PIG: PARALLEL INCONSISTENCY GUIDANCE FOR ACCELERATING DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models accelerate generation by decoding multiple masked positions per forward pass. However, accuracy is known to drop when parallel decoding trajectories deviate from their sequential counterparts, which are slow and expensive "oracles" that adhere to the joint distribution of language. In this paper, we propose *Parallel Inconsistency Guidance* (PIG), a new paradigm that uses a trained auxiliary model to directly estimate the inconsistency between parallel and sequential generation from a single model forward pass. During inference, masked positions are traversed by decreasing confidence, and tokens are added to the parallel decode set one-at-a-time until the total estimated inconsistency between the parallel and sequential generation exceeds a threshold. Surprisingly, it suffices to train a lightweight $\sim$ 2M parameter auxiliary model on a small mixture — drawn from OpenCoder, GSM8K-train, MATH-train, and MBPP-train. For LLaDA-8B Instruct and NemotronDiffusion-3B Instruct, PIG extends the accuracy-TPF frontier on GSM8K, MATH500, MBPP, HumanEval, and MMLU, when compared to the industry standard Fast-dLLM and measures favorably on LLaDA-8B Instruct against DAPD, KLASS, EB-sampler, and other learned unmasking policies.

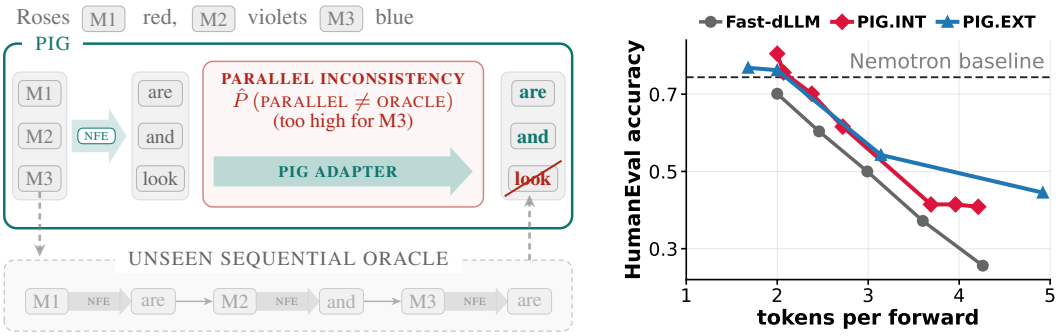

Figure 1: **Parallel Inconsistency Guidance (PIG).** *Left:* from one forward pass, PIG estimates how likely each parallel commit is to disagree with the unseen sequential decode and defers the risky ones (M3). *Right:* on Nemotron-3B HumanEval (64-token blocks), PIG.INT and PIG.EXT extend the accuracy–TPF frontier of the model's own confidence decoder and baseline reported result.

1 INTRODUCTION

Masked diffusion language models (DLMs) enable parallel text generation by predicting many masked positions at once (Austin et al., 2021a; Sahoo et al., 2024; Shi et al., 2024). Because these predictions are available from a single model evaluation, DLM decoders can reveal tokens in flexible orders and commit several positions in parallel, reducing the number of forward passes needed for generation. Recent large-scale implementations demonstrate that this parallelism can yield substantial speedups: Nemotron-Labs-Diffusion (Fu et al., 2026) decodes nearly $6\times$ more tokens per forward pass than Qwen3-8B at comparable accuracy, while DiffusionGemma (Team et al., 2026) reports up to $4\times$ faster token generation with competitive benchmark performance.

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.

However, which positions should be unmasked in parallel? A single DLM forward pass predicts every masked position from the same current canvas, so positions committed together cannot condition on one another. This lack of intermediate conditioning can make individually plausible predictions jointly inconsistent: if one position could be either "dog" or "cat" and another either "barks" or "meows," a parallel commit can produce incoherent pairs such as "dog meows" or "cat barks." (Gu et al., 2018; Zhang et al., 2026) Sequential decoding avoids this problem by updating the context after every commit, but doing so sacrifices the parallelism that makes DLMs attractive in the first place. Parallel decoders must therefore decide which positions can be safely committed together and which should wait for additional context (Campbell et al., 2022; Chen et al., 2025).

Existing parallel decoders select positions using two main signals: *confidence* and *dependence*. Confidence-based methods commit positions whose individual predictions are sufficiently certain; Fast-dLLM (Wu et al., 2025), for example, commits positions above a confidence threshold. Dependency-based methods instead avoid jointly committing positions that appear strongly coupled: DAPD (Kim et al., 2026a) constructs an attention-derived dependency graph, while DEMASK (Ringel et al., 2026) learns pairwise conditional influence. More broadly, these methods use confidence, dependence, or related signals to identify sets of positions that can be decoded together without intermediate conditioning. A common underlying theoretical objective is to select positions with high conditional independence (Chen et al., 2025; Wu et al., 2025; Zhang et al., 2026).

In this work, we instead target a complementary objective: preserving the one-token-at-a-time decoding trajectory while committing multiple positions in parallel. For any candidate sequence of positions proposed for a parallel round, we ask whether the same model would produce the same tokens if those positions were instead decoded one at a time in that sequence. We call any resulting disagreement *sequential–parallel inconsistency*. If this inconsistency could be predicted in advance, parallel decoding would amount to selecting as many positions as possible while keeping the disagreement low.

But how can the decoder know this inconsistency in advance? Naively, it would have to replay the candidate sequence one token at a time, defeating the computational advantage of parallel decoding. Yet this sequential replay turns out not to be necessary. We find that a lightweight adapter to the frozen DLM can accurately predict how committing one candidate would change the model's predictions at other positions using only information from the current forward pass. This gives rise to a simple family of parallel decoding algorithms that we call *Parallel Inconsistency Guidance* (PIG).

Empirically, PIG extends the accuracy–TPF frontier on LLaDA-8B-Instruct and Nemotron-Labs-Diffusion-3B-Instruct across four math and code benchmarks and multiple choice questions (MMLU), with its largest gains at high parallelism. Against the strong Fast-dLLM baseline (Wu et al., 2025), PIG.INT reaches up to $1.7\times$ higher tokens per forward (TPF) at matched accuracy on LLaDA and $1.4\times$ over Nemotron's native confidence decoder; at the fastest confidence settings, PIG retains up to 28 additional accuracy points. PIG also improves over dependency-guided decoders, reaching up to $4.4\times$ higher TPF, and compares favorably with learned unmasking policies (Jazbec et al., 2025b). Notably, these gains require only a $\sim$2M-parameter predictor that is trained once per backbone on a single data mixture and consumes less than 0.1% of the total forward pass MACs. In summary, our contributions are as follows.[1]

- **Parallel Inconsistency Guidance.** We formulate parallel decoding around *sequential–parallel inconsistency*: the disagreement introduced by committing a set of positions in parallel rather than decoding the same positions one at a time. This yields a family of parallel commit-set selection rules: PIG.INT controls inconsistency within the current parallel round, while PIG.EXT controls inconsistency that can be avoided by deferring positions to a later round.
- **Efficient one-pass prediction.** We show that sequential–parallel inconsistency can be predicted before any parallel commits are made using information already available in the current DLM forward pass. A $\sim$2M-parameter adapter to the frozen backbone, learned on a single dataset mixture, estimates the finite-difference structure used for selection, reaching $r \approx 0.8$ per edge and $r \approx 0.9$ on the aggregate statistics used by the decoder on held-out benchmarks.
- **Strong parallel decoding across models.** Across LLaDA and Nemotron on GSM8K, MATH-500, MBPP, and HumanEval, and MMLU, PIG improves the accuracy–TPF frontier over confidence-based decoding, with its largest gains at aggressive parallelism. We additionally

[1] Code and adapters: `https://anonymous.4open.science/r/PIGDLM` (anonymized).

compare against dependency-guided decoders and learned unmasking policies, and show that the same approach transfers across two DLM architectures with different inference structures.

2 Related Work

Confidence-based parallel decoding. Confidence is the most common criterion for parallel unmasking, from confidence-ranked masked generation (Chang et al., 2022; Ghazvininejad et al., 2019) to top-k selection (Nie et al., 2025), confidence thresholds (Cai & Li, 2026; Wu et al., 2025), and entropy criteria (Ben-Hamu et al., 2025). The intuition is that confident predictions are less likely to change as context grows, so several tokens can often be committed together safely. Fast-dLLM (Wu et al., 2025), for example, commits all positions above a confidence threshold, as does Nemotron's native diffusion decoder (Fu et al., 2026). Other methods use entropy bounds (Ben-Hamu et al., 2025), adaptive confidence rules (Liang et al., 2026), token-level stability criteria (Kim et al., 2025), or learned unmasking policies (Jazbec et al., 2025b). PIG instead evaluates a proposed parallel commit set by its predicted disagreement with sequential decoding.

Dependency-aware parallel decoding. A complementary line of work selects commits using dependencies among masked positions, estimated from attention (Deng et al., 2026; Kim et al., 2026a; Sahin et al., 2026; Zhou et al., 2026), clustering (Qi et al., 2026), or learned conditional influence (Ringel et al., 2026). The intuition is that plausible predictions can still conflict when committed together if they depend strongly on one another. AGDO (Deng et al., 2026) and DAPD (Kim et al., 2026a) use attention structure to guide decode order or avoid interacting positions, while CLAD (Qi et al., 2026) and ADAS (Sahin et al., 2026) use cluster-level and adaptive criteria. DEMASK (Ringel et al., 2026) instead learns pairwise conditional influence and constrains cumulative dependence within a parallel set. Recent theory similarly relates parallelization error to distributional dependence, including conditional total correlation and inference scheduling (Campbell et al., 2022; Chen et al., 2025; Zhang et al., 2026). PIG uses these interactions only insofar as they predict whether parallel decoding will disagree with its sequential counterpart.

Blockwise and hybrid decoding. Another line of work changes the generation protocol through blockwise decoding (Arriola et al., 2025), alternative generation orders (Hong et al., 2026), or hybrid autoregressive–diffusion decoding (Fu et al., 2026). The intuition is that structured canvas states can retain diffusion parallelism while recovering autoregressive properties such as stable prefixes and efficient caching. Block Diffusion (Arriola et al., 2025) generates blocks autoregressively while denoising positions within each block, while OeMDM (Hong et al., 2026) studies alternative diffusion generation orders. Nemotron (Fu et al., 2026) supports autoregressive, diffusion, and self-speculative decoding in one model, using diffusion proposals with autoregressive verification. PIG is complementary to these decoding protocols: given the positions currently available for decoding, it decides which should be committed together.

3 Background

Masked diffusion language models. Let $\mathcal{V}$ be a vocabulary and let $\mathbf{m} \notin \mathcal{V}$ denote a special *mask*. A generation canvas is a partially masked token sequence $\mathbf{x} \in (\mathcal{V} \cup \{\mathbf{m}\})^L$ with masked positions $M(\mathbf{x}) = \{i \in [L] : x_i = \mathbf{m}\}$. In contrast to autoregressive models that predict only the next token, a masked diffusion language model (MDLM) (Austin et al., 2021a; Sahoo et al., 2024; Shi et al., 2024) produces a vocabulary distribution $\Delta(\mathcal{V})$ at every masked position in a single forward pass. For a canvas $\mathbf{x}$ where i is masked, write $\hat{x}_i$ for its greedy prediction and ν_i for its confidence:

$$q_\theta^i(\cdot \mid \mathbf{x}) \in \Delta(\mathcal{V}), \qquad \hat{x}_i := \arg\max_{v \in \mathcal{V}} q_\theta^i(v \mid \mathbf{x}), \qquad \nu_i := q_\theta^i(\hat{x}_i \mid \mathbf{x}), \qquad i \in M(\mathbf{x}). \quad (1)$$

In a decoding round, one first performs model evaluation on the canvas $\mathbf{x}$, selects a subset of masked positions $S \subseteq M(\mathbf{x})$, assigns each $i \in S$ a token by sampling or argmax, and finally commits the tokens by updating $\mathbf{x}$ at each $i \in S$. Suppose $y_S \in \mathcal{V}^{|S|}$ were the predicted tokens for S, then let $\mathbf{x} \oplus y_S$ denote the updated canvas. In this work, we focus on greedy (argmax) decoding.

Sequential and parallel decoding. We next formalize our specific notion of sequential and parallel decoding. Suppose $S = (s_1, \ldots, s_k) \subseteq M(\mathbf{x})$ is a subset (sequence) of such masked positions ordered by confidence, i.e., $\nu_{s_i} \geq \nu_{s_j}$ for all $i \geq j$. Then for a canvas $\mathbf{x}$, the *sequential reference*

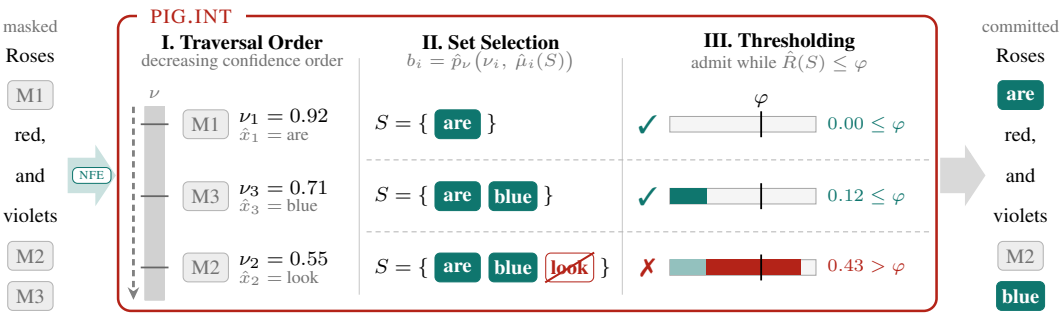

Figure 2: **PIG.INT controls single-round regret.** One forward pass (NFE) predicts every masked slot and ranks them by confidence; that ranking is the traversal order (I). Candidates are iteratively added to S, each with its predicted incremental regret $b_i = \hat{p}_\nu(\nu_i, \hat{\mu}_i(S))$ (II), and admitted only while the predicted regret $\hat{R}(S)$ fits the budget φ (III); the first candidate that does not fit closes the round. We then greedily decode all tokens in S in parallel, before starting the next round.

$x^\star_S$ is the tokens obtained by greedily decoding along S. We equivalently describe this below via stylized canvas transitions, where each arrow denotes one model evaluation followed by an update:

$$\mathbf{x} \xrightarrow{x^\star_{s_1}} \mathbf{x} \oplus x^\star_{s_1} \xrightarrow{x^\star_{s_2}} \cdots \xrightarrow{x^\star_{s_k}} \mathbf{x} \oplus x^\star_S, \qquad x^\star_{s_i} := \arg\max_{v \in \mathcal{V}} q_\theta^{s_i}\left(v \mid \mathbf{x} \oplus x^\star_{S_{<i}}\right). \qquad \text{(Sequential)}$$

Here, $S_{<i} = (s_1, \ldots, s_{i-1})$ is a prefix of S, where let $S_{<1} = \emptyset$. In contrast, our notion of parallel decoding unmasks all the argmax predictions $\hat{x}_S$ from a single model evaluation, as follows:

$$\mathbf{x} \xrightarrow{\hat{x}_S} \mathbf{x} \oplus \hat{x}_S, \qquad \hat{x}_{s_i} := \arg\max_{v \in \mathcal{V}} q_\theta^{s_i}(v \mid \mathbf{x}). \qquad \text{(Parallel)}$$

That is, sequential decoding conditions later predictions on earlier commits in S and requires $|S|$ model evaluations, while parallel decoding skips these intermediate updates and requires only one. We use the sequential output $x^\star_S$ as the reference that the parallel output $\hat{x}_S$ attempts to match, and measure their disagreement via the edit (Hamming) distance.

4 METHODOLOGY

Our main idea is to view sequential–parallel inconsistency as *regret*, and use it to decide which positions to decode. This perspective yields two complementary algorithms: PIG.INT (Section 4.1) controls regret within the current round, while PIG.EXT (Section 4.2) controls the future regret that can be avoided by delaying commits; both rely on a lightweight predictor described in Section 4.3.

4.1 PARALLEL DECODING AS REGRET-CONSTRAINED SUBSET SELECTION

Given a partially masked canvas $\mathbf{x}$, let $S = (s_1, \ldots, s_k)$ be a confidence ordered sequence of positions from $M(\mathbf{x})$ to be committed in parallel. For a single round, our idealized objective is to maximize the number of parallel commits subject to a regret bound:

$$\max_{\substack{S \subseteq M(\mathbf{x}) \\ \text{confidence ordered}}} |S| \quad \text{subject to} \quad R_{\mathbf{x}}(S) := \sum_{i \in S} \mathbf{1}\left\{x^\star_{s_i} \neq \hat{x}_{s_i}\right\} \le \varphi. \qquad (2)$$

We call $R_{\mathbf{x}}(S)$ the *single-round regret*, the realized number of mismatches between the parallel output $\hat{x}_S$ and sequential reference $x^\star_S$. However, solving this objective exactly is impractical for two main reasons: searching over ordered candidate sequence S is combinatorial, and even then, computing $R_{\mathbf{x}}(S)$ requires sequential replay at the cost of $|S|$ model evaluations.

Idea 1: Confidence ordering reduces the problem to prefix search. To avoid searching jointly over candidate orderings, we use current-pass confidence to define a fixed proposal order. Specifically, we sort the masked positions so that $\nu_{s_1} \ge \nu_{s_2} \ge \ldots$ and scan candidates for S in that order. This reduces the search over ordered candidate sets to choosing the longest *prefix* of the confidence ordering that can be committed within the regret budget.

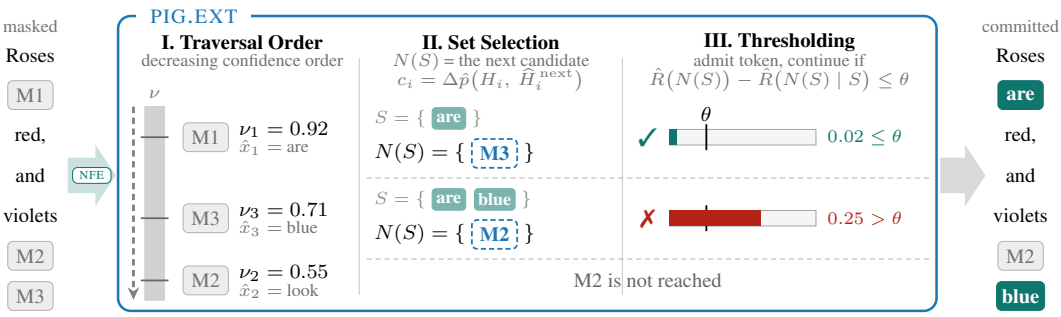

Figure 3: **Cross-round regret measures the value of waiting.** On the canvas and confidence order of Figure 2 (I), PIG.EXT admits each candidate and then prices the next candidate $N(S)$: c_i is how much the mismatch risk of i would drop after first committing S (II). The round continues while $\hat{R}(N(S)) - \hat{R}(N(S) \mid S) \le \theta$ (III): committing "are" changes little for M3, so the walk widens; committing "blue" would turn M2 from "look" into "are", so waiting is worth more than θ, the round closes and commits only "are" and "blue". We implement with $|N(S)| = 16$ look-ahead positions.

Idea 2: Prefix regret is predictable from a single model evaluation. Even with the masked positions ordered, exactly computing each $R_{\mathbf{x}}(S)$ would still require sequential replay at the cost of $|S|$ evaluations. However, a disagreement at the candidate can only arise if revealing S moves the candidate's distribution over the vocabulary enough that the argmax changes. Thus, inconsistency is governed by the slack in the top-1 vs top-2 predicted probabilities and by the size of the perturbation S applies to it, both of which a single forward pass on $\mathbf{x}$ exposes up to two deliberate approximations: the slack is approximated via confidence (or entropy), and the perturbation through a sum of pairwise perturbation estimates over $j \in S$, the first-order term of a set effect whose higher-order interactions we truncate (Section A). We defer implementation details of $\hat{R}_{\mathbf{x}}(S)$ to Section 4.3, and we next present a simple greedy algorithm that we call PIG.INT, also illustrated in Figure 2

The PIG.INT algorithm Given a canvas $\mathbf{x}$ and budget φ, first initialize $S = \varnothing$. Then, run a single forward pass on $\mathbf{x}$ and sort the masked positions $M(\mathbf{x})$ in decreasing confidence. Next, for each $i \in M(\mathbf{x})$, if $\hat{R}_{\mathbf{x}}(S \cup \{i\}) \le \varphi$, we update $S \leftarrow S \cup \{i\}$. Otherwise, we terminate this parallel decoding round and update the canvas as $\mathbf{x} \oplus \hat{x}_S$, where $\hat{x}_S$ are the argmax tokens at S.

4.2 LOOK-AHEAD DECODING AS CROSS-ROUND REGRET

We next take a step back and ask: *can slowing down help?* Once the masked positions are ordered, parallel decoding reduces to partitioning this ordering into rounds, with one model evaluation per round. Whereas PIG.INT greedily places each round boundary using single-round regret, we instead consider whether committing fewer tokens *now* can reduce regret *later*.

For the fixed confidence ordering, let S be a candidate prefix and let $N(S)$ denote the next candidate window in that ordering. We write $R_{\mathbf{x}}(N(S) \mid S)$ for the regret of decoding $N(S)$ after first committing $\hat{x}_S$ and re-evaluating, while keeping the sequential reference anchored at $\mathbf{x}$. We then define the *cross-round regret*:

$$C_{\mathbf{x}}(S) := \underbrace{R_{\mathbf{x}}(S \cup N(S)) - R_{\mathbf{x}}(S)}_{\text{decode } N(S) \text{ now}} - \underbrace{R_{\mathbf{x}}(N(S) \mid S)}_{\text{decode } N(S) \text{ after waiting}} . \tag{3}$$

Thus, a large positive $C_{\mathbf{x}}(S)$ means that waiting one round would avoid substantial regret on the upcoming candidates. Using similar machinery as PIG.INT; exactly computing cross-round regret requires additional model evaluations, but predicting it instead only requires a forward pass on $\mathbf{x}$. We defer implementation to Section 4.3. Nevertheless, access to a cross-round regret estimator yields a simple future–regret-aware algorithm, illustrated in Figure 3 and described next.

PIG.EXT: controlling cross-round regret. PIG.EXT scans candidates in decreasing confidence, adding each candidate to S before evaluating $\hat{C}_{\mathbf{x}}(S)$ on the next window $N(S)$. If $\hat{C}_{\mathbf{x}}(S) \le \theta$, waiting is predicted to buy little and the scan continues. Otherwise, PIG.EXT closes the round and commits the current S in parallel.

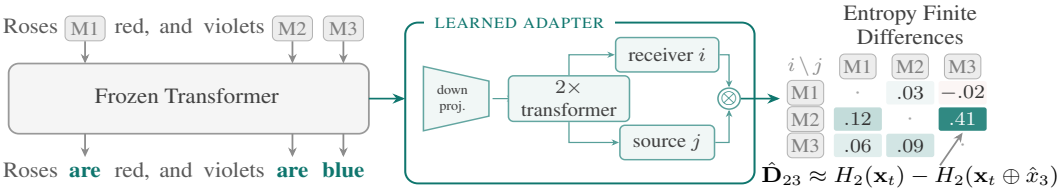

Figure 4: **Finite Differences Estimator (FDE).** The frozen DLM's pass feeds a learned adapter (≈2.2M parameters), which returns in the same pass the full directed matrix $\hat{\mathbf{D}}$ of entropy finite differences of Equation (4), $\hat{\mathbf{D}}_{ij} \approx H_i(\mathbf{x}_t) - H_i(\mathbf{x}_t \oplus \hat{x}_j)$, the change in uncertainty at i if j were committed. Full implementation and training details in (Section B).

4.3 A LIGHTWEIGHT ADAPTER ENABLES ONE-PASS REGRET PREDICTION

In Section 4.1, we argued that sequential–parallel disagreement is governed by two signals: the slack in the argmax's probability, and the perturbation that committing S applies to its distribution. We summarize the slack by confidence ν_i or predictive entropy $H_i(\mathbf{x}) := H(q_\theta^i(\cdot \mid \mathbf{x}))$; the perturbation is counterfactual. For masked positions $i, j \in M(\mathbf{x})$, we use a lightweight adapter to estimate the pairwise perturbation through the directed entropy finite difference

$$\hat{\mathbf{D}}_{ij}(\mathbf{x}) \approx H_i(\mathbf{x}) - H_i(\mathbf{x} \oplus \hat{x}_j), \tag{4}$$

where $\hat{x}_j$ is the current argmax at j. Thus, $\hat{\mathbf{D}}_{ij}$ measures how committing j changes uncertainty at i, but does not itself determine whether the argmax changes. Learning all $\hat{\mathbf{D}}_{ij}$ requires counterfactual backbone evaluations, so a lightweight adapter instead predicts the full matrix $\hat{\mathbf{D}}$ from the current decoding state (Figure 4 and Section B). Motivated by this representation, PIG.INT summarizes a candidate's interaction with the accepted prefix as

$$\hat{\mu}_i(S) := \sum_{j \in S : \nu_j > \nu_i} \max\left(|\hat{\mathbf{D}}_{ij}|, |\hat{\mathbf{D}}_{ji}|\right). \tag{5}$$

This is a deliberately low-order summary, rather than an exact reconstruction of the set effect. In Section A, we show that $\hat{\mu}_i(S)$ is the sum of first-order approximations of the larger entropy perturbation from S. From these extracted features from a single forward pass, we calibrate disagreement jointly from confidence and pairwise-sum of perturbations using offline sequential replays:

$$\begin{aligned} \hat{p}_\nu(\nu, \mu) &\approx \Pr\left(x_i^\star \neq \hat{x}_i \mid \nu_i = \nu,\, \hat{\mu}_i = \mu\right), \\ \hat{p}_H(H, \mu) &\approx \Pr\left(x_i^\star \neq \hat{x}_i \mid H_i = H,\, \hat{\mu}_i = \mu\right). \end{aligned} \tag{6}$$

PIG.INT then sums the confidence-conditioned mismatch probabilities along the prefix:

$$\hat{R}_{\mathbf{x}}(S) := \sum_{i \in S} \hat{p}_\nu\left(\nu_{s_i}, \hat{\mu}_{s_i}(S_{<i})\right). \tag{7}$$

The same representation can also estimate the value of waiting. Because the canvas after committing S is unavailable when the decision is made, PIG.EXT approximates each upcoming candidate's next-round entropy using the largest predicted directed entropy reduction from S:

$$\hat{H}_i(\mathbf{x} \mid S) := H_i(\mathbf{x}) - \max_{j \in S} \hat{\mathbf{D}}_{ij}. \tag{8}$$

We then evaluate the entropy-conditioned disagreement model at the current and predicted next-round entropies:

$$\hat{C}_{\mathbf{x}}(S) := \sum_{i \in N(S)} \left[\hat{p}_H\left(H_i(\mathbf{x}), \hat{\mu}_i(S \cup N(S))\right) - \hat{p}_H\left(\hat{H}_i(\mathbf{x} \mid S), \hat{\mu}_i(S \cup N(S))\right)\right]. \tag{9}$$

Their difference estimates the regret avoided by waiting. The adapter and calibration tables are fitted once per backbone on a **single dataset mixture** (3721 examples drawn from OpenCoder, GSM8K-train, MATH-train, and MBPP-train), and reused at inference without additional backbone evaluations. The adapter has approximately 2M parameters and costs less than 0.1% of a backbone forward pass in MACs; Section B evaluates prediction quality, while Section A gives guarantees conditional on calibration.

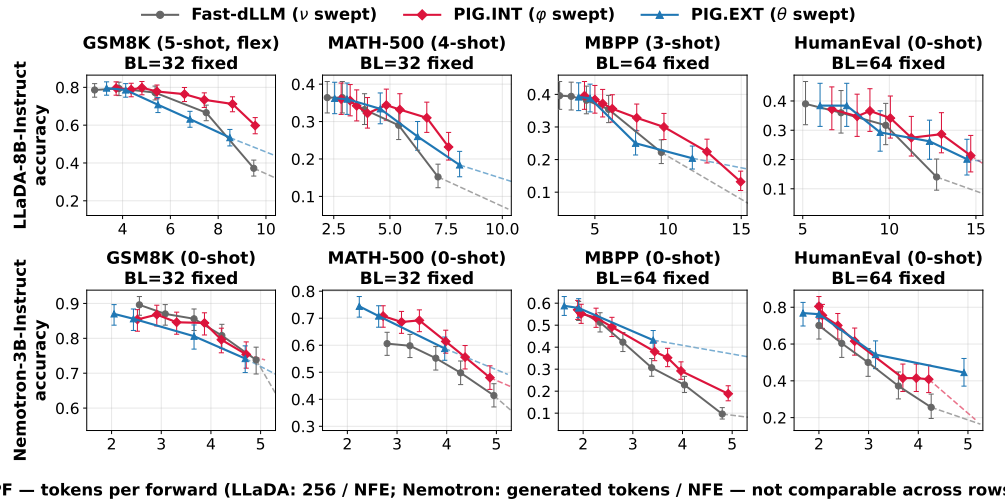

Figure 5: **PIG improves the accuracy-TPF frontier of parallel decoding** On LLaDA-8B-Instruct (top) and Nemotron-Labs-Diffusion-3B-Instruct (bottom), both PIG.INT (red) and PIG.EXT (blue) outperform a strong confidence-based baseline, Fast-dLLM (gray). We report Wilson 95% error bars, and full numerical details in Sections E and E.2.

5 Experiments

We now evaluate PIG on parallel decoding, focusing on the tokens-per-forward (TPF) metric, the average number of tokens generated per model evaluation. We give training details in Section B, and full evaluations in Section E.

Models, benchmarks, and baselines. We evaluate LLaDA-8B-Instruct (Nie et al., 2025) and Nemotron-Labs-Diffusion-3B (Fu et al., 2026) on GSM8K, MATH-500, MBPP, and HumanEval. For LLaDA, we use a 256-token canvas and follow the Fast-dLLM (Wu et al., 2025) evaluation setup: GSM8K is 5-shot, MATH-500 4-shot, MBPP 3-shot, and HumanEval 0-shot. For Nemotron, we use a 2048-token canvas and evaluate all four benchmarks 0-shot. Our primary baseline is confidence-threshold decoding with the same eligible positions as PIG: Fast-dLLM for LLaDA and Nemotron's native confidence decoder for Nemotron. We use greedy commits throughout, report Wilson 95% intervals, and compare TPF only within each model.

