature: Exploiting Temporal Oscillation in Diffusion Language Models. arXiv:2508.09138, 2025.

Train for the Worst, Plan for the Best: Understanding Token Ordering in Masked Diffusions. ICML 2025, arXiv:2502.06768.

UnMaskFork: Test-Time Scaling for Masked Diffusion via Deterministic Action Branching. arXiv:2602.04344, 2026.

Unsure but Certain: the representation–confidence gap in diffusion language models. arXiv:2608.08791, 2026.

A The cost of published correctness signals

B Proofs

Proofs of Propositions 1 and 3; both are elementary and are given here for completeness.

Both halves are immediate: in (i) all trajectories of a prompt share the same v, so their correctness indicators are exchangeable and no ranking of them can beat a random one in expectation; in (ii) all prompts share the same success rate p, and $K \mapsto 1-(1-p)^K$ is strictly concave, so by Jensen any spread of a fixed budget across prompts is *strictly worse* than dividing it evenly — with $p = 0.5$ and a budget of 4 over two prompts, $(2, 2)$ scores 1.50

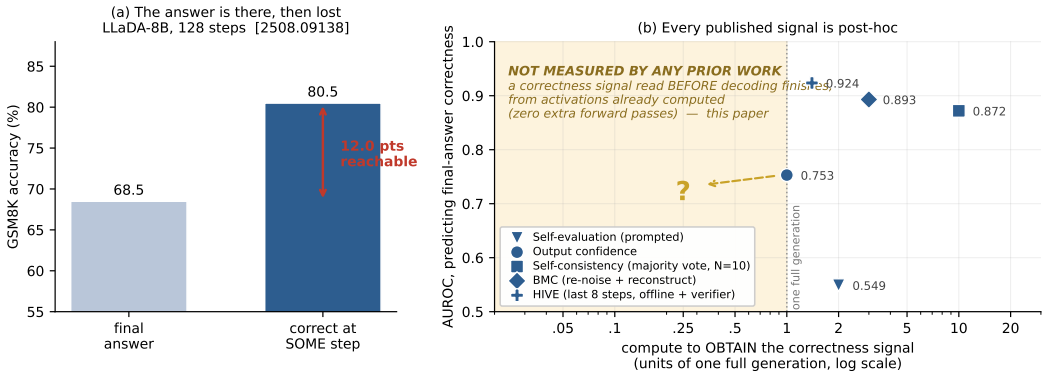


Figure 4: **(a)** The information is present mid-trajectory and then destroyed (Time is a Feature, 2025). **(b)** Every published correctness signal for a diffusion LM sits at or beyond the cost of one complete generation. The shaded quadrant—an informative signal obtained before decoding finishes, from activations already computed—is what this paper measures.

against 0.94 for $(4, 0)$. Allocation pays only by deviating from uniform, and only dispersion in p makes deviating worthwhile.

The proof is one derivative: $\partial_p\, p(1-p)^K = (1-p)^{K-1}\,[1-(K+1)p]$. We state it because the consequence is not the intuitive one, and because it means the schedule in §5.5 is fixed by measured quantities rather than tuned. We claim no novelty for the marginal-value function itself. Figure 6 shows Δ, the resulting allocation, and the gain as a function of H_{between}.

C Additional figures and tables

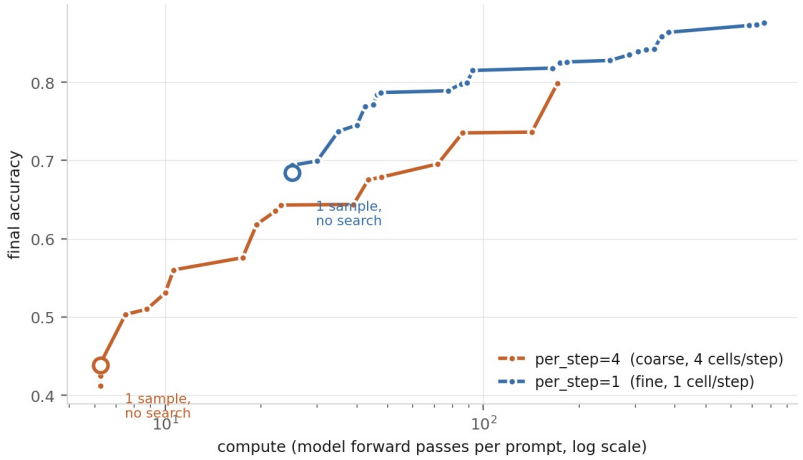


Figure 5: Sudoku-S compute frontier over branch count (2–32) and selection point r, with a perfect selector, for one cell (fine) and four cells (coarse) committed per step ($T = 0.9$, 50 puzzles). Coarse steps lead only below about 25 forward passes per prompt, where a single fine-step sample is already unaffordable; above that, fine steps dominate.

D Controlled-testbed detail

Allocation. Using only the r=0 readout to estimate difficulty, and with the protocol of Appendix E (disjoint estimation/evaluation trajectory pools, cross-fitted isotonic $\hat{p}$, draws



Figure 6: The between-prompt half of (1) prices an allocation rule. **(a)** The marginal value of one more trajectory, $p(1-p)^K$, peaks at $p^\star = 1/(K+1)$ and the peak moves left as budget grows, so compute belongs on prompts of intermediate difficulty. **(b)** The gain from optimal allocation is an increasing function of H_{between} — the quantity we measure is the quantity that prices the method. **(c)** The greedy-optimal allocation is non-monotone in difficulty: it withholds compute from both the easiest prompts (already correct) and the hardest (still wrong).