Evaluation and PIG configurations. For GSM8K and MATH-500, we report flexible extraction in the main text and defer strict extraction to Section E; code is evaluated with pass@1. PIG.INT sweeps the single-round regret budget φ, while PIG.EXT sweeps the cross-round regret budget θ; for LLaDA, we use $\varphi \in [0.015, 3]$ and $\theta \in [0.5, 4]$ with PIG.EXT look-ahead window as $|N(S)| = 16$. On both models the PIG.EXT lane is the same operating rule: its two slowest points additionally floor admission at a confidence ν (θ=0.125 with ν=0.9 and θ=0.25 with ν=0.8), followed by the plain θ sweep. The finite-difference estimator is trained once per backbone on training-side GSM8K, MATH, OpenCoder, and MBPP data, while the regret calibration tables are fitted on GSM8K-train and OpenCoder. Both are reused unchanged across evaluation benchmarks, decoding budgets, and the semi-auto regressive block regimes considered below.

PIG improves the accuracy–TPF frontier

We evaluate PIG.INT and PIG.EXT on LLaDA-8B-Instruct (Nie et al., 2025) and Nemotron-Labs-Diffusion-3B-Instruct (Fu et al., 2026) on GSM8K, MATH-500, MBPP, and HumanEval, against Fast-dLLM (Wu et al., 2025) (on Nemotron it is the model's own decoder). All three decoders share prompts, greedy commits, and the evaluation protocol; each is swept over its single knob (the threshold ν, the budget φ, the budget θ), and we measure decoding efficiency in TPF. For LLaDA-8B-Instruct, we additionally compare against DAPD, EB-Sampler, KLASS, and other learned policies. DAPD we run ourselves from the authors' released code under our protocol, so that comparison is paired; EB-Sampler and KLASS enter as the single operating points reported by Kim et al.

Table 1: **Matched Accuracy TPF Increase.** Each cell: our TPF over that baseline's best operating point, and in parentheses our accuracy minus theirs at the *measured* cell of ours that delivers it. Every cell is from Figure 16. Fast-dLLM and both DAPD variants are our own runs; EB-Sampler and KLASS are the cells published by Kim et al. (2026a). LLaDA-8B-Instruct, BL=64 fixed.

	our protocol (paired)			their protocol (Kim et al., 2026a)	
benchmark	Fast-dLLM	DAPD-Direct	DAPD-Staged	EB-Sampler	KLASS
PIG.INT	relative TPF increase (+ or − accuracy)				
GSM8K	1.15× (+0.0)	1.77× (+2.0)	2.62× (+0.6)	2.51× (+0.5)	2.71× (+1.0)
MATH-500	1.13× (−0.2)	1.65× (+1.0)	1.82× (+0.6)	3.38× (+1.2)	4.38× (+1.4)
MBPP	1.66× (+0.0)	0.83× (+1.0)	1.85× (+0.4)	2.60× (+3.0)	2.05× (+4.0)
HumanEval	1.29× (−1.8)	1.46× (+2.4)	4.28× (+3.1)	2.88× (+0.0)	3.90× (−0.6)
PIG.EXT	relative TPF increase (+ or − accuracy)				
GSM8K	0.85× (−0.6)	1.04× (+4.4)	1.91× (+1.4)	1.47× (+2.9)	1.59× (+3.4)
MATH-500	0.87× (+0.6)	1.00× (+2.4)	1.57× (+0.8)	2.84× (+2.0)	2.97× (+3.8)
MBPP	1.50× (−0.4)	0.90× (+0.2)	1.73× (+0.8)	2.40× (+1.0)	0.98× (+12.8)
HumanEval	1.46× (−0.6)	1.10× (+6.7)	4.06× (+0.6)	3.25× (+1.2)	4.40× (+0.6)

(2026a), whose prompts and grader differ from ours (Section D.5 measures the offset at the shared Fast-dLLM point). Figure 5 shows the math benchmarks under fixed 32-token blocks and the code benchmarks under fixed 64-token blocks; the single-block and remaining block regimes, with every operating point tabulated, are in Section E. Both decoders read their finite differences from one ∼2M-parameter adapter on the frozen backbone, fitted once per model on a **single data mixture** of planted entropy finite differences and never updated while decoding. On held-out benchmarks it recovers the gold plants at a per-edge correlation of about 0.8, and higher on the aggregate measurement the decoders actually consume, the pairwise-sum of perturbations $\hat{\mu}_i$. Section B gives its architecture, training data and objective, and evaluates it on both backbones, the Nemotron twin at a block size held out of training.

PIG consistently improves the accuracy–TPF frontier over Fast-dLLM, with the largest gains appearing as decoding becomes more parallel. We observe that globally, PIG.INT performs best at the slower end of the frontier while PIG.EXT performs best at the faster end of the frontier. We note that through our experiments, PIG helps globally on the whole frontier on all tested datasets and block sizes with the most impressive results occurring at medium length block sizes.

Table 1 reports the relative increase in TPF of PIG on LLaDA-8B-Instruct against Fast-DLLM and dependency-aware decoders DAPD (Kim et al., 2026a), EB-Sampler (Ben-Hamu et al., 2025), and KLASS (Kim et al., 2025), at the four-block (64-token) semi-autoregressive block decoding. At matched accuracy PIG.INT decodes 1.5 to 1.8× more tokens per forward than DAPD-Direct on three of the four benchmarks, and 0.8× on MBPP, in every case at equal or higher accuracy; against the reported EB-Sampler and KLASS points it decodes with 2.1 to 4.4× higher TPF. Meanwhile, we observe that PIG.EXT struggles a bit more at the slower end of the accuracy-TPF frontier. Against Fast-dLLM, PIG.EXT decodes faster on code tasks only at $1.46 - 1.50\times$ higher TPF; against DAPD-Direct it achieves an average increase of 3.4 percentage points at similar TPF ($0.9 - 1.1\times$); against the reported EB-Sampler and KLASS points it decodes 1.5 to 4.4× more tokens per forward. Section D draws the full frontiers behind these numbers (Figure 16) and shows that PIG matches, at their own operating points, the RL unmasking policies of Jazbec et al. (2025b) that are trained per benchmark (Figure 17).

In contrast, Table 2 reports interpolated accuracy of the same set-up at fixed throughputs per model. At 10 TPF on LLaDA, PIG.INT is the most accurate lane on every benchmark, 4 to 19 points above Fast-dLLM; at 15 TPF PIG.EXT takes over on GSM8K and MBPP with 10 to 14 points over Fast-dLLM, and on MATH-500 it is the only lane still measured at 15 TPF. On Nemotron-3B-Instruct, the three lanes stay within a few points of one another on GSM8K, while on code PIG.EXT retains 36.9 against 9.1 points on MBPP at 5 TPF and 44.5 against 19.5 on HumanEval at 4.9 TPF, the last measured cell of its sweep.

Table 2: **Accuracy at Matched TPF.** Reported accuracies and gains relative to Fast-dLLM in parentheses, when above (teal) and below (red). Every lane is read at exactly the target throughput, linearly between the two measured cells that bracket it. A lane whose sweep ends before the target is not extrapolated: its last measured cell is given instead, as accuracy@TPF. All rows at BL=64 fixed.

benchmark	Fast-dLLM	PIG.INT	PIG.EXT	Fast-dLLM	PIG.INT	PIG.EXT
LLaDA-8B-Instruct	interpolated accuracy @ TPF = 10			interpolated accuracy @ TPF = 15		
GSM8K	51.4	70.5 (+19.1)	56.0 (+4.6)	25.6	34.0 (+8.5)	39.7 (+14.2)
MATH-500	14.1	25.5 (+11.4)	20.0 (+5.9)	9.8 @12.2	15.4 @13.4 (—)	13.7 (—)
MBPP	21.0	29.3 (+8.3)	22.4 (+1.4)	7.8	13.2 @15.0 (+5.3)	17.5 (+9.7)
HumanEval	30.4	34.3 (+3.8)	28.7 (−1.7)	9.2	20.2 (+11.0)	19.7 (+10.5)
Nemotron-3B-Instruct	interpolated accuracy @ TPF = 3			interpolated accuracy @ TPF = 5		
GSM8K	86.5	83.7 (−2.8)	83.7 (−2.8)	76.1	76.7 (+0.5)	75.9 (−0.3)
MBPP	38.3	43.7 (+5.5)	47.2 (+8.9)	9.1	18.8 @4.9 (+9.5)	36.9 (+27.8)
HumanEval	49.8	55.8 (+6.0)	57.0 (+7.2)	18.8	22.4 (+3.6)	44.5 @4.9 (+25.0)

Four further experiments sit in Section D. We propose dynamic block decoding, which expands the active block when the candidate pool starves, lifting the frontier for every lane and extending PIG.INT's reach from 9.6 to 12.0 TPF on GSM8K and from 7.6 to 9.7 TPF on MATH-500 (Section D.1). Additionally, a local path-curvature diagnostic measures how committing a set shifts upcoming-candidate entropies, finding both variants widen the round where the decode path is locally flat and narrows it where it bends (Section D.2). Next, we test generalization on MMLU, an unseen multiple-choice benchmark spanning 57 subjects, and observe performance consistent with our main results (Section D.3). Finally, we audit both regret estimates during live decoding on LLaDA-8B-Instruct by replaying committed sets sequentially: PIG.INT's realized regret tracks the prediction and stays below φ at every operating point, and PIG.EXT's cross-round statistic tracks the realized cross-round regret at the slow end and over-predicts it at large θ (Section D.4).

6 Discussion, Future Work, and Conclusion

Parallel decoding for DLMs. Parallel decoding is the primary mechanism by which DLMs achieve test-time speedups. Whereas existing approaches typically seek sets of positions that are approximately conditionally independent, we instead select parallel commits by directly controlling their disagreement with sequential decoding. Empirically, this disagreement provides an effective decoding target, suggesting a complementary design principle for future parallel decoders.

Why is disagreement predictable? Our key hypothesis is that sequential–parallel disagreement can be predicted from current confidence and pairwise perturbations from earlier commits. Prior dependency-aware decoders also use pairwise interactions (Kim et al., 2026a; Ringel et al., 2026), but to our knowledge PIG is the first to target disagreement with sequential decoding directly. That such a simple predictor works suggests that richer estimators could further improve performance.

Limitations. We use greedy sequential decoding as the reference trajectory; other sequential policies could induce different notions of regret and different preferred parallel sets. Our predictor and calibration transfer across tasks without benchmark-specific fitting, but still depend on the quality of training-side data. Moreover, although PIG.INT and PIG.EXT excel in different regimes, we currently lack a principled rule for choosing which to use on a new model or task.

Future work. Our estimator could benefit from richer architectures, objectives, or training data. Sequential–parallel regret may also be predictable from features beyond confidence and aggregate pairwise perturbation. A natural next step is a unified decoder that decides when to control current-round regret and when to invoke PIG.EXT's look-ahead criterion. More generally, methods could jointly learn the regret estimator and selection policy rather than treating them separately.

Conclusion. We study how to accelerate DLM inference by choosing which masked positions can safely be committed together. Our algorithmic framework, PIG, formulates this decision through sequential–parallel regret and shows that this quantity can be predicted cheaply from a single model pass. Across two DLM families and standard math and code benchmarks, PIG.INT and PIG.EXT improve the accuracy–compute frontier over confidence-based parallel decoding.

Ethics Statement

This work studies inference-time decoding for diffusion language models that are already publicly released; it introduces no new model, no new training corpus and no human-subjects component. All experiments use public benchmarks (GSM8K, MATH-500, MBPP, HumanEval) under their released terms, and every generation is produced by a publicly available checkpoint. PIG changes only how many positions a model commits per forward pass, so it inherits the biases, factual errors and misuse potential of the backbone it accelerates rather than introducing new ones. Cheaper inference does lower the cost of generating text at scale, which is a property shared by all decoding acceleration. We report every operating point we ran, including those on which PIG does not improve on the baseline.

Reproducibility Statement

The decoders are written out in full in Section C, and every number in the paper is read from the operating points tabulated in Section E, which lists each cell's accuracy, throughput, problem count and Wilson interval, including the cells the figures crop. Sections E.1 and E.2 state, per model, the benchmarks, prompt templates, shot counts, graders, canvas lengths, block geometries and baseline sweeps that produced them. Section D records where every externally reported number comes from and how we reproduced DAPD from the authors' released code, together with the prompt ablation that accounts for most of the remaining gap. Section B gives the estimator's architecture, inputs, training corpora, harvest procedure, objective, optimiser and checkpoint selection, with its held-out evaluation. The assumptions behind the guarantees are stated in full in Section A, where they are also proved. Code and the trained adapters are released at the repository linked in Section 1.

Use of Large Language Models

Large language models (Anthropic's Claude, through the Claude Code environment) were used throughout the preparation of this paper. In the categories for which ICLR requires disclosure, they were used to help develop the notation and presentation of the theoretical analysis of Section A, including restructuring its definitions and simplifying steps of its proofs; to implement and debug the decoding, evaluation and analysis code. In the categories for which disclosure is recommended, they were used to draft and edit prose, to render the figures and tables from those records, and to check LaTeX. The research questions, the method, the statements of the theorems and their proofs, the experimental design, and every claim in the paper are the authors' own. The authors reviewed all AI-assisted work: every reported number was traced back to the experimental record that produced it, every theorem statement and proof was verified by the authors, and all generated code was checked against the registered results it reproduces.

References

Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM*, 55(5):23:1–23:27, 2008.

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 (ICLR)*, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021a.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *arXiv preprint arXiv:2505.24857*, 2025.

Changxiao Cai and Gen Li. Confidence-based decoding is provably efficient for diffusion language models. *arXiv preprint arXiv:2603.22248*, 2026.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. tau-leaping error bound for parallel discrete-diffusion sampling.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Irène Charon and Olivier Hudry. An updated survey on the linear ordering problem for weighted or unweighted tournaments. *Annals of Operations Research*, 175(1):107–158, 2010.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Sitan Chen, Kevin Cong, and Jerry Li. Optimal inference schedules for masked diffusion models. *arXiv preprint arXiv:2511.04647*, 2025. optimal out-of-order schedules governed by total correlation; need structural prior knowledge.

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.

Jia Deng, Junyi Li, Wayne Xin Zhao, Jinpeng Wang, Hongyu Lu, and Ji-Rong Wen. Beyond fully random masking: Attention-guided denoising and optimization for diffusion language models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. AGDO.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, et al. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. *arXiv preprint arXiv:2607.05722*, 2026.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Michel Grabisch. *Set Functions, Games and Capacities in Decision Making*. Theory and Decision Library C. Springer, 2016.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations (ICLR)*, 2018.

Venkatesan Guruswami, Rajsekar Manokaran, and Prasad Raghavendra. Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks*, 2021b.

Chunsan Hong, Sanghyun Lee, and Jong Chul Ye. Unifying masked diffusion models with various generation orders and beyond. In *International Conference on Machine Learning (ICML)*, 2026. OeMDM / LoMDM.

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

Metod Jazbec, Theo X. Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. ml-rl-dllm: Learning unmasking policies for diffusion language models (official code). `https://github.com/apple/ml-rl-dllm`, 2025a. GitHub repository, commit `1309c06`; accessed 2026-09-15.

Metod Jazbec, Theo X. Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. Learning unmasking policies for diffusion language models. *arXiv preprint arXiv:2512.09106*, 2025b.

Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pp. 85–103. Springer, 1972.

John G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

Bumjun Kim, Dongjae Jeon, Moongyu Jeon, and Albert No. DAPD: Dependency-aware parallel decoding via attention for diffusion LLMs. *arXiv preprint arXiv:2603.12996*, 2026a.

Bumjun Kim, Dongjae Jeon, Moongyu Jeon, and Albert No. DAPD: Dependency-aware parallel decoding via attention for diffusion LLMs (official code and README). `https://github.com/quasar529/DAPD`, 2026b. GitHub repository, commit `05727b0`; accessed 2026-09-15.

Seo Hyun Kim, Sunwoo Hong, Hojung Jung, Youngrok Park, and Se-Young Yun. KLASS: KL-guided fast inference in masked diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yihao Liang, Ze Wang, Hao Chen, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Emad Barsoum, Zicheng Liu, and Niraj K. Jha. CD4LM: Consistency distillation and adaptive decoding for diffusion language models. *arXiv preprint arXiv:2601.02236*, 2026. Confidence-Adaptive Decoding (CAD).

Rafael Martí, Gerhard Reinelt, and Abraham Duarte. A benchmark library and a comparison of heuristic methods for the linear ordering problem. *Computational Optimization and Applications*, 51(3):1297–1317, 2012.

Alantha Newman. Approximating the maximum acyclic subgraph. Master's thesis, Massachusetts Institute of Technology, 2000.

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. *arXiv preprint arXiv:2502.09992*, 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 *International Conference on Learning Representations (ICLR)*, 2025. RADD; absorbing diffusion = clean-data conditionals = any-order AR.

Heqiang Qi, Wei Huang, Mingyuan Bai, and Xiangming Meng. Cluster-level attention-guided parallel decoding for masked diffusion language models. *arXiv preprint arXiv:2605.29607*, 2026.

Liran Ringel, Ameen Ali, and Yaniv Romano. Dependency-guided parallel decoding in discrete diffusion language models. *arXiv preprint arXiv:2604.02560*, 2026.

Gian-Carlo Rota. On the foundations of combinatorial theory I: Theory of Möbius functions. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 2(4):340–368, 1964.

Yusuf Sahin, Ahmed Rockey Saikia, Volkan Cevher, and Paolo Favaro. Attention-discounted adaptive sampler for masked diffusion language models. *arXiv preprint arXiv:2606.10829*, 2026.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37: 103131–103167, 2024.

DiffusionGemma Team, Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, et al. Diffusiongemma technical report. *arXiv preprint arXiv:2608.00146*, 2026.

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.

Shaorong Zhang, Longxuan Yu, Rob Brekelmans, Luhan Tang, Salman Asif, and Greg Ver Steeg. Generation order and parallel decoding in masked diffusion models: An information-theoretic perspective. *arXiv preprint arXiv:2602.00286*, 2026.

Yuyan Zhou, Kai Syun Hou, Weiyu Chen, and James Kwok. Attention-based sampler for diffusion language models. *arXiv preprint arXiv:2604.08564*, 2026.

APPENDIX CONTENTS

A Theoretical Motivation and Analysis **15**

A.1 The decode-ordering objective and its entropy form 15

A.2 The Möbius expansion and the graph representation 16

A.3 The first-order problem is a linear ordering problem, and it is NP hard 16

A.4 Guarantees for PIG.INT and PIG.EXT 17

B The Entropy Finite Differences Estimator: Implementation and Training **21**

B.1 The finite-difference estimator 21

B.2 The calibrated mismatch table 24

C Algorithms **29**

D Additional Experiments and Analyses **32**

D.1 Dynamic block decoding unlocks additional parallelism 32

D.2 PIG adapts to the information curvature of decoding 33

D.3 Domain transfer: PIG carries over to MMLU without refitting 35

D.4 The adapter is a calibrated predictor of regret 36

D.5 Comparison with dependency-guided decoders 39

D.6 A learned unmasking policy at matched block geometry 41

D.7 One number per comparison: matched-accuracy throughput and accuracy at a fixed throughput 42

E Complete Frontier Grids and Tables **43**

E.1 LLaDA-8B-Instruct 43

E.2 Nemotron-Labs-Diffusion-3B 45

E.3 Frontier tables 50

A THEORETICAL MOTIVATION AND ANALYSIS

This appendix gives the mathematical grounding of PIG in two parts. Sections A.1 to A.3 motivate the representation the decoder reads from one forward pass: the decode-ordering objective, its deployable entropy form, its Möbius expansion (whose first-order coefficients are exactly the pairwise entropy finite differences of Equation (4)), and the NP-hardness of the exact ordering problem on that graph, which is why PIG fixes the order by confidence and spends its budget on admission. Section A.4 then states what the two decoders certify: a per-pass budget on parallel–sequential disagreement for PIG.INT (Proposition A.4) and a per-accelerated-token regret bound for PIG.EXT (Proposition A.8), both under a single calibration assumption on the inconsistency estimator.

A.1 THE DECODE-ORDERING OBJECTIVE AND ITS ENTROPY FORM

Recall the MDLM training objective as the masked cross-entropy

$$\mathcal{L}_{\mathrm{MDLM}}(\theta) \;=\; \mathbb{E}_{t\sim U[0,1]}\,\mathbb{E}_{\mathbf{x}\sim p}\Big[\frac{1}{t}\sum_{i\in M(t)} -\log q_\theta\big(x_i \mid \mathbf{x}(t)\big)\Big], \tag{10}$$

and sampling a noise level and a uniform mask is the same experiment as sampling a uniformly random *order* π and revealing a random prefix of it, so Equation (10) is the expected teacher-forced chain loss over random orders (Hoogeboom et al., 2022; Ou et al., 2025):

$$\mathcal{L}_{\mathrm{MDLM}}(\theta) \;=\; \mathbb{E}_{\pi\sim\mathrm{Unif}}\,\mathbb{E}_{\mathbf{x}\sim p}\Big[\textstyle\sum_k -\log q_\theta\big(x_{\pi(k)} \mid x_{\pi(<k)}\big)\Big]. \tag{11}$$

Training *averages* over orders; decoding *executes* one. For a fixed order the chain rule factorizes both the data law and the sampler the order induces,

$$p(\mathbf{x}) = \prod_k p\big(x_{\pi(k)} \mid x_{\pi(<k)}\big), \qquad q_\theta^\pi(\mathbf{x}) = \prod_k q_\theta\big(x_{\pi(k)} \mid x_{\pi(<k)}\big), \tag{12}$$

the left for every π, since the data admits all orders equally. As such diffusion models can be viewed as *any order auto-regressive models*. A finite-capacity model cannot match p under every factorization at once, so orders differ in how far the sampler deviates from p; the deviation of an order is its expected teacher-forced loss,

$$\mathbb{E}_{\mathbf{x}\sim p}\Big[\textstyle\sum_k -\log q_\theta\big(x_{\pi(k)} \mid x_{\pi(<k)}\big)\Big] \;=\; H(p) \;+\; \mathrm{KL}\big(p \,\|\, q_\theta^\pi\big),$$
$$\text{where}\quad \mathrm{KL}\big(p \,\|\, q_\theta^\pi\big) = \sum_k \mathbb{E}_{x_{\pi(<k)}}\Big[\mathrm{KL}\big(p(\cdot \mid x_{\pi(<k)}) \,\|\, q_\theta(\cdot \mid x_{\pi(<k)})\big)\Big], \tag{13}$$

where the second equality substitutes Equation (12) and splits the sum by the tower rule. $H(p)$ does not depend on π, so all order dependence sits in the KL term: the cost an order pays is an accumulated sum of per-step conditional errors, one per reveal, and order selection at a fixed model is

$$\text{Idealized Inference Objective} = \min_\pi \mathrm{KL}(p \,\|\, q_\theta^\pi), \tag{14}$$

a combinatorial problem over $n!$ orders of a cost in which each token's term depends on its whole revealed set.

The entropy form. The summands of Equation (13) are *gold* readouts, measurable under teacher forcing and invisible at deployment, where the revealed states are the sampler's own commits. The deployable substitute is the model's own expectation of the same per-step loss,

$$H\big(q_\theta(\cdot \mid x_{\pi(<k)})\big) = \mathbb{E}_{y\sim q_\theta(\cdot|x_{\pi(<k)})}\big[-\log q_\theta(y \mid x_{\pi(<k)})\big], \tag{15}$$

so that minimizing the cumulative decode entropy,

$$\text{Practical Inference Objective} = \min_\pi \sum_k \mathbb{E}_{\hat{x}_{\pi(<k)}}\Big[H\big(q_\theta(\cdot \mid \hat{x}_{\pi(<k)})\big)\Big], \tag{16}$$

selects the trajectory the model itself prices cheapest: each token is committed once its context has collapsed the uncertainty about it. Everything below applies verbatim with entropy readouts in place of gold readouts, and it is the entropy form that PIG measures.

A.2 THE MÖBIUS EXPANSION AND THE GRAPH REPRESENTATION

Treat each token's readout as a *set function* of its revealed context. Over the masked span $V = \{1, \dots, n\}$ write

$$f_i(S) \;=\; -H_i(\mathbf{x} \oplus x_S) \quad \text{(entropy)}, \qquad S \subseteq V \setminus \{i\}, \tag{17}$$

so that order selection maximizes a sum of set functions, each evaluated on the prefix the order gives it. Every set function decomposes by inclusion–exclusion through its *Möbius transform*, the Taylor series of a set function (Grabisch, 2016; Rota, 1964):

$$G(\pi) \;=\; \sum_i f_i\big(\pi(<i)\big) \;=\; \sum_i \sum_{T \subseteq \pi(<i)} m_i(T), \tag{18}$$

where $m_i(T)$ is the part of f_i attributable jointly to the subset T and to nothing smaller: $m_i(\varnothing) = f_i(\varnothing)$, $m_i(\{j\}) = f_i(\{j\}) - f_i(\varnothing)$, and so on. Truncating the expansion at singletons is the first-order approximation in the sense of set functions, the same move that turns a smooth loss into a gradient step.

Expanding around *any* base context S_0, the first-order coefficients are finite differences,

$$f_i(S) \;\approx\; f_i(S_0) \;+\; \sum_{j \in S \setminus S_0} \mathbf{D}_{ij}(S_0), \qquad \mathbf{D}_{ij}(S_0) \;=\; f_i(S_0 \cup \{j\}) - f_i(S_0), \tag{19}$$

the discrete derivative of f_i in direction j at S_0: the gain in i's readout from revealing j given that S_0 is already present. Substituting into Equation (18) and grouping by interaction order decomposes the objective exactly, in correspondence with $f(x_0 + h) = f(x_0) + \nabla f(x_0)^\top h + O(\|h\|^2)$:

$$G(\pi) \;=\; \underbrace{\sum_i f_i(S_0)}_{\text{constant}} \;+\; \underbrace{\sum_{i,j\,:\,\pi(j)<\pi(i)} \mathbf{D}_{ij}(S_0)}_{\text{first order: collected edges}} \;+\; \underbrace{\sum_i \sum_{\substack{T \subseteq \pi(<i) \\ |T| \geq 2}} m_i(T)}_{\text{higher order: truncated}}. \tag{20}$$

The representation is the one we learn. The first-order term is a directed graph on the masked positions: the numbers $\mathbf{D}_{ij}(S_0)$, one per ordered pair, are its adjacency matrix, and the term collects exactly the edges an order picks up. At deployment the base context is the current canvas, $S_0 = \mathbf{x}_t$, and under the entropy readout of Equation (17) the edge is

$$\mathbf{D}_{ij}(\mathbf{x}_t) \;=\; -H_i(\mathbf{x}_t \oplus \hat{x}_j) + H_i(\mathbf{x}_t), \tag{21}$$

the directed entropy finite difference of Equation (4). The matrix the adapter of Section 4.3 predicts in one pass is therefore the first-order Möbius coefficient field of the decode-ordering objective at the current state, re-expanded at every round; the higher-order layer of Equation (20) is what a single forward pass cannot see, and it is the part that parallel commits expose (within a round, several edges are collected at once, none of them re-evaluated in the others' presence). PIG.INT aggregates the magnitudes of these edges into $\hat{\mu}_i(S)$ (Equation (5)) as the one-pass proxy for how much of a candidate's readout is being extrapolated through omitted interactions, and PIG.EXT reads their sign to predict the next round's entropies (Equation (8)); the calibration tables of Equation (6) then convert the proxy into a mismatch probability.

A.3 THE FIRST-ORDER PROBLEM IS A LINEAR ORDERING PROBLEM, AND IT IS NP HARD

Definition A.1 (Denoising-order LOP)**.** Given an edge graph $\mathbf{D} \in \mathbb{R}^{n \times n}$, the denoising-order linear ordering problem is

$$\pi^\star = \arg\max_{\pi \in S_n} W_{\mathbf{D}}(\pi), \qquad W_{\mathbf{D}}(\pi) = \sum_{i,j\,:\,\pi(j)<\pi(i)} \mathbf{D}_{ij}, \tag{22}$$

the total edge mass flowing forward in π: place each token *after* the tokens that most raise its readout.

Maximizing the first-order term of Equation (20) over orders is exactly Equation (22). Its maximizer is the maximum-weight acyclic subgraph and the complementary backward mass is the minimum feedback arc set, so the LOP is NP-hard (Karp, 1972) and APX-hard (Charon & Hudry, 2010;

Newman, 2000); under the Unique Games Conjecture no polynomial algorithm beats a random order by a constant (Guruswami et al., 2008), and the tournament case is Kemeny rank aggregation (Ailon et al., 2008; Kemeny, 1959). The hardness is intrinsic to the graph representation rather than to our edges: Zhou et al. (2026) bound the order objective of a one-layer attention model and prove the corresponding weighted LOP NP-hard on the attention graph, the same problem our first-order truncation poses on the finite-difference graph, and then retreat to a closed-form rank by attention column sums (the gated attention rule of AGDO (Deng et al., 2026)). Reached on either graph — gold or entropy finite differences here, raw attention there — the conclusion is that the exact ordering problem is not the object to solve per round. Generated spans are short enough that local search would land within about 1% of the LOP optimum (Martí et al., 2012), but the truncation itself, not the solver, is the binding error: the first-order objective prices a token's first two context slots exactly and extrapolates every deeper slot through the omitted interactions, so an exact LOP solution optimizes a largely speculative tail. PIG therefore does not order: it takes the confidence order the model already provides as the search order, and spends its one knob on *admission* — how many tokens of that order can be committed in one pass before the omitted interactions, read through the first-order edges, are predicted to change the tokens (Section 4.1). The regret that PIG bounds is the parallel-versus-sequential disagreement on exactly the sets whose within-round edges the first-order term collects without re-evaluation.

A.4 Guarantees for PIG.INT and PIG.EXT

We now give guarantees on the consistency of the parallel decode against the sequential decode it stands in for. First, we provide the theoretical setup.

Setup. Recall the Sequential and parallel decoding from Section 3. Let $S = (s_1, \dots, s_k) \subseteq M$ be an ordered sequence of distinct masked positions, and write $\mathbf{x} \oplus x_S$ for the canvas obtained by updating S with tokens x_S. We compare sequential and parallel decoding through the following stylized canvas transitions, where each arrow denotes one model evaluation followed by the indicated commit. For the sequential reference $x^\star$, the model is evaluated separately before each commit:

$$\mathbf{x} \xrightarrow{x^\star_{s_1}} \mathbf{x} \oplus x^\star_{s_1} \xrightarrow{x^\star_{s_2}} \cdots \xrightarrow{x^\star_{s_k}} \mathbf{x} \oplus x^\star_S, \qquad x^\star_{s_i} := \arg\max_{v \in \mathcal{V}} q_\theta^{s_i}\left(v \mid \mathbf{x} \oplus x^\star_{s_{<i}}\right). \tag{Sequential}$$

Here $s_{<i} = (s_1, \dots, s_{i-1})$. Parallel decoding instead commits the current predictions $\hat{x}_S$ from a single model evaluation:

$$\mathbf{x} \xrightarrow{\hat{x}_S} \mathbf{x} \oplus \hat{x}_S, \qquad \hat{x}_{s_i} := \arg\max_{v \in \mathcal{V}} q_\theta^{s_i}(v \mid \mathbf{x}). \tag{Parallel}$$

Thus, sequential decoding conditions later predictions on earlier commits in S and requires $|S|$ model evaluations, while parallel decoding skips these intermediate updates and requires only one. We use the sequential output $x^\star_S$ as the reference that the parallel output $\hat{x}_S$ attempts to match, which we measure by their Hamming distance on S only.

Fix a round with canvas $\mathbf{x}_t$ and committed set $S = \{s_1, \dots, s_{|S|}\} \subseteq M$ in decreasing confidence. Every set the walk admits is a prefix of the confidence order, so the sequential chain feeding a position i runs over $M_{\succ i} = \{j \in M : \nu_j > \nu_i\}$ whatever the round, and only the two canvases the walks start from need naming. For a target anchor $\mathbf{x}$ and a parallel anchor $\mathbf{x}'$ let

$$F_\theta(i; \mathbf{x}, \mathbf{x}') \;=\; \mathbf{1}\{x_i^* \neq \hat{x}_i\}, \qquad x_i^* := \arg\max_{v \in \mathcal{V}} q_\theta^i\big(v \mid \mathbf{x} \oplus x^*_{M_{\succ i}}\big), \qquad \hat{x}_i := \arg\max_{v \in \mathcal{V}} q_\theta^i\big(v \mid \mathbf{x}'\big), \tag{23}$$

be the event where the parallel decode fails to sample from the joint distribution over language: the event that the parallel commit at i, predicted from $\mathbf{x}'$, differs from the target sequential decode replayed in confidence order from $\mathbf{x}$. Within a round both walks are anchored at the same canvas, which is what makes the round a single-pass commit, and we write $F_\theta(i; \mathbf{x}_t, \mathbf{x}_t)$; PIG.EXT below advances the second argument by one round and leaves the first where it is. Given the two anchors and a frozen model q_θ, both decodes are deterministic argmaxes, so F_θ is a deterministic $\{0, 1\}$ quantity; all probabilistic statements below are over the population $\mathcal{D}$ of configurations $(\mathbf{x}_t, S, i)$ that decoding visits.

Aggregating rounds requires saying exactly *which* sequential decode each round is compared against; the following definition makes the reference precise.

Definition A.2 (Re-anchored sequential reference and piece-wise total inconsistency)**.** Run the parallel decoder for rounds $r = 1, \dots, T$ (so $T = \text{NFE}$), producing canvases $\mathbf{x}_{t_1}, \dots, \mathbf{x}_{t_T}$ and committed sets $S_1, \dots, S_T$. The *re-anchored reference* at round r is the sequential decode of S_r launched from the canvas $\mathbf{x}_{t_r}$ that the *parallel* decoder actually reached:

$$x^*_{s_i} \;=\; \arg\max_{v \in \mathcal{V}} \; q_\theta\big(v \;\big|\; \mathbf{x}_{t_r} \oplus x^*_{s_{<i}}\big), \qquad s_i \in S_r, \tag{24}$$

conditioning within the round on its *own* outputs $x^*_{s_{<i}}$, and across rounds on the parallel decoder's canvas. The *piece-wise total inconsistency* of the decode is

$$R_{\text{total}} \;=\; \sum_{r=1}^{T} R(S_r) \;=\; \sum_{r=1}^{T} \sum_{i \in S_r} F_\theta(i; \mathbf{x}_{t_r}, \mathbf{x}_{t_r}), \tag{25}$$

the total count of round-level disagreements against these re-anchored references. R_{total} is *not* the Hamming distance between the parallel output and a from-scratch sequential decode of the whole problem: after a disagreement, both trajectories in every later round condition on the disagreeing token, so the reference forgives past divergence and counts only the fresh disagreement each round introduces.

Definition A.3 ($\bar{\epsilon}$-calibrated inconsistency estimator)**.** Let $\mathcal{T}_i$ be a $\mathcal{G}_i$-measurable summary of the walk's observables and $\mathcal{G}_i \;=\; \sigma\Big(\big(\mathcal{T}_j\big)_{j \preceq i}\Big)$ be the sigma-algebra over all prior observable summaries of the round. A *parallel inconsistency estimator* $\hat{p}$ maps a summary statistic $\mathcal{T}_i$ to $[0, 1]$ and is called $\bar{\epsilon}$*-calibrated* if for every canvas - set pair $(\mathbf{x}_t, S)$, its error is bounded by $\bar{\epsilon}(\mathcal{T})$:

$$\Big| \mathbb{E}_{\mathcal{D}}\big[F_\theta(i; \mathbf{x}_t, \mathbf{x}_t) \;\big|\; \mathcal{G}_i\big] \;-\; \hat{p}\big(\mathcal{T}_i\big) \Big| \;\le\; \bar{\epsilon}(\mathcal{T}). \tag{26}$$

To simplify notation, we write $\bar{\epsilon} = \bar{\epsilon}(\mathcal{T})$.

Calibration asks that the deployed estimate be $\bar{\epsilon}$-close to the truth *conditionally on everything the walk observes*.

Proposition A.4 (PIG.INT: piece-wise parallel consistency over a whole decode)**.** *Let $\hat{p}$ be an $\bar{\epsilon}$-calibrated inconsistency estimator (Definition A.3). Then every round of Algorithm 2 satisfies*

$$\mathbb{E}\big[R(S)\big] \;\le\; \mathbb{E}\big[\hat{R}(S)\big] + \bar{\epsilon}\,\mathbb{E}\big[|S|\big] \;\le\; \varphi + \bar{\epsilon}\,\mathbb{E}\big[|S|\big], \tag{27}$$

and the piece-wise total inconsistency for the whole decode with budget φ satisfies

$$\mathbb{E}\big[R_{\textit{total}}\big] \;\le\; \varphi \cdot \mathbb{E}[T] \;+\; \bar{\epsilon} \cdot N \;=\; \varphi \cdot \text{NFE} \;+\; \bar{\epsilon} \cdot N, \tag{28}$$

where N is the canvas length ($\sum_r |S_r| = N$, every position is committed exactly once). In particular, with a well-estimated table ($\bar{\epsilon} \approx 0$), φ is an exchange rate: at most φ expected disagreements per forward pass spent, uniformly over the decode.

Proof. Because the walk visits candidates in decreasing ν and Equation (5)'s coupling sums only over higher-confidence members ($\nu_j > \nu_i$), the estimated inconsistency $b_i = \hat{p}(\nu_i, \hat{\mu}_i(S))$ assessed at admission time equals its value under the final set: later admissions never change $\hat{\mu}_i(S)$. For each candidate i, the admission event $\{i \in S\}$ is $\mathcal{G}_i$-measurable, so by the tower property and calibration (Definition A.3), with the convention $F_\theta(i; \mathbf{x}_t, \mathbf{x}_t)\,\mathbf{1}\{i \in S\} = 0$ for $i \notin S$,

$$\begin{aligned}
\mathbb{E}\big[R(S)\big] &= \sum_i \mathbb{E}\big[F_\theta(i; \mathbf{x}_t, \mathbf{x}_t)\,\mathbf{1}\{i \in S\}\big] \\
&= \sum_i \mathbb{E}\big[\mathbb{E}[F_\theta(i; \mathbf{x}_t, \mathbf{x}_t) \mid \mathcal{G}_i]\,\mathbf{1}\{i \in S\}\big] \\
&\le \sum_i \mathbb{E}\big[\big(\hat{p}\big(\nu_i, \hat{\mu}_i(S)\big) + \bar{\epsilon}\big)\,\mathbf{1}\{i \in S\}\big] \;=\; \mathbb{E}\big[\hat{R}(S)\big] + \bar{\epsilon}\,\mathbb{E}\big[|S|\big],
\end{aligned}$$

the inequality by Equation (26); the stopping rule of PIG.INT enforces $\hat{R}(S) \le \varphi$ path-wise, which gives Equation (27). For Equation (28), apply Equation (27) to each round and sum: $\sum_r |S_r| = N$ deterministically, the decode is Markov in the canvas, and T is a stopping time of the round sequence, so optional stopping applies to the sum of the per-round bounds. □

Remark A.5. We remark that in Equation (28), the total number of rounds $\mathbb{E}[T]$ is a function of φ. In fact, the total number of NFE's over the whole decode is inversely related to φ.

PIG.EXT compares two parallel policies against one sequential target: committing a window now, alongside the set the walk holds, or committing it one round later, after that set has been revealed. The disagreement of the second policy is the following object.

Fix a canvas $\mathbf{x}_t$, a set $S \subseteq M$ and its window $N(S)$, the h most confident masked positions outside S. For $i \in N(S)$ the *deferred disagreement* is $F_\theta(i; \mathbf{x}_t, \mathbf{x}_t \oplus \hat{x}_S)$, the event that the parallel commit at i, re-predicted once the parallel commits $\hat{x}_S$ are revealed, differs from the target. Its value at i depends on the window only through the members more confident than i. The *deferred inconsistency* of the window is $R(N(S) \mid S) = \sum_{i \in N(S)} F_\theta(i; \mathbf{x}_t, \mathbf{x}_t \oplus \hat{x}_S)$.

Definition A.6 ($(\bar{\epsilon}, \bar{\delta})$-calibrated)**.** An inconsistency estimator $\hat{p}$ is evaluated for the deferred round at a $\mathcal{G}_i$-measurable summary statistic $\mathcal{T}_i'$ formed from the current forward pass; we write $\hat{p}(\mathcal{T}_i')$ for this *deferred estimate* (an estimate formed now, not a quantity observed after S is committed) and $\hat{R}(N(S) \mid S) = \sum_{i \in N(S)} \hat{p}(\mathcal{T}_i')$. The estimator is called $(\bar{\epsilon}, \bar{\delta})$-*calibrated* if it is $\bar{\epsilon}$-calibrated (Definition A.3) and its deferred estimate satisfies, for every canvas - set pair $(\mathbf{x}_t, S)$ and every $i \in N(S)$,

$$\Big| \mathbb{E}_{\mathcal{D}}\big[F_\theta(i; \mathbf{x}_t, \mathbf{x}_t \oplus \hat{x}_S) \;\big|\; \mathcal{G}_i\big] \;-\; \hat{p}\big(\mathcal{T}_i'\big) \Big| \;\le\; \bar{\epsilon} + \bar{\delta}, \tag{29}$$

where $\bar{\delta} \ge 0$ is the excess error of the deferred estimate.

Lemma A.7 (PIG.EXT: the marginal inconsistency)**.** *Let $\hat{p}$ be an $(\bar{\epsilon}, \bar{\delta})$-calibrated inconsistency estimator (Definition A.6). Fix a round at canvas $\mathbf{x}_t$, a set S reached by the walk, and its window $N(S)$. Then widening the round by $N(S)$, instead of granting $N(S)$ one extra forward pass, costs*

$$\mathbb{E}\big[R(S \cup N(S))\big] - \mathbb{E}\big[R(S) + R(N(S) \mid S)\big] \;\le\; \theta + |N(S)|\,(2\bar{\epsilon} + \bar{\delta}). \tag{30}$$

Proof. For $i \in N(S)$ write $\mathcal{T}_i$ for the statistic of i as a member of the widened round $S \cup N(S)$ (Definition A.3), so that $\hat{R}(N(S)) = \sum_{i \in N(S)} \hat{p}(\mathcal{T}_i)$ and $\hat{R}(N(S) \mid S) = \sum_{i \in N(S)} \hat{p}(\mathcal{T}_i')$ are the two summands of PIG.EXT. Calibration at the pair $(\mathbf{x}_t, S \cup N(S))$ (Equation (26)) places $\mathbb{E}_{\mathcal{D}}[F_\theta(i; \mathbf{x}_t, \mathbf{x}_t) \mid \mathcal{G}_i]$ within $\bar{\epsilon}$ of $\hat{p}(\mathcal{T}_i)$, and deferred calibration (Equation (29)) places $\mathbb{E}_{\mathcal{D}}[F_\theta(i; \mathbf{x}_t, \mathbf{x}_t \oplus \hat{x}_S) \mid \mathcal{G}_i]$ within $\bar{\epsilon} + \bar{\delta}$ of $\hat{p}(\mathcal{T}_i')$; subtracting,

$$\Big| \mathbb{E}_{\mathcal{D}}\big[F_\theta(i; \mathbf{x}_t, \mathbf{x}_t) - F_\theta(i; \mathbf{x}_t, \mathbf{x}_t \oplus \hat{x}_S) \;\big|\; \mathcal{G}_i\big] \;-\; \big(\hat{p}(\mathcal{T}_i) - \hat{p}(\mathcal{T}_i')\big) \Big| \;\le\; 2\bar{\epsilon} + \bar{\delta}, \tag{31}$$

the per-token form of the bound, reused in the proof of Proposition A.8. The events $\{i \in N(S)\}$ are $\mathcal{G}_i$-measurable, so the tower property and Equation (31) bound it by $\mathbb{E}\big[\hat{R}(N(S)) - \hat{R}(N(S) \mid S)\big] + |N(S)|\,(2\bar{\epsilon} + \bar{\delta})$. Given the stopping rule of Algorithm 3, we know $\Big|\hat{p}(\mathcal{T}_i) - \hat{p}(\mathcal{T}_i')\Big| \le \theta$ and the claim follows. □

To aggregate Lemma A.7 over a decode we apply Definition A.6 one admission at a time within each round, against the round's own target, with the parallel walk one deferral step deep.

Proposition A.8 (PIG.EXT: piece-wise deferred parallel inconsistency over a whole decode)**.** *Order each round's admissions $s_1, \ldots, s_{|S_r|}$ in decreasing confidence with prefixes $S_{r,k} = \{s_1, \ldots, s_k\}$, and define the round's* cascade floor

$$C(S_r) \;=\; \sum_{k=2}^{|S_r|} F_\theta\big(s_k;\; \mathbf{x}_{t_r},\; \mathbf{x}_{t_r} \oplus \hat{x}_{S_{r,k-1}}\big), \qquad C_{\textit{total}} = \sum_{r=1}^{T} C(S_r), \tag{32}$$

the disagreements its admissions would still suffer, against the round's own target, if each were re-predicted after its prefix. Let $\hat{p}$ be an $(\bar{\epsilon}, \bar{\delta})$-calibrated inconsistency estimator whose per-token deferral terms are nonnegative. Then, Algorithm 3 satisfies

$$\mathbb{E}\big[R_{\textit{total}}\big] \;\le\; \mathbb{E}\big[C_{\textit{total}}\big] \;+\; \big(\theta + 2\bar{\epsilon} + \bar{\delta}\big)\big(N - \mathrm{NFE}\big). \tag{33}$$

Proof. Fix a round and drop r. In the notation of Equation (23) the disagreement at s_k does not record which round contained it, so $R(S) = \sum_{k=1}^{|S|} F_\theta(s_k; \mathbf{x}_t, \mathbf{x}_t)$; the $k = 1$ term vanishes, the first token of a round never being in disagreement, $x^*_{s_1} = \hat{x}_{s_1}$ by Equation (Sequential). For $k \geq 2$, s_k is the most confident member of $N(S_{k-1})$, and its statistics $\mathcal{T}_{s_k}, \mathcal{T}'_{s_k}$ are those of the pair $(S_{k-1}, N(S_{k-1}))$. The admission event $\{s_k \in S\}$ is $\mathcal{G}_{s_k}$-measurable, so multiplying Equation (31), taken at $(S, i) = (S_{k-1}, s_k)$, by its indicator and taking expectations,

$$\mathbb{E}\big[F_\theta(s_k; \mathbf{x}_t, \mathbf{x}_t)\,\mathbf{1}\{s_k \in S\}\big] \leq \mathbb{E}\big[F_\theta(s_k; \mathbf{x}_t, \mathbf{x}_t \oplus \hat{x}_{S_{k-1}})\,\mathbf{1}\{s_k \in S\}\big] + \mathbb{E}\Big[\big(\hat{p}(\mathcal{T}_{s_k}) - \hat{p}(\mathcal{T}'_{s_k})\big)\,\mathbf{1}\{s_k \in S\}\Big] + (2\bar{\epsilon} + \bar{\delta})\,P(s_k \in S).$$

The bracket is the term of s_k in the estimated gap of S_{k-1}; the remaining terms of that gap are nonnegative by hypothesis, so on $\{s_k \in S\}$ it is at most $\hat{R}(N(S_{k-1})) - \hat{R}(N(S_{k-1}) \mid S_{k-1}) \leq \theta$ by Algorithm 3's admission rule. Summing over $k \geq 2$, $\mathbb{E}[R(S)] \leq \mathbb{E}[C(S)] + (\theta + 2\bar{\epsilon} + \bar{\delta})\,\mathbb{E}\big[|S| - 1\big]$; summing over rounds, $\sum_r (|S_r| - 1) = N - T$, and optional stopping applies as in Proposition A.4. □

B THE ENTROPY FINITE DIFFERENCES ESTIMATOR: IMPLEMENTATION AND TRAINING

This appendix is self-contained: it specifies what the estimator of Figure 4 consumes, what it is made of, what it is trained on, and how it was trained, for the LLaDA-8B-Instruct instantiation used in every LLaDA experiment of this paper; the Nemotron-Diffusion-3B Instruct twin is described at the end by its differences. Every number is read from the harvest and training code and the shipped checkpoint, not from a design document.

B.1 THE FINITE-DIFFERENCE ESTIMATOR

What the estimator predicts. For a canvas state y_t with masked positions $\mathcal{M}$, the target is the directed *entropy finite difference*

$$\hat{\mathbf{D}}_{ij} \;\approx\; H_i(y_t) \;-\; H_i\big(y_t \text{ with position } j \text{ committed to its argmax}\big), \qquad i \in \mathcal{M},\; j \in \mathcal{M},\; i \neq j, \tag{34}$$

the change in the model's own entropy at receiver i when source j is committed — positive when committing j makes i more certain, negative when it makes i less certain. A single forward pass of the frozen model with j committed measures one full column $\delta_{\cdot j}$ exactly; the estimator's job is to return the whole matrix from the *one* forward pass the decoder already runs on y_t, so that the two set statistics of Algorithm 2 — the aggregate coupling $\mu_i(P)$ and the local curvature $\hat{H}(P)$ — cost no extra model evaluations.

Inputs (per canvas position, from the frozen backbone). The backbone is never trained, never grad-enabled and never modified: it runs in evaluation mode under `no_grad`, and a forward hook on the final normalisation layer (the input to the language-model head) reads the post-norm hidden state $h_i \in \mathbb{R}^{4096}$ for every canvas position. The estimator also receives the canvas token id at each position (the mask token at masked positions), the two attention maps $A_t, A_{t-1} \in \mathbb{R}^{256\times256}$ of the current and previous round, and nine per-position scalars: the entropy H_i and confidence ν_i of the position's predictive distribution, the top-1/top-2 probability margin, a decoded flag, a three-way structural class (decoded / committable / other) as a one-hot, and the decode fraction t. The canvas length is fixed at $L = 256$ throughout.

Architecture (≈2.2M parameters). Per position, three linear maps embed the inputs — h_i through $4096 \to 192$, the token embedding through $4096 \to 64$, the nine scalars through $9 \to 32$ — and a fused linear layer with LayerNorm produces $z_i^{(0)} \in \mathbb{R}^{192}$. Two pre-norm transformer layers over the 256 positions follow ($d = 192$, six heads of dimension 32, rotary relative positions, GELU feed-forward of width 768), each modulated by FiLM conditioning on t (a two-layer MLP maps t to a per-channel scale and shift). A pairwise *attention-bias* MLP maps, for every ordered pair (i, j), the five features $[A_t[i,j], A_t[j,i], A_{t-1}[i,j], A_{t-1}[j,i], \log(1+|i-j|)]$ through $5 \to 16 \to 7$: six outputs are added to the six heads' attention logits, the seventh is an additive edge bias b_{ij} on the readout. The readout is bilinear and directed: a receiver map $f : \mathbb{R}^{192} \to \mathbb{R}^{96}$ and a source map $g : \mathbb{R}^{192} \to \mathbb{R}^{96}$ (each a two-layer GELU MLP) and a learned $W \in \mathbb{R}^{96\times96}$ give

$$\hat{\mathbf{D}}_{ij} \;=\; \frac{f(z_i)^\top W\, g(z_j)}{\sqrt{96}} \;+\; b_{ij}, \tag{35}$$

computed for all pairs at once as one 256×256 matrix product. Two different maps for the two roles are what make $\hat{\mathbf{D}}$ directed ($\hat{\mathbf{D}}_{ij} \neq \hat{\mathbf{D}}_{ji}$); the $1/\sqrt{96}$ is the usual dot-product normalisation that keeps the readout at unit scale at initialisation and has no expressive role.

Cost as a function of canvas length. With L canvas positions the per-position projections cost $O(L)$ ($\approx L \times 1.05$M multiply-adds, dominated by the $4096 \to 192$ and $4096 \to 64$ maps), the two transformer layers $O(L\,d^2 + L^2 d)$ with $d = 192$, and the pairwise terms — the attention-bias MLP over all (i, j), the bilinear readout $FWG^\top$ and the edge bias — $O(L^2)$ with small constants ($\approx 192 + 96$ multiply-adds per pair). At $L = 256$ the total is ≈0.6 GMAC (≈1.2 GFLOP) per round, of which the L-linear projections are about half; the L^2 terms overtake them only beyond $L \approx 10^3$. Memory is $O(L^2)$ for the two attention maps and the bias tensor. Measured in the

production decoder on a shared H100 (the node was not reserved, so this is an upper bound), the estimator adds 12.8 ms per decoding round against 17.9 ms for the attention-input U-Net it replaced, inside the 15 ms per-round budget the decoder is held to.

Training data: states. States are harvested by decoding *training-side* problems with the frozen model and snapshotting the canvas every 24 commits, so that the first snapshot of a decode is taken at $t = 24/256 \approx 0.09$ and the last near $t \approx 0.94$. Four corpora, all training splits: GSM8K, MATH (the Hendrycks training set), OpenCoder, and MBPP-train; no test set of any benchmark enters the harvest. Two decoding policies generate the states: the model's own one-token-per-round confidence-greedy decode, and confidence-greedy-to-budget set decodes at $\theta \in \{25, 50\}$, each on the free canvas and with eligibility restricted to the leftmost 64 positions — so the estimator sees the states that budgeted parallel decoding actually visits, in both block regimes. In total 34,142 states over 3,721 distinct problems, carrying ≈111M labelled directed edges; the mixture by corpus, in problems, states and edges:

corpus	problems	%	greedy singles	budgeted sets	states	%	edges (%)
GSM8K (train)	1,111	29.9	6,750	3,162	9,912	29.0	32.4M (29.2)
MATH (train)	1,230	33.1	6,750	4,276	11,026	32.3	35.7M (32.2)
OpenCoder	1,110	29.8	6,750	3,233	9,983	29.2	32.3M (29.1)
MBPP (train)	270	7.3	1,620	1,601	3,221	9.4	10.4M (9.4)
total	3,721	100	21,870	12,272	34,142	100	111M (100)

Edge counts are |eligible receivers| × 24 per state and vary with t; the percentages show that the mixture is balanced in problems, states and edges alike except for MBPP. No state with a fully masked canvas ($t = 0$) is in the training set: the first round of every decode — prompt plus an all-mask span — is outside the harvested range, and the estimator extrapolates to it through its t-conditioning. The held-out evaluation of Figure 7 does include $t = 0$ states (its first bin), where the estimator tracks the gold local curvature as it does elsewhere; adding a $t = 0$ snapshot to the harvest is a one-line change we have not made.

Splits are by *problem*, never by state: the hash of (corpus, problem id) modulo 10 assigns 0–7 to training, 8 to validation and 9 to an internal test split (27,231 / 3,468 / 3,443 states), so no problem contributes states to two splits. The MBPP arm is about a quarter the size of the other three, so the code side of the mixture is dominated by OpenCoder.

Training data: labels. Each state costs about 28 additional forward passes of the frozen model. Twenty-four *single plants* — the 16 most confident eligible positions and 8 eligible positions drawn at random, to break the selection bias of confidence — each commit one source j and re-run the model, giving the exact column $\delta_{\cdot j}$ of Equation (34) at every eligible receiver: ≈232 × 24 ≈ 3,300 labelled directed edges per state, ≈111M in total. The loss is evaluated *only* on this labelled block — the eligible receivers of the 24 planted sources — never on the full 256 × 256 output: entries whose source was not planted have no label and receive no gradient, so the estimator is supervised on ≈5% of the matrix it emits, at every state. Three *joint plants* of sizes 4, 8 and 12 (confidence-greedy and random sets alternating) give the gold non-additive curvature $|H(S)|$ of a set; these are used for evaluation and diagnostics only and never enter the loss. Each state is stored once in fp16 (≈2.3 MB).

Objective and optimisation. The shipped model is the plainest recipe in the training ladder: a Huber loss ($\delta_{\mathrm{Huber}} = 0.2$, comparable to the typical label magnitude of 0.1–0.2 nats) on the labelled edges only, unweighted, with no aggregate-mass term and no auxiliary head. AdamW with learning rate 3×10^{-4} (warm-up, then cosine decay to 3×10^{-5}), weight decay 0.01, gradient clipping at 1.0, gradient accumulation over 16 states per step, ten epochs configured; the model is selected by validation per-edge correlation at the end of every epoch and training stops when it fails to improve. The shipped checkpoint is the epoch-7 model, validation per-edge correlation 0.827. Held-out quality on the four *test* benchmarks — per-edge, aggregate coupling and local curvature — is reported in Figure 6 (Figure 7 shows the same evaluation as gold-versus-estimate scatter and as recovered edge mass by magnitude).

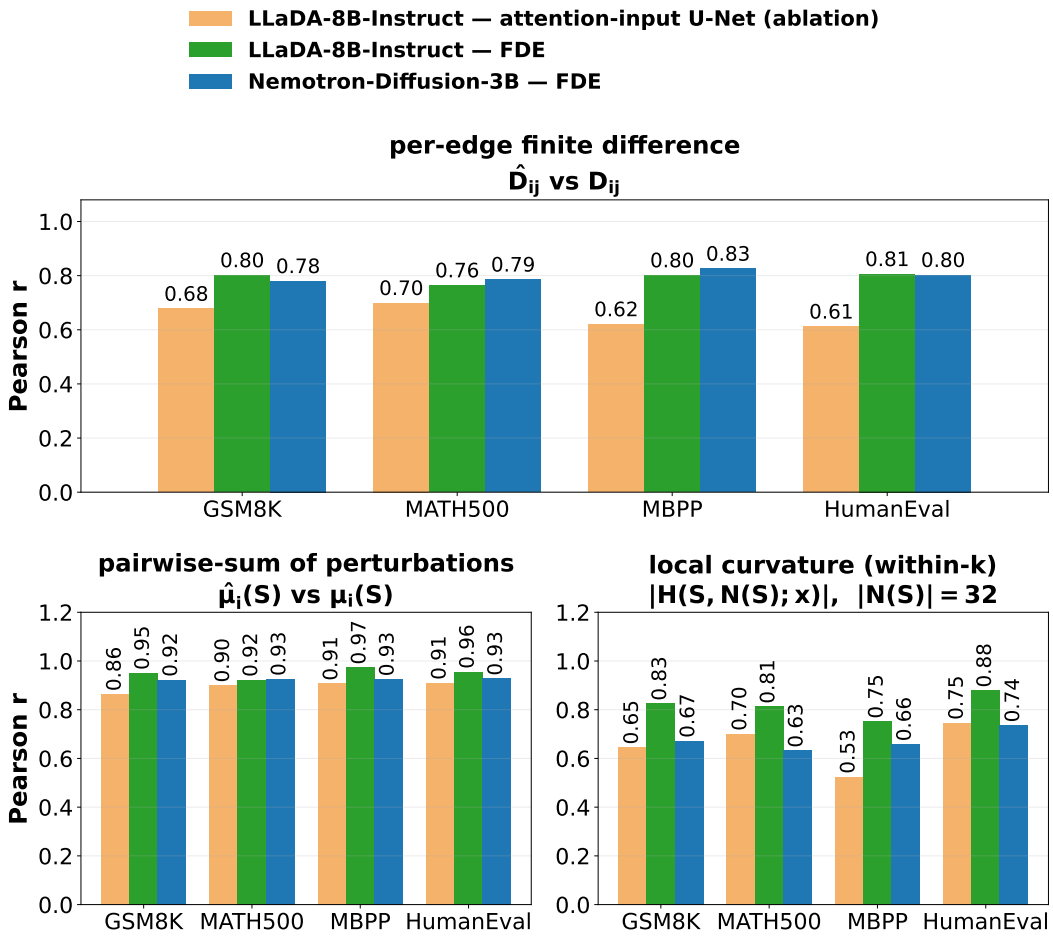

Figure 6: **Estimator quality on held-out test states.** Pearson correlation between the exact quantity and its estimate, per benchmark, for the three quantities the decoders consume: the per-edge finite difference $\hat{\mathbf{D}}_{ij}$ against $\mathbf{D}_{ij}$ (left), the aggregate coupling $\hat{\mu}_i(S)$ along the confidence walk (middle), and the local curvature $|H(S, N(S); \mathbf{x})|$ over the next 32 candidates (right; median over set sizes of the per-size correlation). Green: the LLaDA-8B-Instruct estimator; orange: the attention-input U-Net it replaced (ablation); blue: the Nemotron-Diffusion-3B twin on block-64 test states.

Algorithm 4 states the whole procedure in one place. Its two halves are the two costs of this estimator. The harvest is the expensive one: every stored state pays one forward pass to record the model's own readouts and then twenty-four more, one per planted source, each committing a single position and re-scoring the canvas so that an exact column of the finite-difference matrix can be read off. Nothing is approximated at this stage; the labels are differences of entropies the frozen model actually produced. The fit is the cheap one, a Huber regression of the adapter's output onto that labelled block, and its only subtlety is that the loss touches only the planted columns, so roughly nineteen twentieths of the matrix the adapter emits is never supervised and is held to account solely by the held-out evaluation.

The Nemotron-Diffusion-3B twin. The same architecture and recipe are re-instantiated for the block-causal Nemotron-Labs-Diffusion-3B with three differences forced by the model. (i) The hidden state is 3072-dimensional and is read from the model's own last-hidden-state output rather than a hook (its diffusion head is applied to that output, so this reproduces the logits exactly); the two input projections are re-dimensioned accordingly. (ii) The model never attends across blocks, so the estimator operates on one block: the transformer layers run over the block's positions with block-relative rotary positions, the structural class adds a state for the autoregressively seeded block head,

the pairwise bias MLP takes three features, and states are harvested at block lengths $\{8, 32, 128\}$ stratified over problems — block length 64 is deliberately never harvested and is the held-out generalisation geometry (it is also the sliding-window operating block). (iii) Plants re-run only the current block against the cached prefix, so a fully planted state costs roughly 0.5 s instead of ≈ 5 s. Corpora, split rule (by problem), plant recipe (16 top-confidence + up to 8 random sources), Huber-0.2 objective and optimiser are unchanged. The harvest holds $24{,}167$ states; the shipped checkpoint is the best-validation epoch (epoch 7), at a validation edge correlation of 0.794 — read from the checkpoint's stored metadata, as for LLaDA.