Table 3: The decomposition on the controlled testbed (150 prompts × 8 trajectories, K=16 re-completions, variances debiased). H_{within} is below 10^{-3} until $r \approx 0.5$. Pooled AUROC is high and *falls* with r; the within-prompt AUROC that selection actually needs is near chance throughout and *rises*. The two curves move in opposite directions.

r	$H(r)$	H_{between}	H_{within}	frac. within	AUROC pooled	AUROC within-prompt
0.00	0.0527	0.0531	**0.0000**	0.00	0.768	**0.500**
0.25	0.0521	0.0523	0.0002	0.00	0.772	0.502
0.50	0.0530	0.0525	0.0009	0.02	0.758	0.505
0.75	0.0834	0.0565	0.0311	0.37	0.678	0.524

without replacement, paired bootstrap over prompts), difficulty-aware allocation beats uniform at matched budget by $+1.67, +1.51, +1.67, +1.65, +1.92$ points at budgets 2–6 (n=2000; $p = 0.035, 0.010, 0.015, 0.005, 0.002$), with $H_{\text{between}} = 0.0541$. The realised gain matches the bound from a low-noise difficulty estimate, so the r=0 signal is already near-optimal here and the binding constraint is H_{between} itself. At n=400 the same effect is present but not significant (CI $[-2.0, +5.0]$): this is an effect that underpowered experiments report as absent.

Scope. This model is trained to roughly 3.3M example-views, against the 30M at which comparable Sudoku diffusion models saturate, so it sits in an undertrained regime and its confidence calibration should be read as describing that regime. The oracle result at low r, however, is a property of the task and not of the model: completability of a quarter-filled grid is simply not yet determined.

E Measurement protocol and instrument checks

The claims in this paper are measurements, so the instrument needs validating before the readings mean anything. We state the checks we ran, including the two that caught errors in our own numbers.

Validation on ground truth we constructed. We generate synthetic shards in which the signal is, by construction, a deterministic function of the prompt and carries no trajectory information. The analysis must then report high pooled AUROC and exactly chance within-prompt. It does: 0.778 pooled, 0.500 within-prompt. Any pipeline that reports a within-prompt AUROC materially above 0.5 on this input is mismeasuring, and ours did at an earlier revision.

Folds grouped by prompt. With N trajectories per prompt, an ungrouped cross-validation split places trajectories of the same prompt in both train and test. Since trajectories of a prompt share x_0, this leaks the prompt and inflates every number. All probes use `GroupKFold` on prompt identity.

A permutation null for the within-prompt number. A probe's own per-trajectory noise produces a small apparent within-prompt effect. The exact null permutes labels *inside* each prompt, destroying trajectory-level signal while leaving difficulty untouched. We draw 50 such permutations, report the null mean and standard deviation, and convert the observed value to a z-score. On the synthetic control the null is 0.496 ± 0.034 and the observed 0.565 gives $z = +1.6$ — correctly reported as not significant.

A probe built to find within-prompt structure. A probe trained on pooled labels has little incentive to represent within-prompt variation when between-prompt variance is two orders of magnitude larger, so "our probe found nothing" would be weak evidence. We therefore also fit a probe under the within transformation — features and labels demeaned per prompt, ridge, grouped folds — so that it can *only* use variation across trajectories of one prompt, and report its within-prompt AUROC separately. A null from this probe is evidence that a method built to find the signal did not find it.

Debiasing every variance. $\hat{v}$ is an average of K Bernoulli draws, so any observed variance is inflated by $v(1-v)/K$; we subtract the unbiased estimate $\hat{v}(1-\hat{v})/(K-1)$. This is load-bearing: uncorrected, $H_{\text{within}}(0)$ reads 0.010 at $K = 16$ (and 0.024 at $K = 6$) where the true value is exactly zero.

Pairing and the unit of resampling. Prompts, not trajectories, are the independent unit, so all confidence intervals bootstrap over prompts. For the allocation comparison we draw one shuffled pool per prompt and let both arms take a *prefix* of the same shuffle; resampling the two arms independently destroys the pairing and widens the interval by roughly $\sqrt{2}$. We also draw without replacement and cap the per-prompt budget at the pool size, since resampling with replacement from a finite pool saturates best-of-K and biases the high-budget rows.

A degenerate selector is easy to mistake for a negative result. Our first end-selector ranked candidates by the model's confidence at the *final* denoising step. At that step one cell remains and the model is near-certain regardless of whether the grid is right; the selector scored 0.433 AUROC — worse than chance — and made best-of-4 appear *worse* than a single sample while a correct answer was present in 60.7% of the 4-candidate pools. Replacing it with the sequence log-likelihood (the sum of $\log p$ of each token at the step it was committed) gives 0.810. We report this because the failure mode is silent and would have been indistinguishable from a genuine claim that selection does not help.

Conservation laws. Two identities must hold and are checked at every measurement: $\mathbb{E}[v(x_r)]$ is constant in r (martingale), and $H_{\text{within}}(0) = 0$ exactly. The first caught a bug in which partially decoded positions were presented to the model as prompt positions, which made $\mathbb{E}[v]$ decay $0.274 \to 0.131$ against a base rate of 0.287; after the fix it reads $0.288, 0.304, 0.295, 0.291$ against 0.285.

Diffusion setup. LLaDA-8B-Instruct, GSM8K, generation length 256, block length 32, 256 denoising steps, low-confidence remasking, temperature 0.9 (required: at temperature 0 all trajectories of a prompt are identical and the within-prompt question is vacuous). We record, at each r checkpoint, hidden states at layers $\{8, 16, 24, 32\}$ under three poolings (masked-mean, committed-mean, all-mean), together with confidence, entropy and flicker statistics computed on the answer span only. Readout cost is one linear map on activations already computed, so it adds no forward passes; we report NFE on every row regardless.