B.2 The calibrated mismatch table

The estimator of Section B.1 returns finite differences; the decoder needs probabilities. Both decoders obtain them from a second, far simpler object: a table of empirical mismatch rates indexed by two features of Equation (6), the committing position's aggregate coupling $\hat{\mu}$ to the positions already accepted and either its confidence ν, giving the table $\hat{p}_\nu$ that PIG.INT reads (Figure 8), or its entropy H, giving the table $\hat{p}_H$ that PIG.EXT reads (Figure 10). Neither table is a model. Each is a binned average of labelled commits, frozen once per backbone and read by lookup at decode time. We describe $\hat{p}_\nu$ first, then the two points on which $\hat{p}_H$ differs.

Labels and the data mixture. The labels come from sequential-prefix replay, and from a *different* corpus mixture than the estimator's. Algorithm 5 gives the procedure. Training-split problems from two corpora only, GSM8K-train and OpenCoder, are decoded one token per round in decreasing confidence, so that every state the probe visits is one the decoder would actually reach. Every sixteenth commit the probe stops, builds the production one-pass graph at that canvas, and assembles about thirty candidate sets from six samplers: confidence-greedy sets grown until their coupling crosses a cut, top-K and bottom-K by confidence, uniform random sets, sets grown greedily along the largest edges, and mixtures of the top half by confidence with a random half. The spread is the point: a table fitted only on the sets a budget would choose would have no data in the cells a budget must be able to price in order to refuse them. Each set is then replayed sequentially from that same canvas, and every member yields one row, its confidence, its aggregate coupling to the members ahead of it, and a binary label for whether the parallel commit survived the replay. No test split of any benchmark is touched. The LLaDA table is fitted on $792{,}841$ labelled commits, the Nemotron table on $601{,}730$. The narrower mixture is deliberate: the table has two input features and a few dozen cells, so it saturates on far less data than the estimator's 111M labelled edges, and the two corpora were chosen to bracket the math and code sides of the evaluation.

Binning. Confidence is cut at the fixed edges 0, 0.2, 0.4, 0.6, 0.7, 0.8, 0.9, 0.95, 1, eight bins that tighten where admission actually happens. The coupling axis has no natural scale, so its five bins are cut at the 20th, 40th, 60th and 80th percentiles of the *positive* couplings observed in the probe, with the top edge at the 99.9th percentile of all of them; a cell holding fewer than 50 commits falls back to the pooled rate of its confidence row, which is why no cell is ever read off a handful of observations. The cell value is the raw mismatch frequency, with no smoothing, shrinkage or parametric fit, so the table carries no assumption about how risk varies between cells and can express the non-monotone coupling profile we observe below $\nu = 0.6$. Two consequences are worth stating. The coupling edges are quantiles of *the estimator's own* predicted couplings rather than of gold ones, so the table is matched to the estimator it will be read beside and must be refitted whenever that estimator changes. And because the probe advances one token per round while the sets it scores are much wider, the table is fitted on states a fast decoder reaches but on sets a slow decoder would never commit, which is what lets a single table price both the budget's acceptances and its refusals.

Why the table is fitted per model. The coupling edges are model-specific and differ substantially: $0.26/0.87/2.38/5.91/35.4$ on LLaDA against $0.13/0.76/2.47/5.56/26.7$ on Nemotron, because Nemotron's edge magnitudes are smaller. Levels differ too, with Nemotron mismatching more at low confidence (0.363 against 0.265 in the cheapest cell). A table fitted on one backbone mis-prices the other, so each backbone is decoded with its own; Figure 8 shows both.

Corpus transfer, and one transfer we assume rather than measure. Figure 9 shows the two per-corpus fits and their difference. In the decision region ($\nu \geq 0.6$), where a budget actually spends

itself, the largest absolute disagreement between the GSM8K-train and OpenCoder fits is 0.024 on LLaDA and 0.028 on Nemotron, which is why the table we use pools them. Differences above 0.05 occur only at $\nu < 0.4$, where math tokens mismatch more than code tokens at matched coupling, reaching -0.18 in the LLaDA $[0, 0.2) \times [0.25, 0.86)$ cell. With 0.3–0.6M commits per corpus even a difference of 0.005 is statistically significant, so the agreement above is a statement about size, not about noise. Regime transfer is a weaker claim: the LLaDA table was fitted on free-canvas states and the Nemotron table on a sliding 32-token window, and both are then applied at every block geometry we report. We assume that transfer; we have not measured it.

The entropy-indexed table $\hat{p}_H$. PIG.EXT prices each upcoming candidate $i \in N(S)$ twice, at its current entropy H_i and at the entropy it is predicted to have once S is committed, $\hat{H}_i(\mathbf{x} \mid S) = H_i - \max_{j \in S} \hat{\mathbf{D}}_{ij}$ (Equations (8) and (9)). The second argument is a prediction on the entropy axis to which no confidence corresponds, so the lookahead needs the mismatch rate indexed by the entropy of the committing position's predictive distribution rather than by its confidence. Figure 10 shows the resulting table for both backbones. It is fitted by Algorithm 5 with two changes. The row feature is the receiver's entropy H_{s_k} at the probed canvas, in nats, cut at the fixed edges 0, 0.1, 0.25, 0.5, 1, 1.75, 2.5, 3.5, 4.5, 6 and ∞, ten bins that tighten at low entropy where the confident candidates sit (a position at confidence 0.8 typically has an entropy near 1 nat); and the coupling edges are re-derived on the table's own rows by the same quantile rule, so on LLaDA they differ slightly from those of $\hat{p}_\nu$ $(0.26/0.89/2.45/6.03/36.0)$ and on Nemotron they coincide with them. The thin-cell rule is unchanged. The top row, $H \geq 6$ nats, holds no commit on LLaDA and a single one on Nemotron, since receiver entropies at probe states stay below about 5.5 nats; it is filled with zeros on LLaDA and with the row below it on Nemotron, is not read in practice, and is not drawn. The rows are labelled commits of the same probe. On Nemotron they are the same $601{,}730$ commits: the probe was re-run recording each member's entropy alongside its confidence, and reproduced the confidence table's probe exactly. On LLaDA the confidence probe did not record entropy, so $\hat{p}_H$ is fitted on a separate probe of the same procedure over 64 problems per corpus, $395{,}854$ commits, half the size of the confidence probe.

Two checks. Held out by problem (fitted on the even-numbered problems of each corpus, scored on the odd), the entropy table is calibrated on both backbones, with a calibration slope of 1.02 on LLaDA and 1.00 on Nemotron, at a Brier score of 0.148 and 0.129 against 0.137 and 0.120 for the confidence table on the same rows. Confidence is therefore the slightly sharper single axis for the within-round price, which is why PIG.INT keeps it, and entropy is the axis the lookahead requires. Corpus transfer is weaker than for $\hat{p}_\nu$ and sits elsewhere: the two per-corpus fits agree within 0.054 on LLaDA and 0.041 on Nemotron below 1.75 nats, but at high entropy code tokens mismatch less than math tokens at matched coupling, by up to 0.19 on LLaDA between 2.5 and 4.5 nats (and 0.29 in one cell above 4.5 nats that holds only 52 GSM8K-train commits) and up to 0.17 on Nemotron above 4.5 nats. Unlike the confidence table, whose decisions fall almost entirely where the corpora agree, the lookahead sums over the next h candidates whatever their entropy, so this disagreement can reach a decision. Every PIG.EXT result in the paper uses the pooled table, which averages the two corpora at their commit counts; we have not isolated the effect of the disagreement on the decoded frontier.

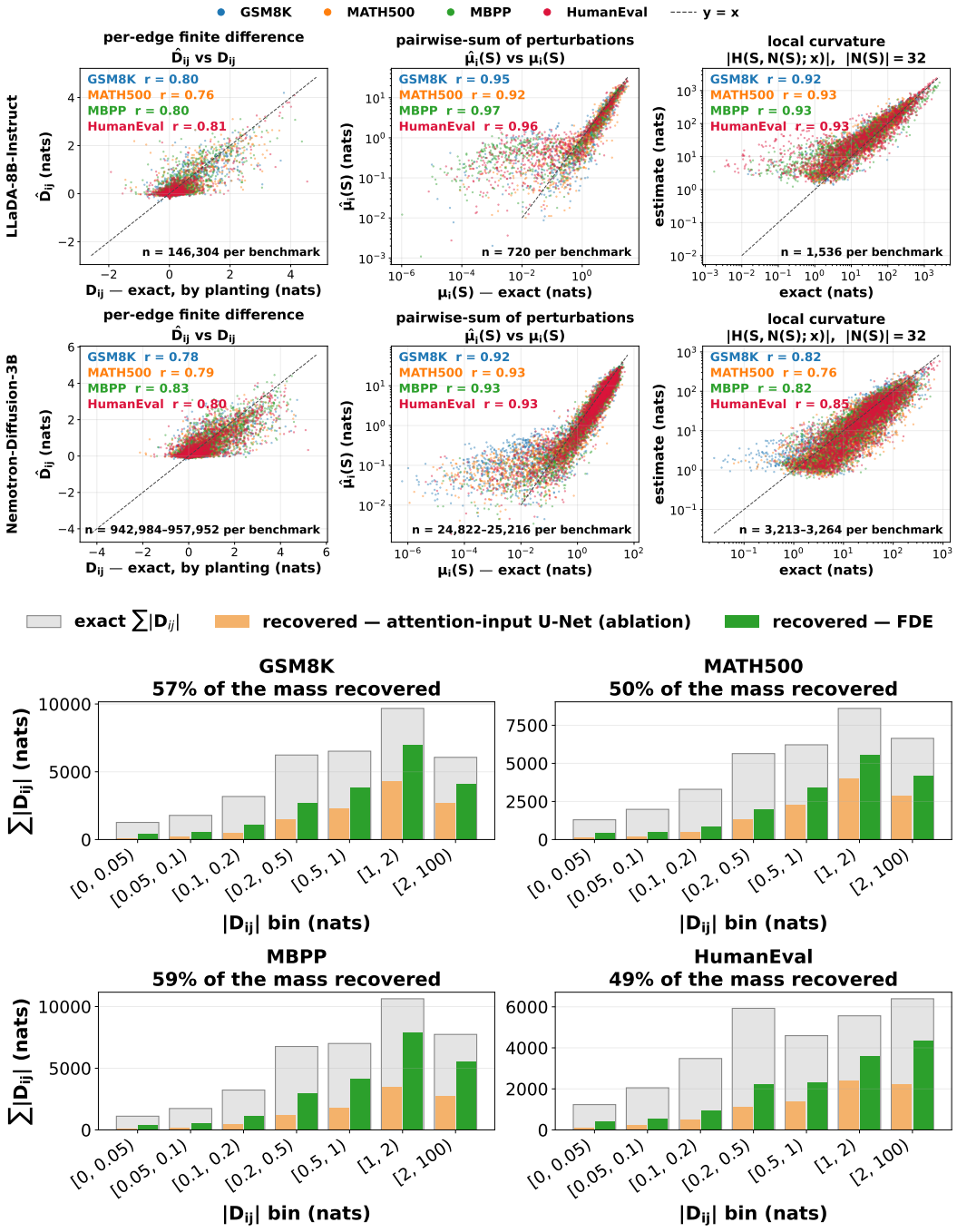

Figure 7: **The same evaluation in detail.** Top: exact (x) versus estimate (y) on test states, one colour per benchmark, LLaDA-8B-Instruct (upper row) and Nemotron-Diffusion-3B (lower row), with the per-benchmark correlation printed in each panel; the curvature correlation here pools set sizes $|S| \in \{2, \ldots, 24\}$ and is therefore higher than the within-size value of Figure 6. Bottom: on LLaDA test states, the exact mass $\sum |\mathbf{D}_{ij}|$ in each magnitude bin (grey) and the sign-consistent mass the estimator recovers (green; 49–59% of the total, the U-Net's 26–34% in orange): the estimator under-shoots in magnitude but does so uniformly across bins.

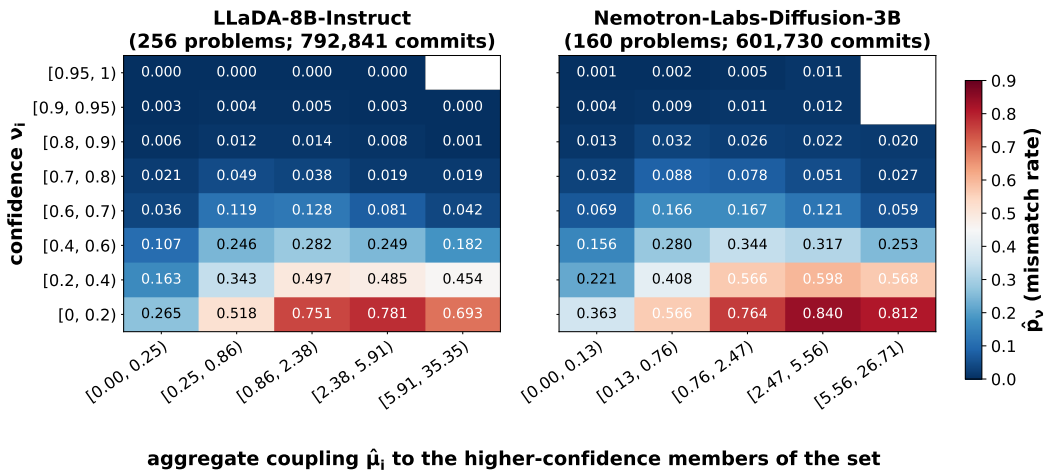

Figure 8: **The calibrated mismatch table $\hat{p}_\nu$ read by PIG.INT**, one per backbone (left LLaDA-8B-Instruct, right Nemotron-Labs-Diffusion-3B). Rows: the committing position's confidence ν_i at the fixed edges of Algorithm 5; columns: its aggregate coupling $\hat{\mu}_i$ to the higher-confidence members of the set, at the quantile edges of each model's own fit. A cell is the empirical rate at which a parallel commit differed from its sequential replay, pooled over the GSM8K-train and OpenCoder probes; a cell holding fewer than 50 commits is left blank and is read by the decoder at the pooled rate of its row.

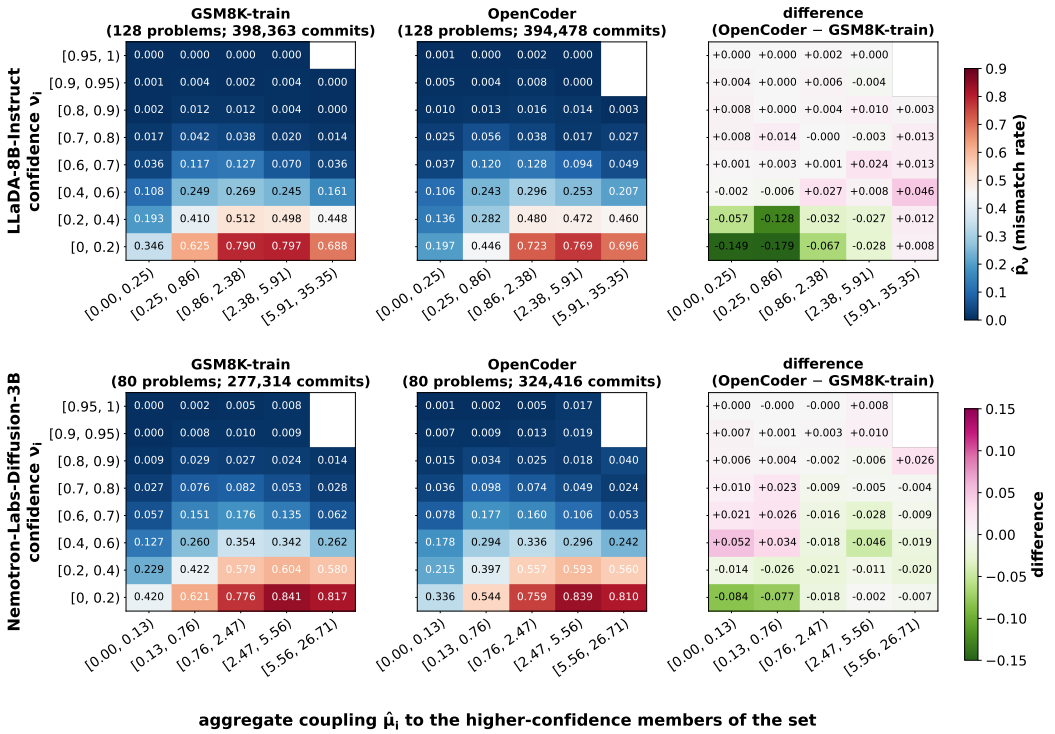

Figure 9: **Corpus transfer of $\hat{p}_\nu$.** The table of Figure 8 fitted on each training corpus alone, GSM8K-train (left) and OpenCoder (middle), at the pooled bin edges, and their difference (right; green where code mismatches less than math at matched coupling, magenta where more). Top row LLaDA-8B-Instruct, bottom row Nemotron-Labs-Diffusion-3B. A cell is drawn only where both corpora hold at least 50 commits.

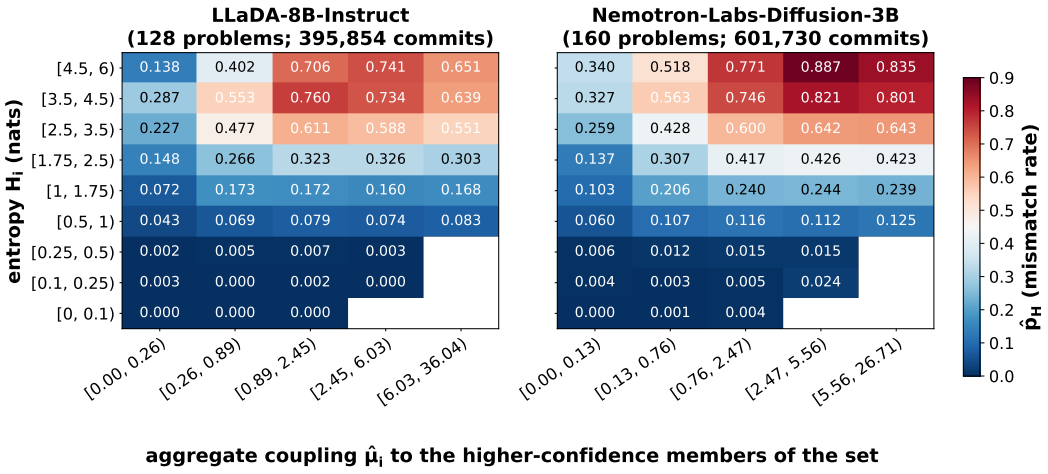

Figure 10: **The entropy-indexed mismatch table $\hat{p}_H$ read by PIG.EXT**, one per backbone (left LLaDA-8B-Instruct, right Nemotron-Labs-Diffusion-3B). As Figure 8, with the committing position's entropy H_i in nats as the row feature, at the fixed edges given in the text; the empty top row ($H_i \geq 6$) is not drawn. Fitted on the labelled commits of the same probe procedure, pooled over GSM8K-train and OpenCoder (Nemotron: the same 601,730 commits as Figure 8; LLaDA: a separate probe of 395,854); a cell holding fewer than 50 commits is left blank.

C ALGORITHMS

The three boxes below state the decoders of Section 4 in full. Algorithm 1 is the outer loop shared by every method in the paper: one forward pass per round, a selection rule that returns the set S to commit, and a parallel commit. Algorithm 2 is PIG.INT, the selection rule that spends the single-round regret budget φ of Equation (2) along the confidence order; Algorithm 3 is PIG.EXT, the rule that widens the round until the predicted cross-round exceeds θ. Both read the same one-pass quantities — the argmax predictions $\hat{x}_i$, confidences ν_i, entropies H_i, and the adapter's finite-difference matrix $\hat{\mathbf{D}}$ of Equation (4) — and the same calibration tables $\hat{p}_\nu, \hat{p}_H$ of Equation (6); they differ only in which regret their single knob bounds, and are alternatives rather than stages of one decoder. Fast-dLLM is Algorithm 1 with the selection rule $S = \{i \in M : \nu_i \geq \nu\}$.

Algorithm 1 Iterative parallel decoding (outer loop, shared by every decoder).

Require: prompt tokens, canvas length N, selection rule SELECT $\in \{\text{PIG.INT}(\varphi),\ \text{PIG.EXT}(\theta),\ \text{Fast-dLLM}(\nu)\}$; the adapter and the calibration tables $\hat{p}_\nu, \hat{p}_H$, fitted once per backbone

$\mathbf{x}_0 \leftarrow$ prompt followed by N mask tokens; $M \leftarrow$ its masked positions; $t \leftarrow 0$
while $M \neq \varnothing$ **do**
  Phase 1: one forward pass
  $q_\theta^i(\cdot \mid \mathbf{x}_t)$ for $i \in M$ from the backbone; $\hat{x}_i \leftarrow \arg\max_v q_\theta^i(v \mid \mathbf{x}_t)$, $\nu_i \leftarrow \max_v q_\theta^i(v \mid \mathbf{x}_t)$, $H_i \leftarrow H\big(q_\theta^i(\cdot \mid \mathbf{x}_t)\big)$
  $\hat{\mathbf{D}} \leftarrow$ ADAPTER(backbone features) ▷ one-pass estimate of Equation (4), $\hat{\mathbf{D}} \in \mathbb{R}^{|M|\times|M|}$
  Phase 2: select the set to commit
  $S \leftarrow \text{SELECT}\big(M,\ \nu,\ H,\ \hat{\mathbf{D}}\big)$ ▷ Algorithm 2 or Algorithm 3
  Phase 3: commit in parallel
  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t \oplus \hat{x}_S$; $M \leftarrow M \setminus S$; $t \leftarrow t+1$
end while
return $\mathbf{x}_t$ ▷ t forward passes; TPF $= N/t$

Algorithm 2 PIG.INT — selection under the single-round regret budget φ.

Require: masked positions M; confidences ν_i; adapter matrix $\hat{\mathbf{D}}$; budget φ; table $\hat{p}_\nu$

$\bar{\mathbf{D}}_{ij} \leftarrow \max(|\hat{\mathbf{D}}_{ij}|, |\hat{\mathbf{D}}_{ji}|)$ for $i, j \in M$ ▷ Equation (5)
$S \leftarrow \varnothing$; $B \leftarrow \varphi$
for $i \in M$ in decreasing ν_i **do**
  $\hat{\mu}_i(S) \leftarrow \sum_{j\in S} \bar{\mathbf{D}}_{ij}$ ▷ aggregate influence of the set already accepted
  $b_i \leftarrow \hat{p}_\nu\big(\nu_i,\ \hat{\mu}_i(S)\big)$ ▷ predicted probability that $\hat{x}_i \neq x_i^\star$, Equation (6)
  if $b_i \leq B$ **then**
    $S \leftarrow S \cup \{i\}$; $B \leftarrow B - b_i$ ▷ the token fits the remaining budget
  else
    break ▷ the round closes at the first candidate that does not fit
  end if
end for
return S

In Algorithm 3 there is no admission test: the statistic closes the round, it never refuses a token, and the token that first pushes the predicted gain of waiting past θ stays committed. Since $N(S)$ is re-read after every admission, near the end of a canvas it is clipped by the pool boundary, the gain of waiting shrinks, and the walk simply runs out the pool.

The two boxes below are offline and are run once per backbone, before any decoding. They produce the two objects every decoder above reads: Algorithm 4 fits the adapter that returns $\hat{\mathbf{D}}$ in one pass, and Algorithm 5 fits the mismatch tables $\hat{p}_\nu$ and $\hat{p}_H$ that convert a confidence, or an entropy, and an aggregate coupling into a probability. Neither is touched at decode time. Section B discusses both.

Algorithm 3 PIG.EXT — selection under the cross-round regret budget θ.

Require: masked positions M; confidences ν_i; entropies H_i; adapter matrix $\hat{\mathbf{D}}$; window size h; budget θ; table $\hat{p}_H$

$S \leftarrow \varnothing$
for $j \in M$ in decreasing ν_j **do**
  $S \leftarrow S \cup \{j\}$ ▷ admit first: the round widens by the next most confident token
  $N(S) \leftarrow$ the h most confident positions of $M \setminus S$
  $\widehat{H}_i(\mathbf{x} \mid S) \leftarrow H_i - \max_{k \in S} \hat{\mathbf{D}}_{ik}$ for $i \in N(S)$ ▷ entropy after committing S, Equation (8)
  $\hat{C}(S) \leftarrow \sum_{i \in N(S)} \left[\hat{p}_H\big(H_i, \hat{\mu}_i(S \cup N(S))\big) - \hat{p}_H\big(\widehat{H}_i(\mathbf{x} \mid S), \hat{\mu}_i(S \cup N(S))\big) \right]$ ▷ Equation (9)
  if $\hat{C}(S) > \theta$ **then**
    break ▷ waiting one round is predicted to remove more than θ regret on $N(S)$; j stays in S
  end if
end for
return S

Algorithm 4 Fitting the entropy finite differences estimator (offline, once per model).

Require: frozen dLLM p_θ; training-side problems $\mathcal{D}$; decoding policies Π (confidence-greedy singles; confidence-greedy-to-budget sets at $\theta \in \{25, 50\}$, free canvas and leftmost-64 block); snapshot period $s = 24$; sources per state 16 top-confidence $+ 8$ random

Harvest (states and exact labels)

for each problem $q \in \mathcal{D}$ and policy $\pi \in \Pi$ **do**
  decode q with p_θ under π; every s commits, snapshot the canvas y
  for each snapshot y, with masked positions C **do**
    forward $p_\theta(y)$: hidden states h_i, token ids, attention maps A_t, A_{t-1}, scalars $(H_i, \nu_i, \ldots, t)$ for all i; $H_i(y)$ for $i \in C$
    $S \leftarrow$ the 16 most confident $i \in C \cup 8$ further $i \in C$ drawn at random
    for each source $j \in S$ **do** ▷ one extra forward per source
      $y^{(j)} \leftarrow y$ with j committed to $\hat{x}_j$; forward $p_\theta(y^{(j)})$
      $\delta_{ij} \leftarrow H_i(y) - H_i(y^{(j)})$ for all $i \in C \setminus \{j\}$ ▷ exact column $\delta_{\cdot j}$, Equation (34)
    end for
    store $\big(h, \text{ids}, A_t, A_{t-1}, \text{scalars}, C, S, \{\delta_{ij}\}\big)$ in fp16
  end for
end for
split the stored states by *problem*: $\mathrm{md5}(\text{corpus:}q) \bmod 10 \in \{0..7\}$ train, 8 validation, 9 test

Fit

initialise the adapter g_ϕ (Equation (35)); AdamW, lr 3×10^{-4} warm-up then cosine to 3×10^{-5}, weight decay 0.01, clip 1.0
for epoch $= 1, \ldots, 10$ **do**
  for each training state, in batches of 16 states per optimiser step **do**
    $\hat{\delta} \leftarrow g_\phi(h, \text{ids}, A_t, A_{t-1}, \text{scalars})$ ▷ one $L \times L$ matrix
    $\mathcal{L} \leftarrow \frac{1}{|C|\,|S|} \sum_{i \in C} \sum_{j \in S,\, j \neq i} \mathrm{Huber}_{0.2}\big(\hat{\delta}_{ij} - \delta_{ij}\big)$ ▷ labelled block only; no gradient elsewhere
    gradient step on ϕ
  end for
  evaluate the per-edge correlation $\mathrm{corr}(\hat{\delta}, \delta)$ on the validation states; keep the checkpoint if it improved, stop if it has not
end for
return the best-validation checkpoint (epoch 7, $r = 0.827$ for LLaDA-8B-Instruct)

Algorithm 5 Fitting the calibrated mismatch table $\hat{p}_\nu$ (offline, once per model). The entropy-indexed table $\hat{p}_H$ follows the same procedure with each member's entropy H_{s_k} in place of its confidence ν_{s_k} and the entropy edges of Section B.2 in place of CT.

Require: frozen dLLM q_θ; the fitted estimator g_ϕ of Algorithm 4; training-side problems $\mathcal{D}$ from GSM8K-train and OpenCoder; probe period 16 commits; samplers Σ; thin-cell floor $\tau = 50$

Probe (labelled commits, on the decoder's own state distribution)

for each problem $q \in \mathcal{D}$ **do**
  decode q committing *one* position per round, the most confident ▷ every visited state is on-distribution
  for every 16th commit, at canvas $\mathbf{x}$ with eligible positions M **do**
    $\hat{\mathbf{D}} \leftarrow g_\phi(\mathbf{x})$ ▷ the production one-pass graph
    build ≈30 candidate sets $S \subseteq M$ with Σ (confidence-greedy to a coupling cut; top-K; bottom-K by confidence; random-K; greedy max-edge; mixed), deduplicated by membership
    for each set $S = (s_1, \ldots, s_{|S|})$ in decreasing confidence **do**
      replay S sequentially from $\mathbf{x}$: $x^\star_{s_k} = \arg\max_v q_\theta\big(v \mid \mathbf{x} \oplus x^\star_{s_{<k}}\big)$
      for each member $s_k \in S$ **do**
        emit the row $\big(\nu_{s_k},\ \hat{\mu}_{s_k}(S_{<k}),\ \mathbf{1}\{x^\star_{s_k} \neq \hat{x}_{s_k}\}\big),\ \hat{\mu}_{s_k}(S_{<k}) = \sum_{j \in S_{<k}} \max\big(|\hat{\mathbf{D}}_{s_k j}|, |\hat{\mathbf{D}}_{j s_k}|\big)$
      end for
    end for
  end for
end for

Fit (bin, average, freeze)

CT $\leftarrow (0,\ 0.2,\ 0.4,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 0.95,\ 1]$ ▷ fixed; tighter where admission happens

MT $\leftarrow \big(0,$ the $20, 40, 60, 80$th percentiles of $\{\hat{\mu} : \hat{\mu} > 0\},$ the 99.9th percentile of $\hat{\mu}\,\big]$

$N_{ab} \leftarrow$ rows in cell (a, b); $K_{ab} \leftarrow$ mismatches in cell (a, b); $P_{ab} \leftarrow K_{ab}/N_{ab}$

for each cell with $N_{ab} < \tau$ **do**
  $P_{ab} \leftarrow \big(\sum_b K_{ab}\big)/\big(\sum_b N_{ab}\big)$ ▷ pooled along its confidence row
end for

return the frozen table $(\mathrm{CT}, \mathrm{MT}, P)$ ▷ LLaDA: 792,841 rows, 8×5 cells

D Additional Experiments and Analyses

This appendix collects the experiments and discussions that support Section 5 but are not frontier grids: dynamic block decoding (Section D.1), the curvature analysis and its diagnostic on all four benchmarks (Section D.2), the transfer of both decoders to MMLU, a benchmark outside the mathematics and code their learned components were fitted on (Section D.3), the audit of both decoders' regret estimates on the decoder's own trajectories, which measures the calibration premises of Section A during live decoding (Section D.4), the comparison with the dependency-guided decoders, including our own reproduction of DAPD and its prompt ablation (Section D.5), the comparison with a learned unmasking policy at matched block geometry (Section D.6), the definitions behind the one-number summaries of Tables 1 and 2 (Section D.7), and two shorter notes on cross-paper protocol drift and on a preliminary information-theoretic diagnostic. The complete grids and the tables of every operating point are in Section E; the protocol under which every lane of this appendix was run is stated there (Sections E.1 and E.2).

D.1 Dynamic block decoding unlocks additional parallelism

Fixed-block decoding is a common heuristic for DLM inference, restricting each round to masked positions inside a predefined block. As decoding progresses through a block, however, the number of remaining masked candidates can shrink even when the decoder could safely commit many more tokens. We call this effect *candidate starvation*: parallelism becomes limited by the available candidate pool rather than by the decoder's own regret criterion. We therefore introduce *dynamic block decoding*, which expands the active block as needed to maintain B masked candidates whenever possible. This changes only candidate eligibility; Fast-dLLM, PIG.INT, and PIG.EXT otherwise use their original selection rules.

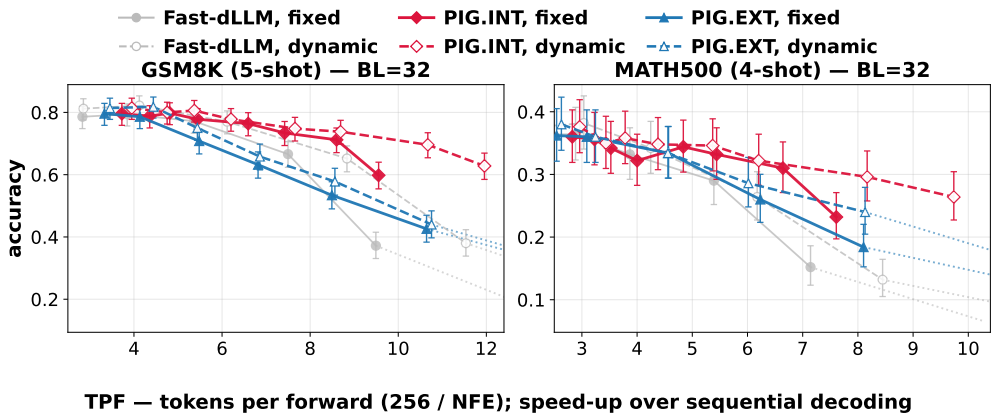

Figure 11: **Dynamic block decoding unlocks additional parallelism.** Accuracy versus TPF on GSM8K and MATH-500 under fixed and dynamic 32-candidate decoding. Filled markers and solid lines denote fixed blocks, while open markers and dashed lines denote dynamic blocks; every decoding rule is evaluated under both candidate geometries.

Dynamic block decoding improves the accuracy–TPF frontier even for Fast-dLLM, confirming that fixed blocks themselves leave useful parallelism on the table. The gains are substantially larger for PIG, whose regret-based selection rules can exploit additional low-regret candidates exposed by the larger block. On GSM8K, PIG.INT reaches 7.6 TPF at 0.75 accuracy (one Wilson interval below Fast-dLLM's slowest point) under dynamic blocks, compared with 7.0 TPF under fixed blocks, and its sweep now extends past 10 TPF, where it holds 0.71 accuracy against 0.54 for dynamic-block Fast-dLLM. On MATH-500, PIG.INT extends from 7.6 to 9.7 TPF and holds 0.28 accuracy at 9 TPF where its fixed-block sweep has already ended, while PIG.EXT gains 4 to 5 points at 7 to 9 TPF (0.27 against 0.23, and 0.22 against 0.17). These results suggest that candidate selection and candidate availability are distinct bottlenecks, leaving further opportunities for DLM parallelization by addressing both jointly.

Table 3 reads the fixed and dynamic 32-token-block sweeps of Figure 11 (LLaDA-8B-Instruct, GSM8K and MATH-500) in the two directions used throughout the paper: the throughput each lane reaches before it gives up one Wilson interval of the slowest fixed-block Fast-dLLM point's accuracy, and the accuracy each lane holds at two fixed throughputs. At matched accuracy the dynamic window helps every lane by a similar factor (1.12–$1.13\times$ for Fast-dLLM, 1.04–$1.08\times$ for PIG.INT, 1.02–$1.14\times$ for PIG.EXT). The difference between the rules appears at the fast end: the dynamic window extends PIG.INT's sweep from 9.6 to 12.0 TPF on GSM8K and from 7.6 to 9.7 on MATH-500, and at 10 and 9 TPF respectively PIG.INT holds 0.71 and 0.28 accuracy, 17 points above its fixed-block extrapolation and 15–16 points above dynamic-block Fast-dLLM, while PIG.EXT gains 3–5 points at every throughput listed. The larger candidate pool therefore matters most to the rules that would otherwise run out of admissible candidates inside a block.

Table 3: **Fixed versus dynamic 32-token blocks on LLaDA-8B-Instruct**, the sweeps of Figure 11. $\mathrm{TPF}^\star$: largest throughput at which the lane stays within one Wilson 95% half-width of the slowest fixed-block Fast-dLLM point's accuracy (a_0; the matched-accuracy rule of Table 1). acc@T: accuracy linearly interpolated at T TPF; § the sweep ends short of T and is extended along its last segment. Best geometry per lane and column in bold.

dataset	lane	$\mathrm{TPF}^\star$		acc @ T_1		acc @ T_2	
		fixed	dynamic	fixed	dynamic	fixed	dynamic
GSM8K (a_0=0.786; T_1=8, T_2=10)	Fast-dLLM	5.80	**6.58**	0.592	**0.688**	0.343	**0.535**
	PIG.INT	6.97	**7.56**	0.723	**0.745**	$0.544^\S$	**0.710**
	PIG.EXT	4.75	**5.44**	0.563	**0.604**	0.458	**0.488**
MATH-500 (a_0=0.364; T_1=7, T_2=9)	Fast-dLLM	4.23	**4.75**	0.163	**0.216**	0.101	**0.122**
	PIG.INT	5.99	**6.20**	0.281	**0.311**	$0.120^\S$	**0.279**
	PIG.EXT	4.84	**4.93**	0.229	**0.265**	0.167	**0.217**

D.2 PIG ADAPTS TO THE INFORMATION CURVATURE OF DECODING

Why can PIG substantially vary its parallel width while maintaining accuracy? One interpretation is that PIG adapts the size of the parallel decode set S to keep the deviation from the sequential 1 NFE per token trajectory smallest. We define this deviation through the decode trajectory curvature of second-order finite differences of entropy,

$$H(S, N(S); \mathbf{x}) := \sum_{i \in S} \sum_{j \in N(S)} H_j(\mathbf{x}) - H_j(\mathbf{x} \oplus x_i) - \sum_{j \in N(S)} H_j(\mathbf{x}) - H_j(\mathbf{x} \oplus x_S) \tag{36}$$

Large curvature indicates that committing the current set S substantially changes the distribution over the vocabulary of the upcoming tokens in $N(S)$. Note that curvature is guaranteed to be zero for any set S of cardinality 1 which corresponds to the sequential oracle. As such, keeping curvature minimal throughout ensures a trajectory that adheres more closely to the sequential 1 NFE per token sequential oracle.

We measure fixed parallel trajectory's curvature and compare it with the size of S selected by PIG.INT, PIG.EXT, and fastDLLM. Figure 12 reports the curvature dynamics together with $|S|$ selected by PIG on all four benchmarks.

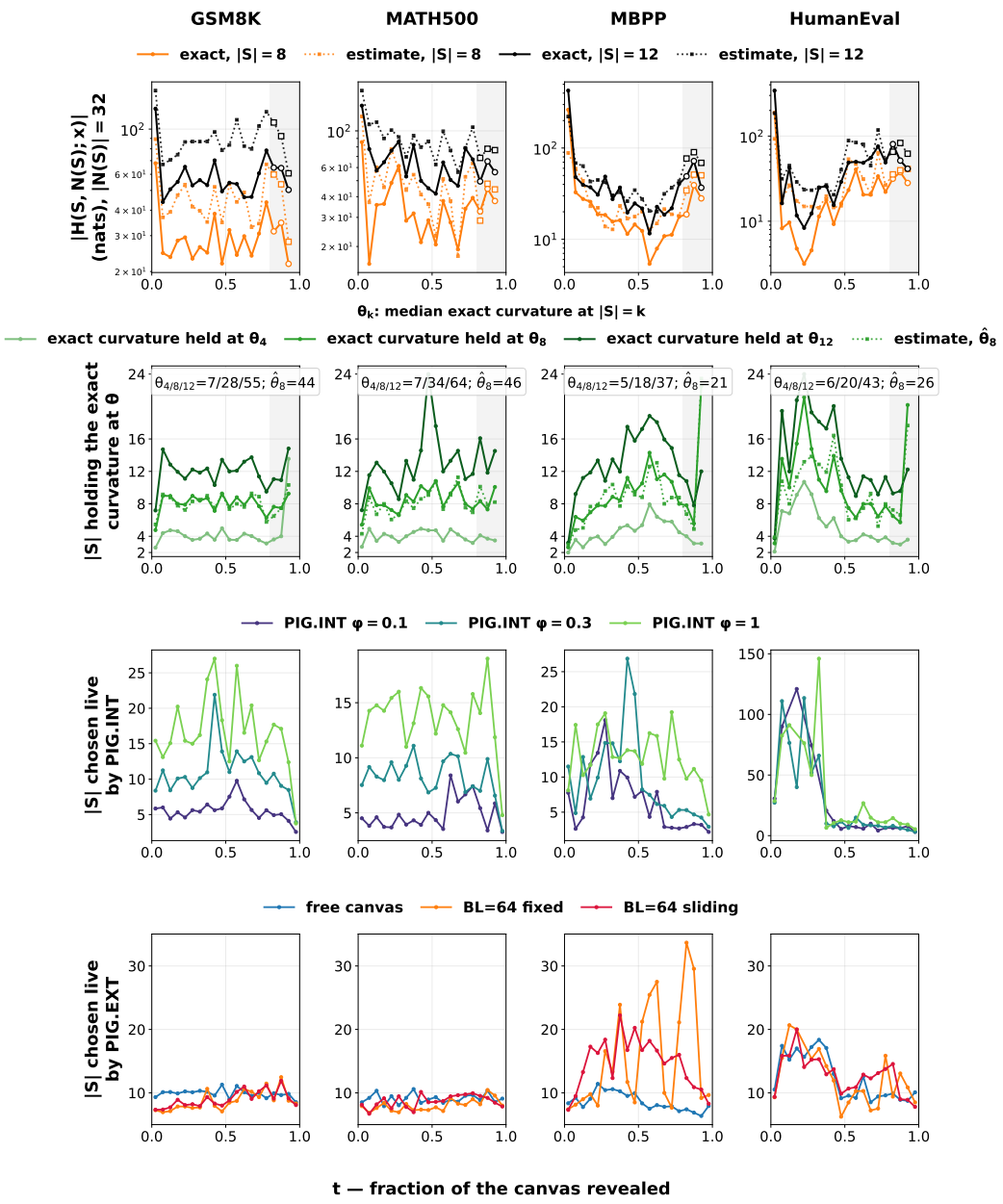

Figure 12: **PIG adapts its decoding width to information curvature, all four benchmarks.** Diagnostic on LLaDA-8B-Instruct across GSM8K, MATH-500, MBPP, and HumanEval. The top row compares exact and estimated changes in the uncertainty of upcoming candidates, while the remaining rows show parallel widths induced by fixed regret budgets and live PIG decoding.

Both PIG variants vary S systematically in line with keeping a constant curvature, taking larger steps in flatter regions and smaller steps as curvature increases. Confidence-based decoding tracks this structure less closely, since token-wise confidence does not directly measure how committing one set changes the predictions that follow. The relationship is especially direct for PIG.EXT, whose cross-round regret explicitly measures how much another model evaluation would change the upcoming candidates.

D.3 DOMAIN TRANSFER: PIG CARRIES OVER TO MMLU WITHOUT REFITTING

Everything PIG learns is fitted once per backbone, on mathematics and code alone. The finite-difference estimator, the adapter of Figure 4, is trained on decoding states harvested from the training splits of GSM8K, MATH, OpenCoder and MBPP (Section B.1), and the calibrated mismatch table on commits replayed from GSM8K-train and OpenCoder (Section B.2). None of these corpora is a multiple-choice benchmark, and none covers a subject outside mathematics and programming. The frontiers of Section 5 therefore leave a question open: whether what these components learn is a property of the backbone's decoding, which should carry over to any input, or a regularity of the math and code they were fitted on. We test this on a benchmark that departs from the training mixture in both format and subject matter.

MMLU (Hendrycks et al., 2021a) poses four-option multiple-choice questions in 57 subjects that span the natural sciences, the humanities, the social sciences and professional fields such as law and medicine. We draw 500 of its 14,042 test questions, stratified by subject in proportion to subject size with a fixed seed, so that every subject appears (4 to 52 questions each). Only 55 of the 500 come from its mathematics and computer-science subjects; the largest single subject is professional law (52 questions), followed by moral scenarios (32). Nemotron-3B-Instruct answers zero-shot under the multiple-choice prompt of the model's released evaluation harness (NeMo-Skills), which asks for a final line of the form `Answer: \boxed{A}`, and the boxed letter is graded against the key. The canvas is 2,048 tokens, decoding runs in fixed blocks of 16, 32 and 64 tokens, and TPF is measured as on every Nemotron panel, as generated tokens before end-of-sequence per forward. Nothing is refitted or retuned: the runs load the same estimator checkpoint and tables as the math and code grids, and sweep subsets of the same knob values, Fast-dLLM at $\nu \in \{0.9, 0.7, 0.5\}$, PIG.INT at $\varphi \in \{0.015, 0.05, 0.1, 0.3, 0.5\}$, and PIG.EXT under the same composite rule, its two slowest points ($\theta = 0.125$ and 0.25) floored at confidence $\nu \geq 0.9$ and $\nu \geq 0.8$ and the rest plain at $\theta \in \{0.5, 1, 1.5\}$.

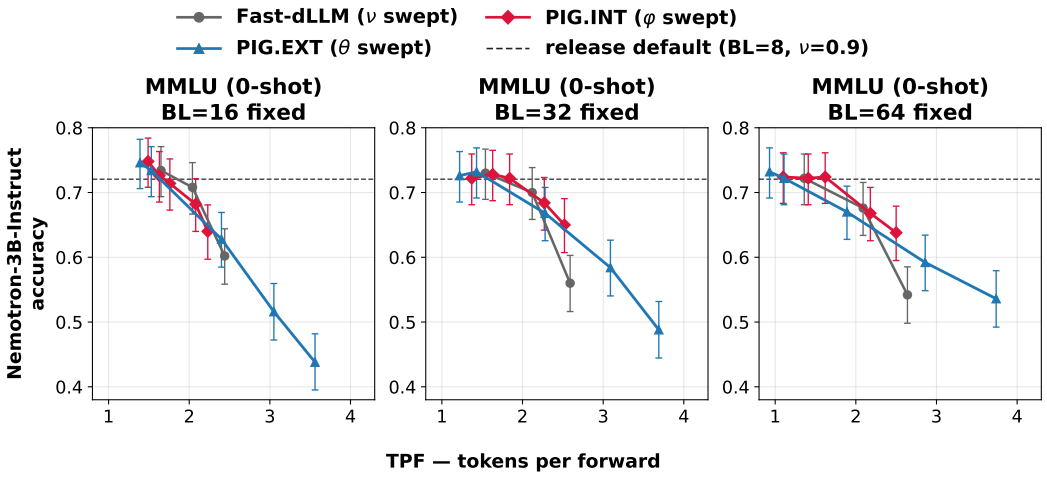

Figure 13: **Domain transfer: Nemotron-3B-Instruct on MMLU**, zero-shot, 500 questions stratified over all 57 subjects, at fixed blocks of 16, 32 and 64 tokens (canvas 2,048). Grey: Fast-dLLM (on Nemotron, the model's own confidence-threshold decoder, the same rule); red: PIG.INT; blue: PIG.EXT, whose two slowest operating points additionally floor admission at a confidence ν. The estimator and tables are the ones fitted on math and code (Section B), used unchanged. Dashed line: the accuracy the model's release reports at its default setting (BL=8, $\nu = 0.9$) (Fu et al., 2026). Error bars are Wilson 95% intervals.

Figure 13 shows the frontier keeping the shape it has on Nemotron's math and code benchmarks. At their most conservative settings the three decoders agree, at 0.72–0.75 accuracy for every block size, around the 0.721 that the model's release reports at its default setting (Fu et al., 2026): the harness is consistent with the release on a benchmark our components never saw, and neither PIG variant gives up accuracy at its slow end. As Fast-dLLM's threshold is lowered to $\nu = 0.5$, its accuracy

falls by 13 to 18 points, to 0.602, 0.560 and 0.542 at BL=16, 32 and 64. At BL=32 and 64, PIG.INT is more accurate than Fast-dLLM at matched TPF (Fast-dLLM's sweep linearly interpolated at the PIG.INT cell's TPF) on every one of its four cells inside Fast-dLLM's range, by 0.3 to 6.8 and 0.3 to 6.2 points respectively; at the fast end, where PIG.INT runs within 0.15 TPF of Fast-dLLM's fastest point, the two lanes' Wilson intervals no longer overlap (0.650 at 2.52 TPF against 0.560 at 2.59 on BL=32, 0.638 at 2.50 against 0.542 at 2.64 on BL=64). PIG.EXT extends the frontier past Fast-dLLM's fastest point: on BL=32 it reaches 0.584 at 3.09 TPF, faster and more accurate than that point, and on BL=64 it holds 0.592 at 2.86 TPF and matches Fast-dLLM's fastest accuracy at 1.4× its throughput (0.536 at 3.74 against 0.542 at 2.64). At BL=16 the lanes stay within each other's intervals, PIG.INT 1.2 to 1.7 points below Fast-dLLM on its three cells at matched TPF and PIG.EXT 1.5 points above at 2.40 TPF, which is also how PIG.INT and Fast-dLLM compare on GSM8K at that block size (Figure 21).

Neither component is specific to a domain by construction, although nothing guaranteed that it would transfer. The estimator's target is defined by the backbone alone, as the change in the model's own entropy at one position when another is committed, and the table indexes an empirical mismatch rate by two features, the committing position's confidence and its predicted coupling, whose meaning does not depend on the subject of the question. Section B.2 found the table's GSM8K-train and OpenCoder fits within 0.03 of each other where the budget is spent; MMLU is a stronger version of that test, a domain neither corpus resembles, read on the decoded frontier rather than on the table. The evidence is one benchmark on one backbone at $n = 500$, which puts the Wilson half-widths near four points, and every lane stays below 4 TPF, a narrower band than on math and code. Within that scope, the three decoders keep the ordering they have on math and code without any component being refitted, which is what one would expect if what PIG learns is how the backbone's predictions depend on one another rather than what mathematics and code look like.

D.4 THE ADAPTER IS A CALIBRATED PREDICTOR OF REGRET

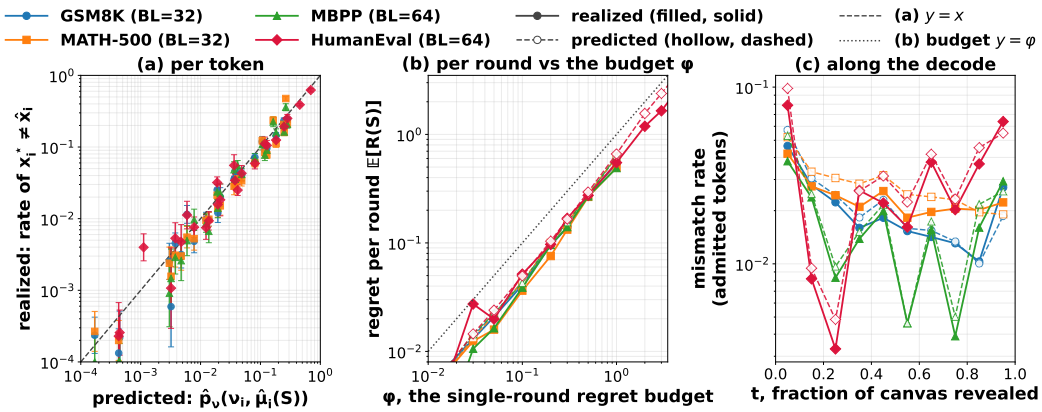

Figure 14: **The regret calibration on the decoder's own trajectories: PIG.INT on LLaDA-8B-Instruct** at the geometry of Figure 5 (GSM8K and MATH-500 at BL=32 fixed, MBPP and HumanEval at BL=64 fixed), every third round audited by replaying the committed set one token at a time from the round's own canvas in walk order (Definition A.2). (a) Realized disagreement rate per table cell against the cell's value, pooled over the φ ladder (Wilson 95%). (b) Per φ: realized regret per audited round $\mathbb{E}[R(S)]$ (filled) and predicted $\mathbb{E}[\hat{R}(S)]$ (hollow) against the budget line $y = \varphi$. (c) Realized (solid) and predicted (dashed) mismatch rate per decile of the decode, token-weighted.

The guarantees of Section A rest on one empirical premise: that the mismatch table is calibrated on the states the decoder actually visits, Definition A.3, from which Proposition A.4 bounds the realized regret of every PIG.INT round by its predicted regret and hence by the budget, $\mathbb{E}[R(S)] \leq \mathbb{E}[\hat{R}(S)] + \bar{\epsilon}\,\mathbb{E}|S| \leq \varphi + \bar{\epsilon}\,\mathbb{E}|S|$ (Equation (27)). The table was fitted offline on replayed probes (Section B.2); whether it stays calibrated during live decoding, on the sets the budget actually admits, is a separate question, and we measured it on the runs of Figure 5 themselves.

What was measured. Every PIG.INT cell of the LLaDA-8B-Instruct frontier at the geometry of Figure 5 (GSM8K and MATH-500 at BL=32 fixed, MBPP and HumanEval at BL=64 fixed; φ from 0.015 to 1, and to 10 on HumanEval; 36 cells) ran with an audit switched on: at every third decoding round (every eighth on the two slowest HumanEval cells), once the walk had chosen its set S, the harness replayed S one token at a time from a copy of the round's own pre-commit canvas, in walk order, which is exactly the re-anchored reference of Definition A.2. For every admitted token after the seed it recorded the confidence ν_i and coupling $\hat{\mu}_i(S_{<i})$ the token was admitted under and the realized disagreement $\mathbf{1}\{x_i^\star \neq \hat{x}_i\}$; the predicted value is the table read at those same two features, the lookup the walk itself performed, so realized and predicted are paired per token. The decode is untouched by the audit: the parallel set is what is committed, and the replay forwards are not counted in NFE, so the throughputs and accuracies of Figure 5 are those of the audited runs. The population is the admitted tokens, $754{,}594$ of them over the 36 cells; calibration conditional on admission is what Proposition A.4 requires, and refused candidates are not replayed here. The seed token of each round is consistent by construction and is excluded from both sides.

What we observe. Figure 14 reads the audit three ways. *Per token* (a): grouping the admitted tokens by the table cell they were priced in, the realized mismatch rate follows the table's value over three decades. Token-weighted over each benchmark's whole ladder, realized over predicted is 0.92 on GSM8K, 0.84 on MATH-500, 0.86 on MBPP and 0.83 on HumanEval; per cell, over the 72 cells holding at least 500 tokens at $\hat{p}_\nu \geq 0.01$, where the budget is spent, the ratio runs from 0.49 to 1.79 with median 0.84. The table errs on the conservative side. *Per round* (b): the realized regret $\mathbb{E}[R(S)]$ and the predicted $\mathbb{E}[\hat{R}(S)]$ track each other across two decades of φ on all four benchmarks, realized over predicted lying between 0.7 and 1.1 on every cell with at least $1{,}000$ audited rounds except the slowest MBPP cell (0.29, six mismatches in $3{,}146$ rounds), and both stay below the budget line: realized regret is 0.13–$0.56\,\varphi$ against a predicted 0.41–$0.66\,\varphi$ on those cells, and the largest realized regret over all 36 cells is $0.91\,\varphi$. The margin has two sources. The walk closes at the first candidate that does not fit, so it typically spends about half of φ, and the table's slight over-prediction adds to it. *Along the decode* (c): realized over predicted stays within 0.7–1.0 for the first nine deciles of the canvas on every benchmark and rises to 1.1–1.5 in the last, where few tokens remain; the saw-tooth of the two BL=64 benchmarks is block structure (the tails of the four 64-token blocks, where the few remaining candidates are near-certain), and the two curves move through it together.

Empirically, then, Definition A.3 holds on the decoder's own trajectories with a small $\bar{\epsilon}$ and, where it does not hold exactly, errs in the safe direction, and the conclusion of Proposition A.4 is met with margin at every budget: φ is an exchange rate the decode does not exceed. The refused candidates are never replayed on these runs, so the statement is conditional on admission, as the proposition requires. It is one backbone, LLaDA-8B-Instruct, at one geometry per benchmark; HumanEval has 164 problems, and its two slowest cells carry about 430 audited rounds against more than $1{,}200$ elsewhere.

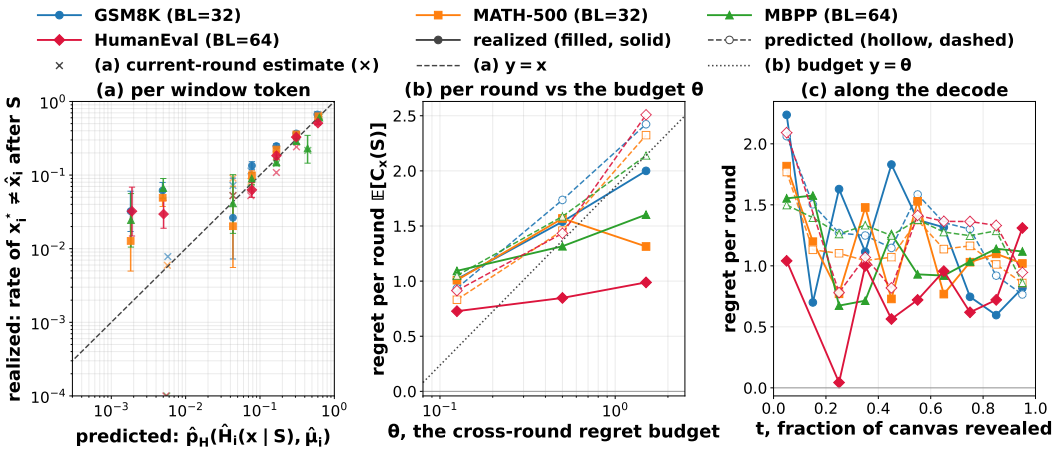

Figure 15: **The cross-round regret calibration on the decoder's own trajectories: PIG.EXT on LLaDA-8B-Instruct** at the geometry of Figure 5 (composite rule, 32 problems per cell, every sixth round audited by replaying S and then $N(S)$ sequentially from the round's own canvas, Definition A.2). (a) Realized deferred disagreement of the window tokens against the deferred estimate $\hat{p}_H(\hat{H}_i(\mathbf{x} \mid S), \hat{\mu}_i)$ (Wilson 95%); crosses: the current-round estimate of the same tokens. (b) Per θ: realized cross-round regret $C_{\mathbf{x}}(S)$ per audited round (filled) against the rule's $\hat{C}_{\mathbf{x}}(S)$ (hollow) and the budget line $y = \theta$. (c) The same per decile of the decode.

PIG.EXT: the deferred estimate and the cross-round regret. PIG.EXT rests on a second premise. Its statistic $\hat{C}_{\mathbf{x}}(S)$ (Equation (9)) evaluates the table twice for every window token, at the current entropy and at the entropy predicted once S is committed, and Definition A.6 asks that the second, deferred evaluation be calibrated as well, with an excess error $\bar{\delta}$; Lemma A.7 then bounds the realized cross-round regret of a round, $C_{\mathbf{x}}(S)$ of Equation (3), by $\theta + |N(S)|(2\bar{\epsilon} + \bar{\delta})$. We measured both on the composite PIG.EXT rule at the geometry of Figure 5, three cells per benchmark, the floored point ($\theta = 0.125$, $\nu \geq 0.9$) and the plain $\theta = 0.5$ and 1.5, on the first 32 problems of each benchmark. At every sixth round, once the rule had chosen S, the harness replayed S and then its window $N(S)$, the 16 most confident remaining positions of the block, one token at a time from the round's own canvas, committing the sequential argmax at each step, which is the reference of Definition A.2 extended to the window, and it read the deferred prediction of every window token from the canvas the decoder actually reached after committing S. Each window token therefore carries two realized disagreements, committed now alongside S and committed one round later, and the two table values the rule priced it with, $\hat{p}_H(H_i, \hat{\mu}_i)$ and $\hat{p}_H(\hat{H}_i(\mathbf{x} \mid S), \hat{\mu}_i)$; per round, the realized $C_{\mathbf{x}}(S)$ is the difference of the two disagreement counts and $\hat{C}_{\mathbf{x}}(S)$ the difference of the two table sums, the number the rule compared with θ. The replays cost $|S| + |N(S)|$ forward passes per audited round, which is why these cells are small, 73 to 469 audited rounds and 1,100 to 5,500 window tokens each. As before, the decode itself is untouched and the replay forwards are not counted in NFE.

Figure 15 reads the audit the same three ways. *Per window token* (a): binned by its predicted value, the deferred estimate is calibrated wherever it is not tiny. For $\hat{p}_H \geq 0.05$ the realized deferred disagreement over the prediction lies between 1.13 and 1.72 on GSM8K, 1.03 and 1.33 on MATH-500, 0.53 and 1.16 on MBPP and 0.81 and 1.10 on HumanEval, close to the diagonal above $\hat{p}_H \approx 0.2$. In the tail it under-predicts: where the table predicts less than 1%, 1 to 7% of the window tokens still change once S is revealed, while the current-round estimate of the same tokens is accurate there (0 to 0.8% realized against 0.02 to 0.6% predicted), so the miss is specific to the deferred term, an absolute $\bar{\delta}$ of a few percent. *Per round* (b): the realized cross-round regret per audited round is 0.73–1.09 at the floored point, 0.85–1.57 at $\theta = 0.5$ and 0.99–2.00 at $\theta = 1.5$, against a predicted 0.83–1.05, 1.43–1.74 and 2.14–2.51: the statistic tracks the realized regret at the slow end (realized over predicted 0.80–1.21) and over-predicts it as θ grows (0.59–1.08 at $\theta = 0.5$, 0.39–0.83 at $\theta = 1.5$), the same conservative direction as $\hat{p}_\nu$ in the PIG.INT audit, larger here. Both quantities exceed θ at the two slow points. That is the rule, not a calibration failure: Algorithm 3 keeps the token that first pushes $\hat{C}_{\mathbf{x}}$ past θ, so at the committed S the statistic sits above the budget on every

round the statistic closed, and the floored rounds are closed by the floor rather than by $\hat{C}_{\mathbf{x}}$ at all. Lemma A.7 allows for this through its slack $|N(S)|(2\bar{\epsilon} + \bar{\delta})$, which at $|N(S)| = 16$ and calibration errors of a few percent is of order one disagreement per round, the level observed; the bound is consistent with the data without being tight. *Along the decode* (c): the realized regret per round falls from 1.0–2.2 in the first decile of the canvas to 0.8–1.3 in the last and the prediction from 1.5–2.1 to 0.8–1.0; the two move together, the realized curve noisy at 32 problems.

Definition A.6 therefore holds on the decoder's own trajectories up to a tail in which the deferred estimate is too optimistic by a few percent, and every other error is in the safe direction: PIG.EXT predicts more benefit from waiting than it realizes, and waits accordingly. The caveats are the cell size, 32 problems with every sixth round audited, and that under the floor the window is the paper's $N(S)$, the rule's own above-floor pool being nearly empty there.

D.5 Comparison with dependency-guided decoders

Provenance of the reported points. No external number in this appendix was produced by us; each is transcribed from a published table or an official repository, and the decoders themselves are due to their respective authors. Fast-dLLM is the confidence-threshold parallel decoder of Wu et al. (2025); EB-Sampler is the entropy-bounded unmasking rule of Ben-Hamu et al. (2025); KLASS is the KL-guided stability sampler of Kim et al. (2025); DAPD-Staged and DAPD-Direct are the attention-graph independent-set decoders of Kim et al. (2026a); the RL unmasking policy is that of Jazbec et al. (2025b). In Figure 16, the BL=64 Fast-dLLM, EB-Sampler and KLASS points are Table 3 of Kim et al. (2026a) (LLaDA-8B-Instruct, four 64-token blocks, accuracy and steps), which their repository README reproduces verbatim (Kim et al., 2026b); the two DAPD points at BL=64 appear only in that README (its "4-block" rows, torch `2.5.1+cu121` according to the README) (Kim et al., 2026b), the paper itself reporting DAPD at a single 256-token block only (its Tables 3 and 5), which we do not draw for the reasons given below. Steps are converted to TPF as $256/\text{steps}$, since their harness performs one forward per step; the Wilson intervals on these points use their problem counts (GSM8K 1,319, MATH-500 500, MBPP 500, HumanEval 164), not ours. In Figure 17, the RL-policy and Fast-dLLM points are Appendix J of Jazbec et al. (2025b) (Tables 2 and 3, the block-32 rows, values the authors state correspond directly to their plotted points), with their reported three-seed spread as the error bar; that their evaluation is 0-shot under the chat template, and that their block-32 decoder unmasks contiguous blocks in order, we read from the official code (Jazbec et al., 2025a) rather than the paper. Our own lanes on every panel are the cells tabulated in Section E.3.

Our reproduction of DAPD. The filled black markers in Figure 16 are DAPD run by us, under our protocol, on every panel. We transcribed the decoder from the released implementation (Kim et al., 2026b): the dependency graph is the head-averaged attention of the last 30% of layers (ten of LLaDA's 32), symmetrised, restricted to the masked positions of the active block and normalised per row by its maximum; candidates are ranked by normalised degree times normalised confidence; a greedy independent set accepts a candidate when its dependency to every accepted token is at most τ, with τ rising linearly in decoded fraction between the endpoints of their runner scripts (Staged 0.01–0.15 on HumanEval and 0.01–0.05 on GSM8K; Direct 0.01–0.05 and 0.005–0.05); Direct first commits every position of confidence 1.0, Staged also commits every position of confidence at least 0.9 once half the canvas is decoded, confidence being read in the model's logits precision as in their code. The transcription was then checked against their code itself: we ran their `generate_dapd` unmodified, on our prompts and weights, over all 164 HumanEval problems at their default single block, and obtained pass@1 0.256 at 27.7 forwards (Staged) and 0.280 at 24.9 (Direct) against 0.262 at 27.4 and 0.287 at 25.5 from our transcription; per problem, the two agree on pass/fail for 86% of problems with a median forward-count difference of two, the residual being the one deliberate numerical difference (their code recomputes attention in single precision, ours reads the model's half-precision maps). The BL=64 cells drawn use our transcription at the same τ schedules with the same n as our lanes, on all four benchmarks.

Under our protocol, DAPD sits on the Fast-dLLM frontier rather than above it: on GSM8K, Staged reaches 0.776 at 2.3 TPF and Direct 0.746 at 4.2 against Fast-dLLM's 0.800–0.810 at 2.9–4.0; on MATH-500, Staged 0.364 at 2.0 and Direct 0.348 at 3.2 against 0.380–0.368 at 2.2–3.0; on MBPP, Staged 0.380 at 2.7 and Direct 0.386 at 5.2, at parity with Fast-dLLM and the two PIG lanes at that

speed; on HumanEval, Staged 0.256 at 3.0 and Direct 0.317 at 6.9 against 0.390 at 5.2 and 0.360 at 7.2. Our forward counts at BL=64 track their README's on every benchmark (MATH-500 126 and 80 against 123 and 85; MBPP 95 and 50 against 97 and 37), while the accuracies move with the protocol: higher than theirs by eight to nine points on MATH-500, equal on MBPP, and lower by about twelve points on HumanEval even though our Fast-dLLM at the same geometry is not; so the offset is specific to DAPD under our prompt. A prompt ablation with their code locates most of it: run on HumanEval at their default single block under lm-eval's raw completion prompt, the prompt their released runners use, their code reproduces the step counts of their Table 3 exactly and DAPD-Direct's accuracy within a point, and DAPD-Staged's to within six points (n=164); under our chat-template prompt the same code at the same τ decodes in 27 forwards instead of 45, so the prompt changes the dependency graph itself, not only the answers. Our observations are therefore that a substantial part of the discrepancy between the reported DAPD numbers and our runs comes down to the prompt protocol, which reshapes the attention graph DAPD selects from; it does not account for all of it, the ablation covers one benchmark and one geometry, and we have not run their code inside their own harness.

DAPD's operating points on the BL=64 plane. Figure 16 places every BL=64 operating point that Kim et al. (2026a) report for LLaDA-8B-Instruct on our BL=64 fixed-block panels, the geometry the two match: Fast-dLLM, EB-Sampler and KLASS at BL=64, and both DAPD variants at BL=64 (DAPD-Staged, the algorithm of their main text, and the acceleration variant DAPD-Direct), the latter two reported only in the project README. Their setting is otherwise close to ours — canvas 256, greedy, lm-eval default shots ($5/4/0/3$), one forward per step so TPF $= 256/$steps — but each method is a single best operating point, no τ_t sweep is reported, and the comparison is not paired: prompts, graders and the GSM8K problem set (their full $1{,}319$, our first 500) differ, their KV-cache use is unstated, and their released lm-eval runners default to *no* chat template for the Instruct model where Fast-dLLM's own harness applies it, so whether their LLaDA numbers are raw few-shot cannot be determined from the repository.

Read at this geometry, DAPD's advantage is confined to the single-block setting the paper headlines. At BL=64 DAPD-Staged needs 93–123 steps (2.1–2.8 TPF), no faster than their own Fast-dLLM; DAPD-Direct is faster (GSM8K 0.758 at 4.2 TPF, HumanEval 0.427 at 4.3, MBPP 0.388 at 7.0) and on the code panels sits at or above our lanes at those speeds, while on the math panels our PIG.INT and PIG.EXT lanes pass through or above it. We do not draw their single-block results: there the baselines are run with end-of-sequence suppression after collapsing without it, a protocol our decoder does not use, and their BL=64 Fast-dLLM on MBPP (0.206 at 6.2 TPF against our 0.354 at 5.9, and 0.404 in their own suppressed single-block run) shows the same truncation at work.

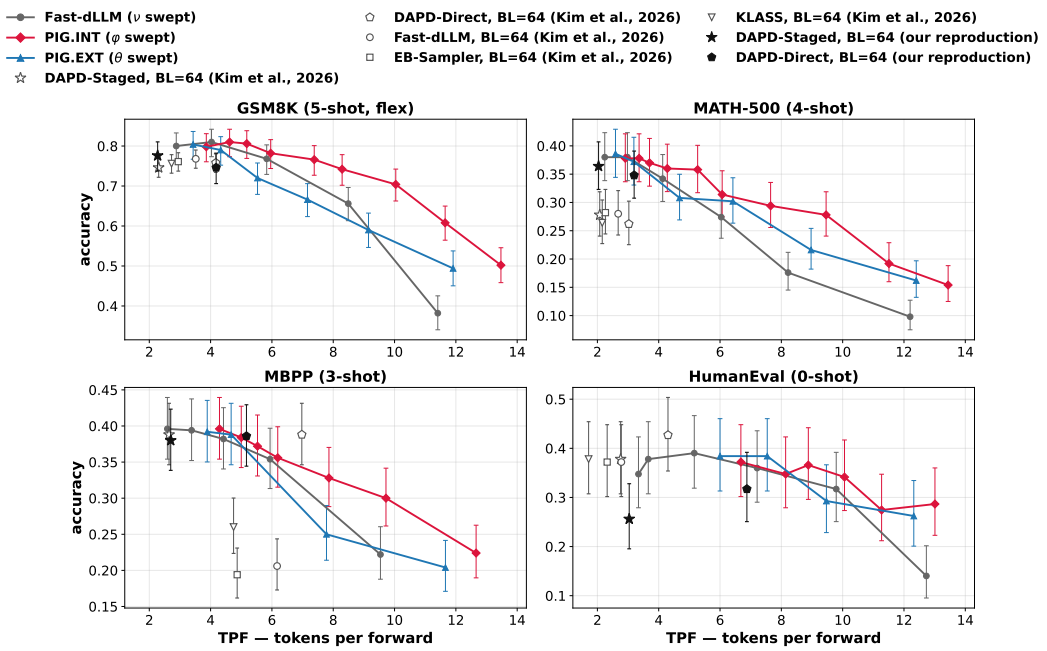

Figure 16: **Our BL=64 frontier against every BL=64 operating point DAPD reports** on LLaDA-8B-Instruct (math above, code below; flexible extraction for GSM8K/MATH-500, pass@1 for code with Wilson 95% bars. Lines: our Fast-dLLM, PIG.INT and PIG.EXT sweeps at BL=64 fixed. Hollow markers, all at BL=64 (four 64-token blocks): Fast-dLLM (Wu et al., 2025), EB-Sampler (Ben-Hamu et al., 2025) and KLASS (Kim et al., 2025) as reported in Table 3 of Kim et al. (2026a); DAPD-Staged (star) and DAPD-Direct (pentagon) from the project README (Kim et al., 2026b), absent from the paper. Filled black star and pentagon: DAPD-Staged and DAPD-Direct run by us under our protocol.

D.6 A LEARNED UNMASKING POLICY AT MATCHED BLOCK GEOMETRY

Figure 17 places the published BL=32 series of Jazbec et al. (2025b) on our BL=32 fixed-block GSM8K and MATH-500 panels: their RL unmasking policy, one policy per reward weight $\alpha \in \{0, 0.3, 1, 3, 10\}$ with their three-seed spread, and Fast-dLLM as run by them (their Appendix J, Tables 2 and 3). Their decoder unmasks contiguous 32-token blocks in order on a 256-token canvas with one forward per step, so block geometry, canvas and NFE accounting match our BL=32 fixed regime exactly; what differs is the prompt (theirs 0-shot under the chat template, ours 5-shot GSM8K and 4-shot MATH-500) and the grader, so the figure is a reference at matched geometry rather than a paired comparison. Points beyond 14 TPF — their fastest policy (α=10, about 25 TPF, GSM8K 0.35 and MATH-500 0.15), their fastest Fast-dLLM threshold, and our θ=3 cell — are not drawn; a dashed segment continues each series toward its hidden neighbour. Read this way, the learned policy sits on our lanes in the 2–4.5 TPF band where it is reported, and the range in which PIG.INT and PIG.EXT extend the frontier (5–14 TPF) is one where the policy is reported only at its α=10 end. Their own Fast-dLLM sweep tracks ours to within about 0.03 over 5–14 TPF on GSM8K, which serves as the protocol-alignment check between the two rigs.

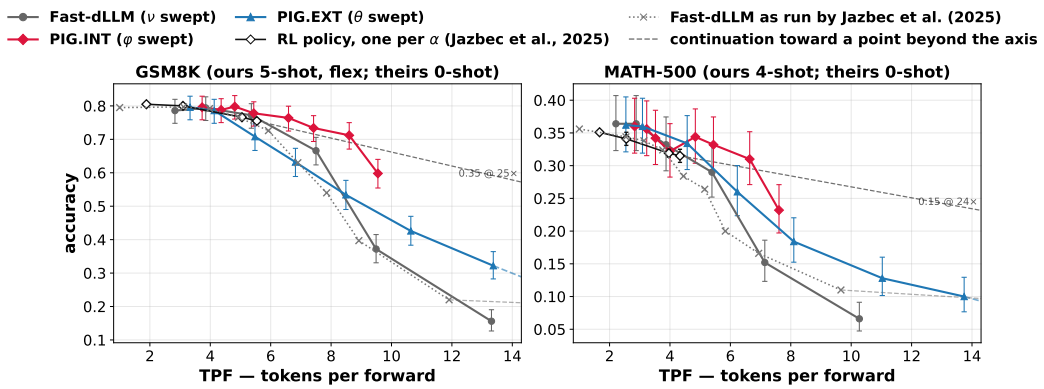

Figure 17: **Our BL=32 frontier against the published BL=32 series of Jazbec et al. (2025b)** on LLaDA-8B-Instruct (GSM8K flexible extraction, MATH-500 with Wilson 95% bars. Lines: our Fast-dLLM, PIG.INT and PIG.EXT sweeps at BL=32 fixed. Black diamonds: their RL policy, one per reward weight α, $\pm$ their three-seed spread; dotted grey: Fast-dLLM as run by them; both at BL=32 fixed, canvas 256, 0-shot. Same block geometry, canvas and NFE accounting.

D.7 One number per comparison: matched-accuracy throughput and accuracy at a fixed throughput

The frontier figures compare sweeps; Tables 1 and 2 in Section 5 reduce each comparison to one number. Table 1 compares both decoders against every baseline of this appendix in the direction they are built for, throughput at matched accuracy. Each baseline supplies one target, its best operating point: an accuracy a_{ref} at a throughput TPF_{ref}. Against it we report the fastest *measured* cell of the lane's sweep whose accuracy is at least a_{ref}, and give two numbers for that cell, its speed-up $\text{TPF}/\text{TPF}_{\text{ref}}$ and its accuracy minus a_{ref} in points; where no cell of a lane reaches the target, the lane's most accurate cell is reported and the difference is negative. Nothing is interpolated and no tolerance is applied, so both numbers name a run we performed. Fast-dLLM supplies the most accurate cell of its own sweep on our rig and DAPD our port of the authors' code at BL=64 (Section D.5), so those three columns are paired; EB-Sampler and KLASS supply only the single cells published by Kim et al. (2026a) at the same four-block geometry, on their rig, so those columns are not paired, and the protocol offset of Section D.5 (our Fast-dLLM exceeds theirs by 4–14 points at equal TPF on three of four benchmarks) accounts for a large part of them. Read on the paired columns, PIG.INT reaches DAPD-Direct's accuracy at 1.5–1.8$\times$ its throughput on three of the four benchmarks and at 0.8$\times$ on MBPP, in every case at equal or higher accuracy.

Table 2 compares the three lanes we sweep on both models in the other direction, accuracy at fixed throughputs: 10 and 15 TPF on LLaDA-8B-Instruct, 3 and 5 TPF on Nemotron-3B-Instruct, every row at BL=64 fixed, each lane's accuracy linearly interpolated between the two measured cells that bracket the target. These are estimates read off the frontier rather than measured operating points, and no lane is extrapolated: where a sweep ends before the target, the table gives that lane's last measured cell, marked with its own throughput, and takes its gain against Fast-dLLM at that same throughput. Four cells are of this kind (PIG.INT on LLaDA MATH-500 and MBPP at 15 TPF; PIG.INT on Nemotron MBPP and PIG.EXT on Nemotron HumanEval at 5 TPF), and on LLaDA MATH-500 at 15 TPF Fast-dLLM's own sweep ends at 12.2 TPF, so no gain is given in that group. On LLaDA, PIG.INT is the most accurate lane at 10 TPF on every benchmark (+4 to +19 points over Fast-dLLM) and PIG.EXT takes over at 15 TPF on GSM8K and MBPP (+14 and +10 points over Fast-dLLM); on MATH-500 it is the only lane still measured at 15 TPF. On Nemotron the three lanes are within a few points on GSM8K at both throughputs (Fast-dLLM ahead at 3 TPF), while on code PIG.EXT is the lane that stays accurate: +9/+7 points over Fast-dLLM at 3 TPF, 36.9 against 9.1 points on MBPP at 5 TPF, and 44.5 against 19.5 on HumanEval at 4.9 TPF, the last cell of its sweep. Nemotron MATH-500 has no BL=64 sweep and is not listed.

E COMPLETE FRONTIER GRIDS AND TABLES

This appendix is the complete record of the frontier experiments: the protocol under which every lane was run, the full frontier grids of both models, and the tables of every operating point behind them, including the cells the figures crop. The analyses that read these results are in Section D. Contents:

- **LLaDA-8B-Instruct** (Section E.1): benchmarks, prompts, graders and decode regimes; Figure 18, the five-regime $\times$ four-benchmark grid (flexible extraction); Figure 19, the same grid under strict extraction.
- **Nemotron-Labs-Diffusion-3B** (Section E.2): benchmarks, prompts, graders and decode regimes; Figure 20, the three-geometry grid at BL=32; Figures 21 and 22, the same grid at BL=16 and BL=64.
- **Tables** (Section E.3), one per model and benchmark, every operating point with its Wilson 95% interval: LLaDA-8B-Instruct GSM8K (Table 4), MATH-500 (Table 5), MBPP (Table 6), HumanEval (Table 7); Nemotron-Labs-Diffusion-3B GSM8K (Table 8), MATH-500 (Table 9), MBPP (Table 10), HumanEval (Table 11).

E.1 LLADA-8B-INSTRUCT

Benchmarks, prompts and graders. We evaluate LLaDA-8B-Instruct (Nie et al., 2025) on GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021b), MBPP (Austin et al., 2021b) and HumanEval (Chen et al., 2021) under the evaluation protocol published with Fast-dLLM (Wu et al., 2025), which is the only decoding paper in this space that states every knob: lm-eval task formats and graders, the shot counts $5/4/3/0$, a 256-token canvas, greedy commits, no KV cache. Prompts follow the lm-eval task formats. GSM8K is five-shot in the `Question:`/`Answer:` format with the `####` answer line, the five exemplars fixed once from the GSM8K training split. MATH-500 uses lm-eval's Minerva template and its four fixed exemplars. MBPP is three-shot with the task's own `[BEGIN]`/`[DONE]` exemplar format, and HumanEval is zero-shot function completion. Every prompt is wrapped in LLaDA-Instruct's own chat template, as the model's release and Fast-dLLM's harness do for the Instruct model; this is our one stated deviation from stock lm-eval, whose LLaDA runs are raw few-shot completions, and it matters (on a BL=64 probe the chat template gave 0.72 flexible-extraction accuracy on GSM8K against 0.31 for the raw prompt). Graders are lm-eval's: for GSM8K both the strict extractor (the `####` line) and the flexible one (last number in the generation), the latter being the main-text metric because the instruct model omits the `####` line in roughly a third of its answers; for MATH-500 the last `\boxed{}` answer under Minerva normalisation and sympy equivalence; for HumanEval execution-based pass@1 with the standard stop sequences; for MBPP execution of the task's three tests in a sandbox. Problem counts are the first 500 of GSM8K's $1{,}319$ test problems, all 500 of MATH-500, the 500 problems of the classic MBPP test split, and all 164 of HumanEval; Wilson 95% intervals are drawn on every accuracy. TPF is 256 divided by the mean number of forwards per problem.

Decode regimes. Five candidate geometries are evaluated on every benchmark, each with its own baseline sweep: fixed 32- and 64-token blocks, the semi-autoregressive block decoding of the LLaDA release; the dynamic 32- and 64-candidate windows of Section D.1, which advance as commits land; and the free canvas, where all 256 positions are eligible. On every geometry the PIG.EXT lane is the operating rule stated in Section 5: two confidence-floored cells at the slow end (θ=0.125 with $\nu\geq$0.9, θ=0.25 with $\nu\geq$0.8), then the plain θ ladder from 0.5; the tables mark the floored rows. The main text draws the fixed 32-token block and the free canvas (Figure 5) and the two 32-wide geometries (Figure 11); the full grid is Figure 18.

Baseline. The baseline on every panel is Fast-dLLM's confidence-threshold rule run by us: at each round, every eligible masked position whose top-1 probability exceeds the threshold is committed, at least one always, with the same eligible positions, prompts, canvas and grader as PIG.INT and PIG.EXT, and with the threshold swept from 0.95 down to 0.1 to trace a frontier rather than the single default of 0.9. No KV cache is used on any lane, so TPF counts forwards over the full canvas for every method alike. Every number taken from another paper, and our own reproduction of DAPD, is documented in Section D.5.

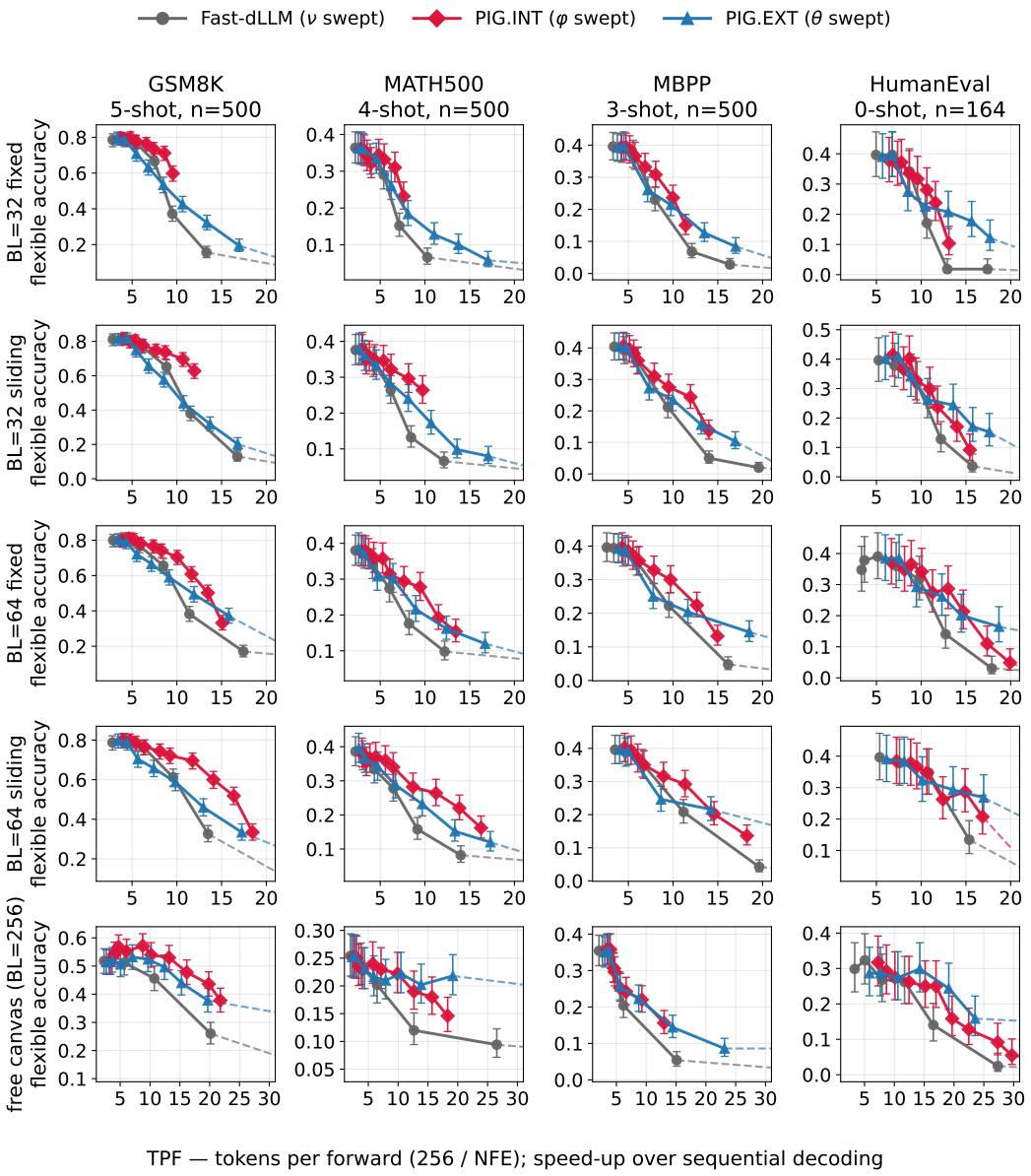

Figure 18: **The full LLaDA-8B-Instruct frontier** (canvas 256, greedy, no KV cache; flexible extraction for GSM8K/MATH-500, pass@1 for MBPP/HumanEval; bars: Wilson 95%). Rows: decode regime; columns: benchmark (shots and n in the title). Grey: Fast-dLLM confidence threshold, swept $0.95 \rightarrow 0.1$; red: PIG.INT (φ); blue: PIG.EXT, with its two slowest operating points additionally flooring admission at a confidence ν equal to 0.9 and 0.8 respectively.

Frontier grids. Figure 18 is the complete LLaDA-8B-Instruct frontier: five decode regimes (rows) $\times$ four benchmarks (columns), all three lanes at full range, flexible extraction. Dashed segments continue a lane toward its next operating point beyond a panel edge. The main-text figures are subsets of this grid; every operating point, including those beyond the crops, is tabulated in Section E.3.

Strict extraction. Figure 19 repeats Figure 18 under the benchmarks' own answer formats (`####` for GSM8K, last `\boxed{}` for MATH-500; code panels are unchanged). Strict GSM8K accuracy is deflated for every lane alike by format non-compliance of the instruct model (roughly a third of generations omit `####`), which is why flexible extraction is the main-text metric; the ordering of the lanes is the same under both.

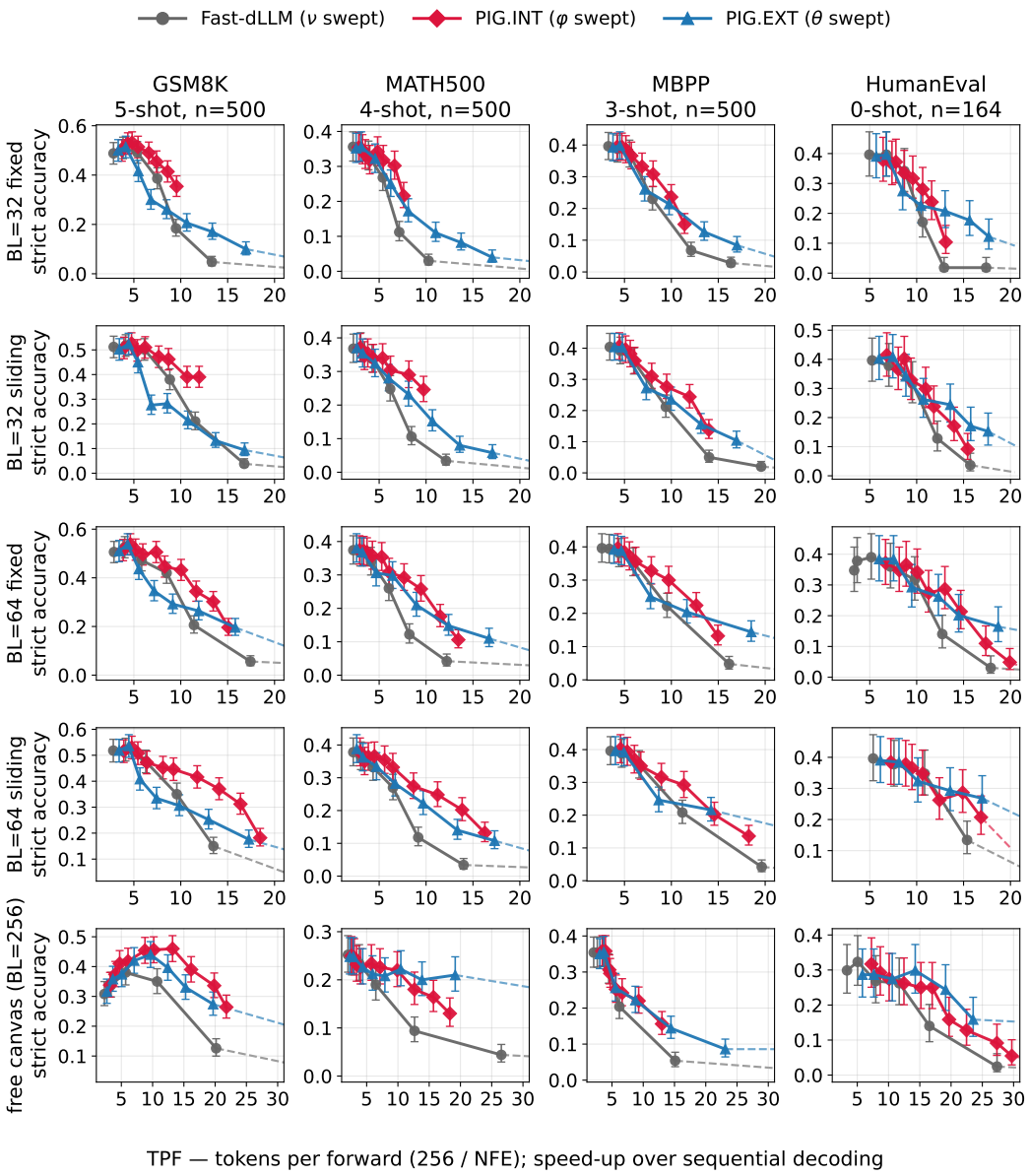

Figure 19: Figure 18 under strict extraction (GSM8K `####`, MATH-500 last `\boxed{}`); MBPP and HumanEval panels are identical to the main text.

E.2 NEMOTRON-LABS-DIFFUSION-3B

Benchmarks, prompts and graders. We evaluate Nemotron-Labs-Diffusion-3B (Fu et al., 2026) on the same four benchmarks, zero-shot throughout, following the model's own evaluation practice: its released harness (NeMo-Skills) prompts the instruct model without exemplars, and few-shot prompting makes it answer the exemplars. Each task body is the benchmark's own text with a minimal instruction (the GSM8K question as is; the MATH-500 problem followed by "Please reason step by step"; the MBPP task description with its three tests; "Complete the following Python function" before the HumanEval prompt), wrapped in the model's chat template. The canvas is 2,048 tokens, the model's end-of-sequence handling stopping generation early, and the graders are the LLaDA ones above, imported unchanged: flexible extraction for GSM8K (strict extraction is void at zero-shot, the model answering in `\boxed{}` rather than with a `####` line), last boxed answer under Minerva normalisation for MATH-500, execution for MBPP and HumanEval. Problem counts are

the first 500 of GSM8K and of MATH-500, 499 of MBPP, and all 164 of HumanEval. TPF is the mean number of generated tokens per forward; because the two models' canvases differ, TPF is not comparable across models.

Decode regimes and baseline. Nemotron decodes in 32-token blocks with a confidence threshold as its native decoder, and has no free-canvas mode, so three geometries are evaluated at each block size: the fixed block, a sliding window of the same width, and a commit band of that width nested in a window twice as wide. The main text uses the model's own BL=32; Section E.2 repeats the grid at BL=16 and BL=64, and the HumanEval panel of Figure 1 uses the fixed 64-token block. The baseline on every panel is the model's own confidence-threshold decoder under the same geometry, swept from 0.9 (its default) down to 0.3: on the fixed 32-token block it is the release's `generate` called as shipped, and on every other geometry the same rule re-implemented over the window, with the release's end-of-sequence guard applied to every lane alike. No numbers from other papers appear on the Nemotron panels; the dashed line of Figure 1 is the accuracy the model's release reports at its default setting.

Frontier grids. Figure 20 is the complete Nemotron-Diffusion-3B frontier at the model's own block size: three geometries at BL=32 $\times$ four benchmarks. The main-text figure (Figure 5, bottom row) is a subset; every operating point, including those beyond the crops, is tabulated in Section E.3. Figures 21 and 22 repeat the grid at BL=16 and BL=64, the three geometries scaled with the block (fixed block, sliding window, and the block nested in a window twice as wide); MATH-500 was not run at BL=64, and at BL=16 its PIG.EXT lane has no confidence-floored cells. Here the baseline is the model's threshold rule re-implemented over the window at every geometry, with the same end-of-sequence guard as our lanes, and PIG.EXT is the composite operating rule of every other figure. The ordering of the lanes is the same at all three block sizes.

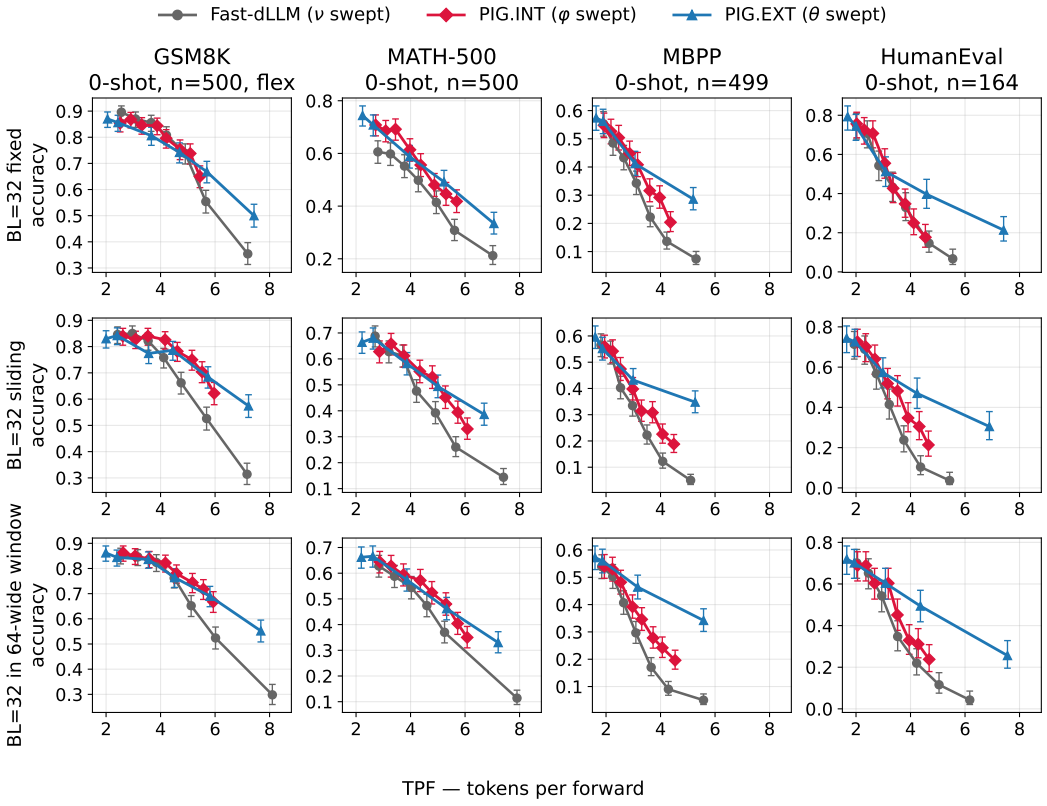

Figure 20: **The full Nemotron-Diffusion-3B frontier** (0-shot, canvas 2048, greedy; flexible extraction for GSM8K, pass@1 for code; bars: Wilson 95%). Rows: the model's own fixed 32-token block, a sliding 32-token window, and a 32-token commit band inside a 64-wide window; columns: benchmark. Grey: Fast-dLLM, the model's own confidence-threshold decoder swept $0.9 \rightarrow 0.3$ under the same geometry; red: PIG.INT; blue: PIG.EXT, the composite operating rule of every figure. Cells beyond 8.5 TPF are omitted; TPF is not comparable across the two models' canvases.

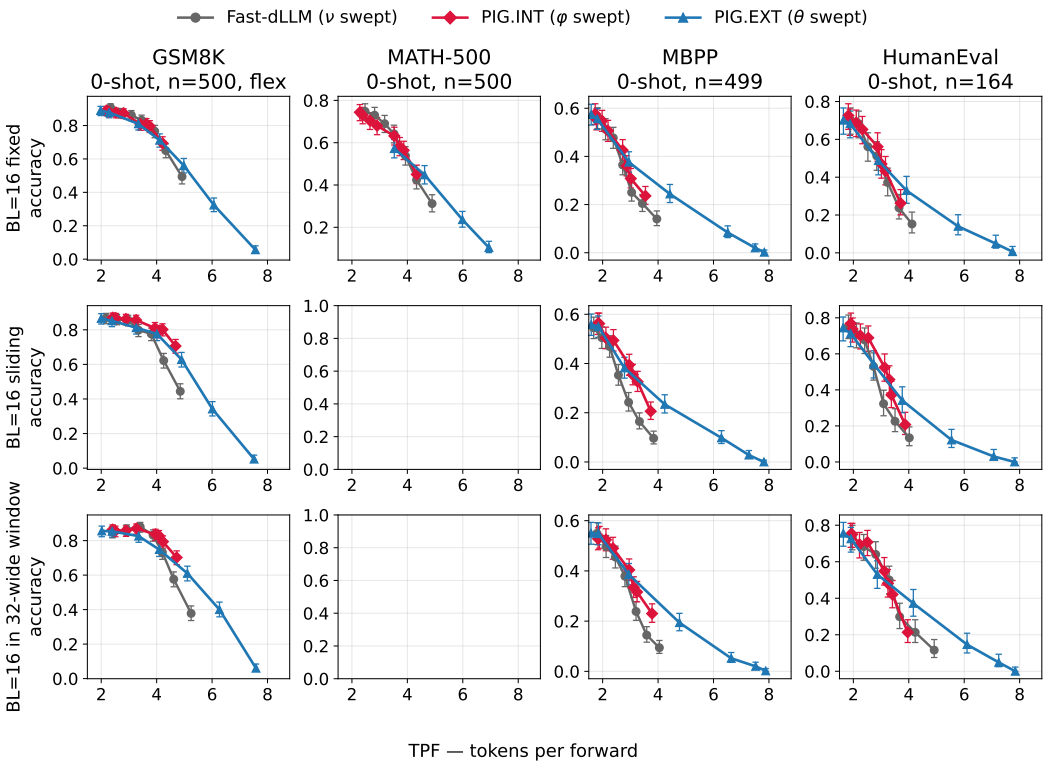

Figure 21: **The Nemotron-Diffusion-3B frontier at BL=16** (0-shot, canvas 2048, greedy; flexible extraction for GSM8K, pass@1 for code; bars: Wilson 95%). Rows: fixed 16-token block, sliding 16-token window, and a 16-token commit band inside a 32-wide window; columns: benchmark. Grey: the model's confidence-threshold decoder under the same geometry, swept $0.9 \rightarrow 0.3$; red: PIG.INT; blue: PIG.EXT, the composite operating rule of Figure 5. Cells beyond 8.5 TPF are omitted.

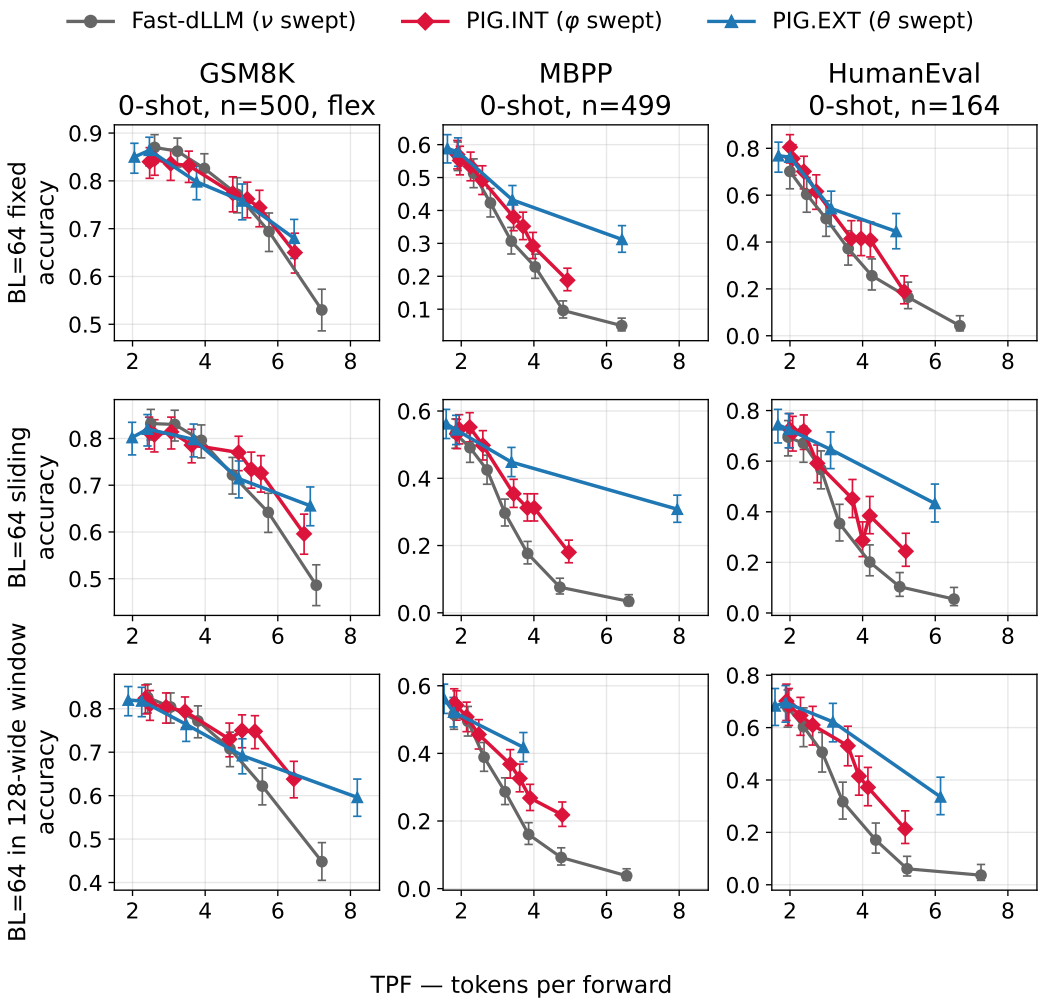

Figure 22: **The Nemotron-Diffusion-3B frontier at BL=64**, as Figure 21 with a fixed 64-token block, a sliding 64-token window, and a 64-token commit band inside a 128-wide window; MATH-500 was not run at this block size.

E.3 FRONTIER TABLES

The eight tables that follow list every operating point behind the frontier figures of this appendix, each accuracy with its Wilson 95% interval, including the cells the figures crop. Tables 4 to 7 are the LLaDA-8B-Instruct grids of Figure 18 and Figure 19 (flexible and strict extraction); Tables 8 to 11 are the Nemotron-Labs-Diffusion-3B grids of Figure 20. In every table the Fast-dLLM lane is the confidence-threshold decoder swept over ν, and PIG.EXT is the composite operating rule of every figure.

Table 4: **Frontier table, GSM8K, 5-shot.** LLaDA-8B-Instruct, canvas 256, greedy, no KV cache; problems $[0, 500)$ (HumanEval all 164). Accuracy with its Wilson 95% interval; flexible and strict extraction. Every operating point behind Figure 18 and Figure 19, including cells beyond the figures' crops; PIG.EXT is the composite operating rule of every figure: its two slowest rows additionally floor admission at a confidence ν (written θ ($\nu{\geq}c$)), then the plain θ ladder from 0.5.

regime	Fast-dLLM				PIG.INT				PIG.EXT			
	ν	TPF	acc (flex)	acc (strict)	φ	TPF	acc (flex)	acc (strict)	θ	TPF	acc (flex)	acc (strict)
BL=32 fixed	0.95	2.83	$0.786^{+0.034}_{-0.038}$	$0.488^{+0.044}_{-0.044}$	0.015	3.73	$0.796^{+0.033}_{-0.038}$	$0.500^{+0.044}_{-0.044}$	0.125 ($\nu{\geq}0.9$)	3.33	$0.796^{+0.033}_{-0.038}$	$0.500^{+0.044}_{-0.044}$
	0.85	3.85	$0.794^{+0.033}_{-0.038}$	$0.510^{+0.044}_{-0.044}$	0.03	4.36	$0.788^{+0.034}_{-0.038}$	$0.528^{+0.043}_{-0.044}$	0.25 ($\nu{\geq}0.8$)	4.13	$0.786^{+0.034}_{-0.038}$	$0.514^{+0.044}_{-0.044}$
	0.7	5.36	$0.772^{+0.035}_{-0.039}$	$0.492^{+0.044}_{-0.044}$	0.05	4.81	$0.798^{+0.033}_{-0.037}$	$0.532^{+0.043}_{-0.044}$	0.5	5.48	$0.708^{+0.038}_{-0.041}$	$0.416^{+0.044}_{-0.042}$
	0.5	7.50	$0.666^{+0.040}_{-0.042}$	$0.386^{+0.045}_{-0.042}$	0.1	5.44	$0.778^{+0.034}_{-0.038}$	$0.514^{+0.044}_{-0.044}$	1	6.82	$0.632^{+0.041}_{-0.043}$	$0.300^{+0.042}_{-0.039}$
	0.3	9.49	$0.372^{+0.043}_{-0.041}$	$0.184^{+0.036}_{-0.032}$	0.2	6.59	$0.764^{+0.035}_{-0.039}$	$0.490^{+0.044}_{-0.044}$	1.5	8.49	$0.534^{+0.043}_{-0.044}$	$0.260^{+0.040}_{-0.037}$
	0.2	13.31	$0.156^{+0.034}_{-0.029}$	$0.048^{+0.022}_{-0.016}$	0.3	7.42	$0.734^{+0.037}_{-0.040}$	$0.454^{+0.044}_{-0.043}$	2	10.64	$0.426^{+0.044}_{-0.043}$	$0.206^{+0.038}_{-0.033}$
					0.5	8.60	$0.712^{+0.038}_{-0.041}$	$0.414^{+0.044}_{-0.042}$	2.5	13.37	$0.322^{+0.042}_{-0.039}$	$0.170^{+0.035}_{-0.030}$
					1	9.55	$0.598^{+0.042}_{-0.044}$	$0.354^{+0.043}_{-0.041}$	3	16.95	$0.194^{+0.037}_{-0.032}$	$0.100^{+0.029}_{-0.023}$
BL=64 fixed	0.95	2.88	$0.800^{+0.033}_{-0.037}$	$0.506^{+0.044}_{-0.044}$	0.015	3.86	$0.798^{+0.033}_{-0.037}$	$0.514^{+0.044}_{-0.044}$	0.125 ($\nu{\geq}0.9$)	3.43	$0.804^{+0.032}_{-0.037}$	$0.510^{+0.044}_{-0.044}$
	0.85	4.03	$0.810^{+0.032}_{-0.037}$	$0.526^{+0.043}_{-0.044}$	0.03	4.62	$0.810^{+0.032}_{-0.037}$	$0.538^{+0.043}_{-0.044}$	0.25 ($\nu{\geq}0.8$)	4.33	$0.790^{+0.033}_{-0.038}$	$0.538^{+0.043}_{-0.044}$
	0.7	5.83	$0.768^{+0.035}_{-0.039}$	$0.472^{+0.044}_{-0.043}$	0.05	5.18	$0.806^{+0.032}_{-0.037}$	$0.516^{+0.044}_{-0.044}$	0.5	5.53	$0.720^{+0.038}_{-0.041}$	$0.436^{+0.044}_{-0.043}$
	0.5	8.49	$0.656^{+0.040}_{-0.043}$	$0.420^{+0.044}_{-0.042}$	0.1	5.96	$0.782^{+0.034}_{-0.038}$	$0.496^{+0.044}_{-0.044}$	1	7.16	$0.666^{+0.040}_{-0.042}$	$0.346^{+0.043}_{-0.040}$
	0.3	11.41	$0.382^{+0.043}_{-0.042}$	$0.206^{+0.038}_{-0.033}$	0.2	7.38	$0.766^{+0.035}_{-0.039}$	$0.506^{+0.044}_{-0.044}$	1.5	9.15	$0.590^{+0.042}_{-0.044}$	$0.292^{+0.041}_{-0.038}$
	0.2	17.43	$0.170^{+0.035}_{-0.030}$	$0.056^{+0.024}_{-0.017}$	0.3	8.29	$0.742^{+0.036}_{-0.040}$	$0.446^{+0.044}_{-0.043}$	2	11.91	$0.494^{+0.044}_{-0.044}$	$0.264^{+0.040}_{-0.037}$
					0.5	10.04	$0.704^{+0.038}_{-0.041}$	$0.432^{+0.044}_{-0.043}$	2.5	15.81	$0.372^{+0.043}_{-0.041}$	$0.196^{+0.037}_{-0.032}$
					1	11.65	$0.608^{+0.042}_{-0.043}$	$0.344^{+0.043}_{-0.040}$				
					2	13.47	$0.502^{+0.044}_{-0.044}$	$0.302^{+0.042}_{-0.039}$				
					3	15.08	$0.332^{+0.042}_{-0.040}$	$0.196^{+0.037}_{-0.032}$				
BL=32 sliding	0.95	2.85	$0.812^{+0.032}_{-0.037}$	$0.512^{+0.044}_{-0.044}$	0.015	3.95	$0.814^{+0.032}_{-0.036}$	$0.508^{+0.044}_{-0.044}$	0.125 ($\nu{\geq}0.9$)	3.45	$0.814^{+0.032}_{-0.036}$	$0.502^{+0.044}_{-0.044}$
	0.85	4.12	$0.822^{+0.031}_{-0.036}$	$0.518^{+0.043}_{-0.044}$	0.03	4.75	$0.800^{+0.033}_{-0.037}$	$0.526^{+0.043}_{-0.044}$	0.25 ($\nu{\geq}0.8$)	4.44	$0.818^{+0.031}_{-0.036}$	$0.524^{+0.043}_{-0.044}$
	0.7	6.12	$0.770^{+0.035}_{-0.039}$	$0.502^{+0.044}_{-0.044}$	0.05	5.37	$0.806^{+0.032}_{-0.037}$	$0.498^{+0.044}_{-0.044}$	0.5	5.44	$0.750^{+0.036}_{-0.040}$	$0.450^{+0.044}_{-0.043}$
	0.5	8.84	$0.652^{+0.040}_{-0.043}$	$0.380^{+0.043}_{-0.041}$	0.1	6.20	$0.778^{+0.034}_{-0.038}$	$0.510^{+0.044}_{-0.044}$	1	6.86	$0.658^{+0.040}_{-0.043}$	$0.276^{+0.041}_{-0.037}$
	0.3	11.54	$0.380^{+0.043}_{-0.041}$	$0.210^{+0.038}_{-0.033}$	0.2	7.66	$0.748^{+0.036}_{-0.040}$	$0.472^{+0.044}_{-0.043}$	1.5	8.56	$0.578^{+0.043}_{-0.044}$	$0.282^{+0.041}_{-0.038}$
	0.2	16.77	$0.130^{+0.032}_{-0.027}$	$0.038^{+0.021}_{-0.014}$	0.3	8.69	$0.738^{+0.037}_{-0.040}$	$0.462^{+0.044}_{-0.043}$	2	10.76	$0.440^{+0.044}_{-0.043}$	$0.214^{+0.038}_{-0.034}$
					0.5	10.67	$0.696^{+0.039}_{-0.042}$	$0.390^{+0.043}_{-0.042}$	2.5	13.72	$0.318^{+0.042}_{-0.039}$	$0.132^{+0.032}_{-0.027}$
					1	11.96	$0.628^{+0.041}_{-0.043}$	$0.390^{+0.043}_{-0.042}$	3	16.81	$0.202^{+0.037}_{-0.033}$	$0.094^{+0.029}_{-0.023}$
BL=64 sliding	0.95	2.81	$0.788^{+0.034}_{-0.038}$	$0.518^{+0.043}_{-0.044}$	0.015	3.93	$0.798^{+0.033}_{-0.037}$	$0.516^{+0.044}_{-0.044}$	0.125 ($\nu{\geq}0.9$)	3.43	$0.798^{+0.033}_{-0.037}$	$0.518^{+0.043}_{-0.044}$
	0.85	4.12	$0.802^{+0.033}_{-0.037}$	$0.522^{+0.044}_{-0.044}$	0.03	4.77	$0.798^{+0.033}_{-0.037}$	$0.530^{+0.044}_{-0.044}$	0.25 ($\nu{\geq}0.8$)	4.46	$0.788^{+0.034}_{-0.038}$	$0.536^{+0.043}_{-0.044}$
	0.7	6.27	$0.764^{+0.035}_{-0.039}$	$0.476^{+0.044}_{-0.043}$	0.05	5.43	$0.786^{+0.034}_{-0.038}$	$0.508^{+0.044}_{-0.044}$	0.5	5.64	$0.704^{+0.038}_{-0.041}$	$0.408^{+0.044}_{-0.042}$
	0.5	9.58	$0.612^{+0.042}_{-0.043}$	$0.350^{+0.043}_{-0.041}$	0.1	6.39	$0.766^{+0.035}_{-0.039}$	$0.472^{+0.044}_{-0.043}$	1	7.42	$0.658^{+0.040}_{-0.043}$	$0.334^{+0.042}_{-0.040}$
	0.3	13.48	$0.326^{+0.042}_{-0.040}$	$0.150^{+0.034}_{-0.029}$	0.2	8.10	$0.744^{+0.036}_{-0.040}$	$0.452^{+0.044}_{-0.043}$	1.5	9.85	$0.588^{+0.042}_{-0.044}$	$0.306^{+0.042}_{-0.039}$
					0.3	9.23	$0.722^{+0.037}_{-0.041}$	$0.446^{+0.044}_{-0.043}$	2	12.97	$0.460^{+0.044}_{-0.043}$	$0.252^{+0.040}_{-0.036}$
					0.5	11.75	$0.696^{+0.039}_{-0.042}$	$0.416^{+0.044}_{-0.042}$	2.5	17.26	$0.334^{+0.042}_{-0.040}$	$0.176^{+0.036}_{-0.031}$
					1	14.09	$0.600^{+0.042}_{-0.044}$	$0.370^{+0.043}_{-0.041}$				
					2	16.37	$0.518^{+0.043}_{-0.044}$	$0.312^{+0.042}_{-0.039}$				
					3	18.47	$0.334^{+0.042}_{-0.040}$	$0.182^{+0.036}_{-0.031}$				
free canvas	0.95	2.29	$0.518^{+0.043}_{-0.044}$	$0.308^{+0.042}_{-0.039}$	0.015	3.19	$0.516^{+0.044}_{-0.044}$	$0.338^{+0.043}_{-0.040}$	0.125 ($\nu{\geq}0.9$)	2.68	$0.514^{+0.044}_{-0.044}$	$0.316^{+0.042}_{-0.039}$
	0.85	3.36	$0.514^{+0.044}_{-0.044}$	$0.338^{+0.043}_{-0.040}$	0.03	4.04	$0.542^{+0.043}_{-0.044}$	$0.374^{+0.043}_{-0.041}$	0.25 ($\nu{\geq}0.8$)	3.65	$0.522^{+0.043}_{-0.044}$	$0.358^{+0.043}_{-0.041}$
	0.7	5.69	$0.512^{+0.044}_{-0.044}$	$0.380^{+0.043}_{-0.041}$	0.05	4.73	$0.568^{+0.043}_{-0.044}$	$0.410^{+0.044}_{-0.042}$	0.5	5.06	$0.506^{+0.044}_{-0.044}$	$0.374^{+0.043}_{-0.041}$
	0.5	10.72	$0.456^{+0.044}_{-0.043}$	$0.350^{+0.043}_{-0.041}$	0.1	6.07	$0.552^{+0.043}_{-0.044}$	$0.418^{+0.044}_{-0.042}$	1	7.05	$0.532^{+0.043}_{-0.044}$	$0.420^{+0.044}_{-0.042}$
	0.3	20.16	$0.260^{+0.040}_{-0.037}$	$0.126^{+0.032}_{-0.026}$	0.2	8.78	$0.572^{+0.043}_{-0.044}$	$0.454^{+0.044}_{-0.043}$	1.5	9.67	$0.524^{+0.043}_{-0.044}$	$0.440^{+0.044}_{-0.043}$
					0.3	10.18	$0.540^{+0.043}_{-0.044}$	$0.456^{+0.044}_{-0.043}$	2	12.44	$0.496^{+0.044}_{-0.044}$	$0.396^{+0.044}_{-0.042}$
					0.5	13.19	$0.530^{+0.043}_{-0.044}$	$0.460^{+0.044}_{-0.043}$	2.5	15.27	$0.440^{+0.044}_{-0.043}$	$0.330^{+0.042}_{-0.040}$
					1	16.16	$0.478^{+0.044}_{-0.043}$	$0.390^{+0.043}_{-0.042}$	3	19.68	$0.378^{+0.043}_{-0.041}$	$0.274^{+0.041}_{-0.037}$
					2	19.85	$0.436^{+0.044}_{-0.043}$	$0.336^{+0.043}_{-0.040}$				
					3	21.79	$0.378^{+0.043}_{-0.041}$	$0.264^{+0.040}_{-0.037}$				

Table 5: **Frontier table, MATH-500, 4-shot.** LLaDA-8B-Instruct, canvas 256, greedy, no KV cache; problems $[0, 500)$ (HumanEval all 164). Accuracy with its Wilson 95% interval; flexible and strict extraction. Every operating point behind Figure 18 and Figure 19, including cells beyond the figures' crops; PIG.EXT is the composite operating rule of every figure: its two slowest rows additionally floor admission at a confidence ν (written θ ($\nu{\geq}c$)), then the plain θ ladder from 0.5.

regime	Fast-dLLM				PIG.INT				PIG.EXT			
	ν	TPF	acc (flex)	acc (strict)	φ	TPF	acc (flex)	acc (strict)	θ	TPF	acc (flex)	acc (strict)
BL=32 fixed	0.95	2.21	$0.364^{+0.043}_{-0.041}$	$0.356^{+0.043}_{-0.041}$	0.015	2.83	$0.360^{+0.043}_{-0.041}$	$0.354^{+0.043}_{-0.041}$	0.125 ($\nu{\geq}0.9$)	2.54	$0.362^{+0.043}_{-0.041}$	$0.352^{+0.043}_{-0.041}$
	0.85	2.88	$0.364^{+0.043}_{-0.041}$	$0.354^{+0.043}_{-0.041}$	0.03	3.23	$0.356^{+0.043}_{-0.041}$	$0.348^{+0.043}_{-0.040}$	0.25 ($\nu{\geq}0.8$)	3.09	$0.360^{+0.043}_{-0.041}$	$0.356^{+0.043}_{-0.041}$
	0.7	3.87	$0.332^{+0.042}_{-0.040}$	$0.330^{+0.042}_{-0.040}$	0.05	3.52	$0.342^{+0.043}_{-0.040}$	$0.334^{+0.042}_{-0.040}$	0.5	4.57	$0.334^{+0.042}_{-0.040}$	$0.322^{+0.042}_{-0.039}$
	0.5	5.39	$0.290^{+0.041}_{-0.038}$	$0.268^{+0.040}_{-0.037}$	0.1	4.00	$0.322^{+0.042}_{-0.039}$	$0.318^{+0.042}_{-0.039}$	1	6.23	$0.260^{+0.040}_{-0.037}$	$0.250^{+0.040}_{-0.036}$
	0.3	7.14	$0.152^{+0.034}_{-0.029}$	$0.112^{+0.031}_{-0.025}$	0.2	4.84	$0.344^{+0.043}_{-0.040}$	$0.342^{+0.043}_{-0.040}$	1.5	8.10	$0.184^{+0.036}_{-0.032}$	$0.172^{+0.036}_{-0.031}$
	0.2	10.27	$0.066^{+0.025}_{-0.019}$	$0.030^{+0.019}_{-0.012}$	0.3	5.44	$0.332^{+0.042}_{-0.040}$	$0.318^{+0.042}_{-0.039}$	2	11.03	$0.128^{+0.032}_{-0.026}$	$0.110^{+0.030}_{-0.025}$
					0.5	6.64	$0.310^{+0.042}_{-0.039}$	$0.302^{+0.042}_{-0.039}$	2.5	13.74	$0.100^{+0.029}_{-0.023}$	$0.082^{+0.027}_{-0.021}$
					1	7.61	$0.232^{+0.039}_{-0.035}$	$0.216^{+0.038}_{-0.034}$	3	17.05	$0.058^{+0.024}_{-0.017}$	$0.040^{+0.021}_{-0.014}$
BL=64 fixed	0.95	2.24	$0.380^{+0.043}_{-0.041}$	$0.374^{+0.043}_{-0.041}$	0.015	2.91	$0.378^{+0.043}_{-0.041}$	$0.374^{+0.043}_{-0.041}$	0.125 ($\nu{\geq}0.9$)	2.59	$0.386^{+0.043}_{-0.042}$	$0.380^{+0.043}_{-0.041}$
	0.85	2.98	$0.380^{+0.043}_{-0.041}$	$0.368^{+0.043}_{-0.041}$	0.03	3.36	$0.378^{+0.043}_{-0.041}$	$0.372^{+0.043}_{-0.041}$	0.25 ($\nu{\geq}0.8$)	3.19	$0.372^{+0.043}_{-0.041}$	$0.368^{+0.043}_{-0.041}$
	0.7	4.13	$0.342^{+0.043}_{-0.040}$	$0.340^{+0.043}_{-0.040}$	0.05	3.70	$0.370^{+0.043}_{-0.041}$	$0.368^{+0.043}_{-0.041}$	0.5	4.68	$0.308^{+0.042}_{-0.039}$	$0.306^{+0.042}_{-0.039}$
	0.5	6.04	$0.274^{+0.041}_{-0.037}$	$0.260^{+0.040}_{-0.037}$	0.1	4.28	$0.360^{+0.043}_{-0.041}$	$0.358^{+0.043}_{-0.041}$	1	6.42	$0.302^{+0.042}_{-0.039}$	$0.298^{+0.042}_{-0.038}$
	0.3	8.22	$0.176^{+0.036}_{-0.031}$	$0.122^{+0.032}_{-0.026}$	0.2	5.27	$0.358^{+0.043}_{-0.041}$	$0.354^{+0.043}_{-0.041}$	1.5	8.97	$0.216^{+0.038}_{-0.034}$	$0.210^{+0.038}_{-0.033}$
	0.2	12.20	$0.098^{+0.029}_{-0.023}$	$0.042^{+0.021}_{-0.014}$	0.3	6.07	$0.314^{+0.042}_{-0.039}$	$0.308^{+0.042}_{-0.039}$	2	12.40	$0.162^{+0.035}_{-0.030}$	$0.148^{+0.034}_{-0.028}$
					0.5	7.65	$0.294^{+0.041}_{-0.038}$	$0.292^{+0.041}_{-0.038}$	2.5	16.72	$0.120^{+0.031}_{-0.026}$	$0.110^{+0.030}_{-0.025}$
					1	9.46	$0.278^{+0.041}_{-0.037}$	$0.258^{+0.040}_{-0.036}$				
					2	11.51	$0.192^{+0.037}_{-0.032}$	$0.176^{+0.036}_{-0.031}$				
					3	13.44	$0.154^{+0.034}_{-0.029}$	$0.106^{+0.030}_{-0.024}$				
BL=32 sliding	0.95	2.25	$0.376^{+0.043}_{-0.041}$	$0.368^{+0.043}_{-0.041}$	0.015	2.96	$0.376^{+0.043}_{-0.041}$	$0.370^{+0.043}_{-0.041}$	0.125 ($\nu{\geq}0.9$)	2.63	$0.380^{+0.043}_{-0.041}$	$0.370^{+0.043}_{-0.041}$
	0.85	3.03	$0.382^{+0.043}_{-0.042}$	$0.372^{+0.043}_{-0.041}$	0.03	3.44	$0.350^{+0.043}_{-0.041}$	$0.346^{+0.043}_{-0.040}$	0.25 ($\nu{\geq}0.8$)	3.25	$0.360^{+0.043}_{-0.041}$	$0.354^{+0.043}_{-0.041}$
	0.7	4.25	$0.342^{+0.043}_{-0.040}$	$0.338^{+0.043}_{-0.040}$	0.05	3.78	$0.358^{+0.043}_{-0.041}$	$0.354^{+0.043}_{-0.041}$	0.5	4.56	$0.334^{+0.042}_{-0.040}$	$0.324^{+0.042}_{-0.040}$
	0.5	6.18	$0.264^{+0.040}_{-0.037}$	$0.248^{+0.040}_{-0.036}$	0.1	4.38	$0.348^{+0.043}_{-0.040}$	$0.338^{+0.043}_{-0.040}$	1	6.02	$0.286^{+0.041}_{-0.038}$	$0.280^{+0.041}_{-0.038}$
	0.3	8.45	$0.132^{+0.032}_{-0.027}$	$0.106^{+0.030}_{-0.024}$	0.2	5.37	$0.346^{+0.043}_{-0.040}$	$0.340^{+0.043}_{-0.040}$	1.5	8.13	$0.240^{+0.039}_{-0.035}$	$0.232^{+0.039}_{-0.035}$
	0.2	12.12	$0.066^{+0.025}_{-0.019}$	$0.034^{+0.020}_{-0.013}$	0.3	6.20	$0.322^{+0.042}_{-0.039}$	$0.304^{+0.042}_{-0.039}$	2	10.68	$0.172^{+0.036}_{-0.031}$	$0.152^{+0.034}_{-0.029}$
					0.5	8.17	$0.296^{+0.041}_{-0.038}$	$0.290^{+0.041}_{-0.038}$	2.5	13.59	$0.098^{+0.029}_{-0.023}$	$0.080^{+0.027}_{-0.021}$
					1	9.74	$0.264^{+0.040}_{-0.037}$	$0.246^{+0.040}_{-0.036}$	3	17.08	$0.080^{+0.027}_{-0.021}$	$0.058^{+0.024}_{-0.017}$
BL=64 sliding	0.95	2.24	$0.386^{+0.043}_{-0.042}$	$0.378^{+0.043}_{-0.041}$	0.015	2.96	$0.382^{+0.043}_{-0.042}$	$0.378^{+0.043}_{-0.041}$	0.125 ($\nu{\geq}0.9$)	2.61	$0.396^{+0.044}_{-0.042}$	$0.388^{+0.043}_{-0.042}$
	0.85	3.05	$0.374^{+0.043}_{-0.041}$	$0.366^{+0.043}_{-0.041}$	0.03	3.45	$0.356^{+0.043}_{-0.041}$	$0.350^{+0.043}_{-0.041}$	0.25 ($\nu{\geq}0.8$)	3.26	$0.366^{+0.043}_{-0.041}$	$0.362^{+0.043}_{-0.041}$
	0.7	4.35	$0.334^{+0.042}_{-0.040}$	$0.332^{+0.042}_{-0.040}$	0.05	3.84	$0.366^{+0.043}_{-0.041}$	$0.364^{+0.043}_{-0.041}$	0.5	4.72	$0.342^{+0.043}_{-0.040}$	$0.334^{+0.042}_{-0.040}$
	0.5	6.47	$0.278^{+0.041}_{-0.037}$	$0.270^{+0.041}_{-0.037}$	0.1	4.50	$0.370^{+0.043}_{-0.041}$	$0.366^{+0.043}_{-0.041}$	1	6.71	$0.288^{+0.041}_{-0.038}$	$0.282^{+0.041}_{-0.038}$
	0.3	9.16	$0.158^{+0.035}_{-0.029}$	$0.118^{+0.031}_{-0.025}$	0.2	5.58	$0.360^{+0.043}_{-0.041}$	$0.354^{+0.043}_{-0.041}$	1.5	9.69	$0.232^{+0.039}_{-0.035}$	$0.222^{+0.038}_{-0.034}$
	0.2	14.00	$0.082^{+0.027}_{-0.021}$	$0.034^{+0.020}_{-0.013}$	0.3	6.45	$0.340^{+0.043}_{-0.040}$	$0.332^{+0.042}_{-0.040}$	2	13.34	$0.152^{+0.034}_{-0.029}$	$0.140^{+0.033}_{-0.028}$
					0.5	8.66	$0.282^{+0.041}_{-0.038}$	$0.274^{+0.041}_{-0.037}$	2.5	17.31	$0.120^{+0.031}_{-0.026}$	$0.108^{+0.030}_{-0.024}$
					1	11.23	$0.264^{+0.040}_{-0.037}$	$0.248^{+0.040}_{-0.036}$				
					2	13.85	$0.220^{+0.038}_{-0.034}$	$0.202^{+0.037}_{-0.033}$				
					3	16.27	$0.162^{+0.035}_{-0.030}$	$0.132^{+0.032}_{-0.027}$				
free canvas	0.95	2.01	$0.254^{+0.040}_{-0.036}$	$0.252^{+0.040}_{-0.036}$	0.015	2.58	$0.252^{+0.040}_{-0.036}$	$0.250^{+0.040}_{-0.036}$	0.125 ($\nu{\geq}0.9$)	2.31	$0.252^{+0.040}_{-0.036}$	$0.248^{+0.040}_{-0.036}$
	0.85	2.64	$0.250^{+0.040}_{-0.036}$	$0.246^{+0.040}_{-0.036}$	0.03	3.02	$0.250^{+0.040}_{-0.036}$	$0.246^{+0.040}_{-0.036}$	0.25 ($\nu{\geq}0.8$)	2.87	$0.252^{+0.040}_{-0.036}$	$0.248^{+0.040}_{-0.036}$
	0.7	3.80	$0.238^{+0.039}_{-0.035}$	$0.234^{+0.039}_{-0.035}$	0.05	3.39	$0.236^{+0.039}_{-0.035}$	$0.232^{+0.039}_{-0.035}$	0.5	4.37	$0.230^{+0.039}_{-0.035}$	$0.226^{+0.039}_{-0.034}$
	0.5	6.47	$0.202^{+0.037}_{-0.033}$	$0.190^{+0.037}_{-0.032}$	0.1	4.01	$0.230^{+0.039}_{-0.035}$	$0.226^{+0.039}_{-0.034}$	1	5.95	$0.216^{+0.038}_{-0.034}$	$0.212^{+0.038}_{-0.034}$
	0.3	12.66	$0.120^{+0.031}_{-0.026}$	$0.094^{+0.029}_{-0.023}$	0.2	5.76	$0.240^{+0.039}_{-0.035}$	$0.234^{+0.039}_{-0.035}$	1.5	7.92	$0.210^{+0.038}_{-0.033}$	$0.208^{+0.038}_{-0.033}$
	0.2	26.54	$0.094^{+0.029}_{-0.023}$	$0.044^{+0.022}_{-0.015}$	0.3	7.14	$0.230^{+0.039}_{-0.035}$	$0.226^{+0.039}_{-0.034}$	2	10.47	$0.222^{+0.038}_{-0.034}$	$0.222^{+0.038}_{-0.034}$
					0.5	9.89	$0.222^{+0.038}_{-0.034}$	$0.220^{+0.038}_{-0.034}$	2.5	13.85	$0.202^{+0.037}_{-0.033}$	$0.200^{+0.037}_{-0.033}$
					1	12.65	$0.190^{+0.037}_{-0.032}$	$0.180^{+0.036}_{-0.031}$	3	19.16	$0.218^{+0.038}_{-0.034}$	$0.210^{+0.038}_{-0.033}$
					2	15.68	$0.180^{+0.036}_{-0.031}$	$0.164^{+0.035}_{-0.030}$				
					3	18.31	$0.146^{+0.034}_{-0.028}$	$0.130^{+0.032}_{-0.027}$				

Table 6: **Frontier table, MBPP, 3-shot.** LLaDA-8B-Instruct, canvas 256, greedy, no KV cache; problems $[0, 500)$ (HumanEval all 164). Accuracy with its Wilson 95% interval; pass@1. Every operating point behind Figure 18 and Figure 19, including cells beyond the figures' crops; PIG.EXT is the composite operating rule of every figure: its two slowest rows additionally floor admission at a confidence ν (written θ ($\nu{\geq}c$)), then the plain θ ladder from 0.5.

regime	Fast-dLLM			PIG.INT			PIG.EXT		
	ν	TPF	acc	φ	TPF	acc	θ	TPF	acc
BL=32 fixed	0.95	3.29	$0.396^{+0.044}_{-0.042}$	0.015	4.21	$0.394^{+0.043}_{-0.042}$	0.125 ($\nu{\geq}0.9$)	3.78	$0.392^{+0.043}_{-0.042}$
	0.85	4.26	$0.388^{+0.043}_{-0.042}$	0.03	4.73	$0.394^{+0.043}_{-0.042}$	0.25 ($\nu{\geq}0.8$)	4.50	$0.398^{+0.044}_{-0.042}$
	0.7	5.53	$0.372^{+0.043}_{-0.041}$	0.05	5.16	$0.386^{+0.043}_{-0.042}$	0.5	7.15	$0.260^{+0.040}_{-0.037}$
	0.5	7.99	$0.230^{+0.039}_{-0.035}$	0.1	5.73	$0.366^{+0.043}_{-0.041}$	1	9.76	$0.214^{+0.038}_{-0.034}$
	0.3	12.10	$0.068^{+0.026}_{-0.019}$	0.2	6.87	$0.332^{+0.042}_{-0.040}$	1.5	13.50	$0.126^{+0.032}_{-0.026}$
	0.2	16.35	$0.028^{+0.018}_{-0.011}$	0.3	8.05	$0.308^{+0.042}_{-0.039}$	2	17.00	$0.084^{+0.028}_{-0.021}$
				0.5	9.98	$0.236^{+0.039}_{-0.035}$			
				1	11.41	$0.150^{+0.034}_{-0.029}$			
BL=64 fixed	0.99	2.59	$0.396^{+0.044}_{-0.042}$	0.015	4.29	$0.396^{+0.044}_{-0.042}$	0.125 ($\nu{\geq}0.9$)	3.89	$0.392^{+0.043}_{-0.042}$
	0.95	3.38	$0.394^{+0.043}_{-0.042}$	0.03	5.00	$0.384^{+0.043}_{-0.042}$	0.25 ($\nu{\geq}0.8$)	4.67	$0.388^{+0.043}_{-0.042}$
	0.85	4.42	$0.382^{+0.043}_{-0.042}$	0.05	5.53	$0.372^{+0.043}_{-0.041}$	0.5	7.78	$0.250^{+0.040}_{-0.036}$
	0.7	5.94	$0.354^{+0.043}_{-0.041}$	0.1	6.19	$0.356^{+0.043}_{-0.041}$	1	11.66	$0.204^{+0.038}_{-0.033}$
	0.5	9.54	$0.222^{+0.038}_{-0.034}$	0.2	7.86	$0.328^{+0.042}_{-0.040}$	1.5	18.52	$0.144^{+0.033}_{-0.028}$
	0.3	16.15	$0.048^{+0.022}_{-0.016}$	0.3	9.72	$0.300^{+0.042}_{-0.039}$			
				0.5	12.66	$0.224^{+0.039}_{-0.034}$			
				1	14.96	$0.132^{+0.032}_{-0.027}$			
BL=32 sliding	0.95	3.44	$0.404^{+0.044}_{-0.042}$	0.015	4.45	$0.406^{+0.044}_{-0.042}$	0.125 ($\nu{\geq}0.9$)	3.95	$0.404^{+0.044}_{-0.042}$
	0.85	4.55	$0.392^{+0.043}_{-0.042}$	0.03	5.13	$0.396^{+0.044}_{-0.042}$	0.25 ($\nu{\geq}0.8$)	4.78	$0.402^{+0.044}_{-0.042}$
	0.7	5.99	$0.358^{+0.043}_{-0.041}$	0.05	5.61	$0.382^{+0.043}_{-0.042}$	0.5	7.33	$0.272^{+0.041}_{-0.037}$
	0.5	9.41	$0.212^{+0.038}_{-0.034}$	0.1	6.08	$0.360^{+0.043}_{-0.041}$	1	9.97	$0.236^{+0.039}_{-0.035}$
	0.3	14.02	$0.050^{+0.023}_{-0.016}$	0.2	7.90	$0.310^{+0.042}_{-0.039}$	1.5	13.15	$0.156^{+0.034}_{-0.029}$
	0.2	19.57	$0.020^{+0.016}_{-0.009}$	0.3	9.50	$0.276^{+0.041}_{-0.037}$	2	16.94	$0.104^{+0.030}_{-0.024}$
				0.5	11.94	$0.244^{+0.040}_{-0.036}$			
				1	14.00	$0.138^{+0.033}_{-0.027}$			
BL=64 sliding	0.95	3.52	$0.396^{+0.044}_{-0.042}$	0.015	4.58	$0.402^{+0.044}_{-0.042}$	0.125 ($\nu{\geq}0.9$)	4.03	$0.396^{+0.044}_{-0.042}$
	0.85	4.70	$0.388^{+0.043}_{-0.042}$	0.03	5.30	$0.394^{+0.043}_{-0.042}$	0.25 ($\nu{\geq}0.8$)	4.94	$0.392^{+0.043}_{-0.042}$
	0.7	6.56	$0.354^{+0.043}_{-0.041}$	0.05	6.06	$0.370^{+0.043}_{-0.041}$	0.5	8.65	$0.246^{+0.040}_{-0.036}$
	0.5	11.17	$0.208^{+0.038}_{-0.033}$	0.1	6.74	$0.350^{+0.043}_{-0.041}$	1	14.23	$0.216^{+0.038}_{-0.034}$
	0.3	19.63	$0.042^{+0.021}_{-0.014}$	0.2	8.93	$0.316^{+0.042}_{-0.039}$			
				0.3	11.34	$0.292^{+0.041}_{-0.038}$			
				0.5	14.58	$0.202^{+0.037}_{-0.033}$			
				1	18.25	$0.136^{+0.033}_{-0.027}$			
free canvas	0.99	2.08	$0.354^{+0.043}_{-0.041}$	0.015	3.44	$0.358^{+0.043}_{-0.041}$	0.125 ($\nu{\geq}0.9$)	3.05	$0.350^{+0.043}_{-0.041}$
	0.95	2.66	$0.354^{+0.043}_{-0.041}$	0.03	3.96	$0.358^{+0.043}_{-0.041}$	0.25 ($\nu{\geq}0.8$)	3.70	$0.356^{+0.043}_{-0.041}$
	0.85	3.54	$0.356^{+0.043}_{-0.041}$	0.05	4.58	$0.306^{+0.042}_{-0.039}$	0.5	5.57	$0.256^{+0.040}_{-0.036}$
	0.7	4.48	$0.308^{+0.042}_{-0.039}$	0.1	4.72	$0.296^{+0.041}_{-0.038}$	1	8.75	$0.222^{+0.038}_{-0.034}$
	0.5	6.19	$0.204^{+0.038}_{-0.033}$	0.2	5.63	$0.252^{+0.040}_{-0.036}$	1.5	14.43	$0.144^{+0.033}_{-0.028}$
	0.3	15.11	$0.054^{+0.023}_{-0.017}$	0.3	6.62	$0.242^{+0.039}_{-0.035}$	2	23.12	$0.086^{+0.028}_{-0.022}$
				0.5	9.27	$0.220^{+0.038}_{-0.034}$			
				1	13.00	$0.156^{+0.034}_{-0.029}$			

Table 7: **Frontier table, HumanEval, 0-shot.** LLaDA-8B-Instruct, canvas 256, greedy, no KV cache; problems $[0, 500)$ (HumanEval all 164). Accuracy with its Wilson 95% interval; pass@1. Every operating point behind Figure 18 and Figure 19, including cells beyond the figures' crops; PIG.EXT is the composite operating rule of every figure: its two slowest rows additionally floor admission at a confidence ν (written θ ($\nu{\geq}c$)), then the plain θ ladder from 0.5.

regime	Fast-dLLM			PIG.INT			PIG.EXT		
	ν	TPF	acc	φ	TPF	acc	θ	TPF	acc
BL=32 fixed	0.95	4.95	$0.396^{+0.076}_{-0.072}$	0.015	6.42	$0.378^{+0.076}_{-0.071}$	0.125 ($\nu{\geq}0.9$)	5.69	$0.390^{+0.076}_{-0.071}$
	0.85	6.76	$0.396^{+0.076}_{-0.072}$	0.03	7.37	$0.366^{+0.076}_{-0.070}$	0.25 ($\nu{\geq}0.8$)	6.77	$0.396^{+0.076}_{-0.072}$
	0.7	8.69	$0.342^{+0.075}_{-0.068}$	0.05	7.79	$0.372^{+0.076}_{-0.070}$	0.5	8.50	$0.274^{+0.073}_{-0.063}$
	0.5	10.64	$0.171^{+0.065}_{-0.050}$	0.1	8.67	$0.335^{+0.075}_{-0.068}$	1	10.46	$0.226^{+0.070}_{-0.057}$
	0.3	12.92	$0.018^{+0.034}_{-0.012}$	0.2	9.59	$0.317^{+0.075}_{-0.066}$	1.5	13.03	$0.207^{+0.068}_{-0.055}$
	0.2	17.41	$0.018^{+0.034}_{-0.012}$	0.3	10.64	$0.281^{+0.073}_{-0.063}$	2	15.64	$0.177^{+0.066}_{-0.051}$
				0.5	11.60	$0.238^{+0.071}_{-0.059}$	2.5	17.69	$0.122^{+0.059}_{-0.042}$
				1	13.10	$0.104^{+0.056}_{-0.038}$			
BL=64 fixed	0.9	3.34	$0.348^{+0.076}_{-0.069}$	0.015	6.68	$0.372^{+0.076}_{-0.070}$	0.125 ($\nu{\geq}0.9$)	6.00	$0.384^{+0.076}_{-0.071}$
	0.99	3.66	$0.378^{+0.076}_{-0.071}$	0.03	8.14	$0.348^{+0.076}_{-0.069}$	0.25 ($\nu{\geq}0.8$)	7.54	$0.384^{+0.076}_{-0.071}$
	0.95	5.16	$0.390^{+0.076}_{-0.071}$	0.05	8.88	$0.366^{+0.076}_{-0.070}$	0.5	9.47	$0.293^{+0.074}_{-0.064}$
	0.85	7.21	$0.360^{+0.076}_{-0.069}$	0.1	10.06	$0.342^{+0.075}_{-0.068}$	1	12.32	$0.262^{+0.072}_{-0.061}$
	0.7	9.79	$0.317^{+0.075}_{-0.066}$	0.2	11.27	$0.274^{+0.073}_{-0.063}$	1.5	14.48	$0.201^{+0.068}_{-0.054}$
	0.5	12.73	$0.140^{+0.061}_{-0.045}$	0.3	13.01	$0.287^{+0.073}_{-0.064}$	2	18.69	$0.165^{+0.064}_{-0.049}$
	0.3	17.90	$0.030^{+0.039}_{-0.017}$	0.5	14.70	$0.213^{+0.069}_{-0.056}$			
				1	17.37	$0.110^{+0.057}_{-0.039}$			
				2	19.93	$0.049^{+0.045}_{-0.024}$			
BL=32 sliding	0.95	5.25	$0.396^{+0.076}_{-0.072}$	0.015	6.82	$0.415^{+0.077}_{-0.073}$	0.125 ($\nu{\geq}0.9$)	6.00	$0.402^{+0.076}_{-0.072}$
	0.85	7.04	$0.378^{+0.076}_{-0.071}$	0.03	8.04	$0.366^{+0.076}_{-0.070}$	0.25 ($\nu{\geq}0.8$)	7.45	$0.408^{+0.076}_{-0.072}$
	0.7	9.54	$0.317^{+0.075}_{-0.066}$	0.05	8.68	$0.402^{+0.076}_{-0.072}$	0.5	8.85	$0.342^{+0.075}_{-0.068}$
	0.5	12.21	$0.128^{+0.060}_{-0.043}$	0.1	9.39	$0.329^{+0.075}_{-0.067}$	1	10.72	$0.262^{+0.072}_{-0.061}$
	0.3	15.72	$0.037^{+0.041}_{-0.020}$	0.2	10.96	$0.299^{+0.074}_{-0.065}$	1.5	13.58	$0.244^{+0.071}_{-0.059}$
				0.3	11.83	$0.238^{+0.071}_{-0.059}$	2	15.76	$0.171^{+0.065}_{-0.050}$
				0.5	13.99	$0.171^{+0.065}_{-0.050}$	2.5	17.62	$0.152^{+0.063}_{-0.047}$
				1	15.44	$0.091^{+0.054}_{-0.035}$			
BL=64 sliding	0.95	5.36	$0.396^{+0.076}_{-0.072}$	0.015	7.25	$0.384^{+0.076}_{-0.071}$	0.125 ($\nu{\geq}0.9$)	6.13	$0.390^{+0.076}_{-0.071}$
	0.85	7.58	$0.378^{+0.076}_{-0.071}$	0.03	8.83	$0.378^{+0.076}_{-0.071}$	0.25 ($\nu{\geq}0.8$)	8.11	$0.384^{+0.076}_{-0.071}$
	0.7	10.85	$0.348^{+0.076}_{-0.069}$	0.05	9.50	$0.366^{+0.076}_{-0.070}$	0.5	10.12	$0.323^{+0.075}_{-0.067}$
	0.5	15.37	$0.134^{+0.061}_{-0.044}$	0.1	10.63	$0.348^{+0.076}_{-0.069}$	1	13.57	$0.293^{+0.074}_{-0.064}$
				0.2	12.42	$0.262^{+0.072}_{-0.061}$	1.5	16.98	$0.268^{+0.073}_{-0.062}$
				0.3	14.90	$0.287^{+0.073}_{-0.064}$			
				0.5	16.86	$0.207^{+0.068}_{-0.055}$			
free canvas	0.99	3.35	$0.299^{+0.074}_{-0.065}$	0.015	7.27	$0.317^{+0.075}_{-0.066}$	0.125 ($\nu{\geq}0.9$)	5.82	$0.287^{+0.073}_{-0.064}$
	0.95	5.06	$0.323^{+0.075}_{-0.067}$	0.03	8.68	$0.293^{+0.074}_{-0.064}$	0.25 ($\nu{\geq}0.8$)	7.65	$0.287^{+0.073}_{-0.064}$
	0.85	7.96	$0.268^{+0.073}_{-0.062}$	0.05	10.10	$0.274^{+0.073}_{-0.063}$	0.5	10.60	$0.274^{+0.073}_{-0.063}$
	0.7	11.71	$0.262^{+0.072}_{-0.061}$	0.1	12.47	$0.262^{+0.072}_{-0.061}$	1	14.27	$0.299^{+0.074}_{-0.065}$
	0.5	16.54	$0.140^{+0.061}_{-0.045}$	0.2	15.18	$0.250^{+0.071}_{-0.060}$	1.5	19.13	$0.244^{+0.071}_{-0.059}$
	0.3	27.35	$0.024^{+0.037}_{-0.015}$	0.3	16.98	$0.250^{+0.071}_{-0.060}$	2	23.53	$0.159^{+0.064}_{-0.048}$
				0.5	19.68	$0.159^{+0.064}_{-0.048}$			
				1	22.48	$0.128^{+0.060}_{-0.043}$			
				2	27.33	$0.091^{+0.054}_{-0.035}$			
				3	29.71	$0.055^{+0.046}_{-0.026}$			

Table 8: **Frontier table, Nemotron-3B, GSM8K, 0-shot (flex).** Nemotron-Labs-Diffusion-3B, canvas 2048, 0-shot, greedy; the Fast-dLLM lane is the model's own confidence-threshold decoder at the same geometry; PIG.EXT is the composite operating rule (the two slowest points additionally floor admission at a confidence ν, written θ ($\nu \geq c$)). Accuracy with its Wilson 95% interval; the full three-geometry grid is Figure 20.

regime	Fast-dLLM			PIG.INT			PIG.EXT		
	ν	TPF	acc	φ	TPF	acc	θ	TPF	acc
BL=32 fixed	0.9	2.56	$0.896^{+0.024}_{-0.030}$	0.03	2.52	$0.854^{+0.028}_{-0.034}$	0.125	2.05	$0.870^{+0.027}_{-0.032}$
	0.8	3.08	$0.870^{+0.027}_{-0.032}$	0.05	2.91	$0.868^{+0.027}_{-0.032}$	0.25	2.44	$0.856^{+0.028}_{-0.033}$
	0.7	3.66	$0.856^{+0.028}_{-0.033}$	0.1	3.31	$0.846^{+0.029}_{-0.034}$	0.5	3.66	$0.806^{+0.032}_{-0.037}$
	0.6	4.22	$0.808^{+0.032}_{-0.037}$	0.2	3.87	$0.844^{+0.029}_{-0.034}$	1	4.69	$0.742^{+0.036}_{-0.040}$
	0.5	4.91	$0.738^{+0.037}_{-0.040}$	0.3	4.21	$0.796^{+0.033}_{-0.038}$	1.5	5.70	$0.668^{+0.040}_{-0.042}$
	0.4	5.66	$0.554^{+0.043}_{-0.044}$	0.5	4.71	$0.754^{+0.036}_{-0.040}$			
				0.7	5.09	$0.738^{+0.037}_{-0.040}$			
				1	5.44	$0.650^{+0.041}_{-0.043}$			
BL=32 sliding	0.9	2.42	$0.846^{+0.029}_{-0.034}$	0.03	2.62	$0.840^{+0.030}_{-0.035}$	0.125	2.00	$0.830^{+0.030}_{-0.035}$
	0.8	2.97	$0.850^{+0.029}_{-0.034}$	0.05	3.08	$0.828^{+0.031}_{-0.036}$	0.25	2.41	$0.842^{+0.029}_{-0.035}$
	0.7	3.54	$0.822^{+0.031}_{-0.036}$	0.1	3.54	$0.840^{+0.030}_{-0.035}$	0.5	3.56	$0.774^{+0.034}_{-0.039}$
	0.6	4.11	$0.758^{+0.035}_{-0.039}$	0.2	4.17	$0.826^{+0.031}_{-0.036}$	1	4.45	$0.786^{+0.034}_{-0.038}$
	0.5	4.75	$0.662^{+0.040}_{-0.043}$	0.3	4.61	$0.782^{+0.034}_{-0.038}$	1.5	5.74	$0.684^{+0.039}_{-0.042}$
	0.4	5.69	$0.526^{+0.043}_{-0.044}$	0.5	5.15	$0.750^{+0.036}_{-0.040}$			
				0.7	5.52	$0.704^{+0.038}_{-0.041}$			
				1	5.97	$0.622^{+0.041}_{-0.043}$			
BL=32 in 64-wide window	0.9	2.53	$0.852^{+0.028}_{-0.034}$	0.03	2.63	$0.862^{+0.027}_{-0.033}$	0.125	1.99	$0.862^{+0.027}_{-0.033}$
	0.8	3.16	$0.846^{+0.029}_{-0.034}$	0.05	3.07	$0.850^{+0.029}_{-0.034}$	0.25	2.40	$0.844^{+0.029}_{-0.034}$
	0.7	3.85	$0.824^{+0.031}_{-0.036}$	0.1	3.59	$0.838^{+0.030}_{-0.035}$	0.5	3.55	$0.836^{+0.030}_{-0.035}$
	0.6	4.50	$0.762^{+0.035}_{-0.039}$	0.2	4.18	$0.822^{+0.031}_{-0.036}$	1	4.52	$0.764^{+0.035}_{-0.039}$
	0.5	5.12	$0.652^{+0.040}_{-0.043}$	0.3	4.58	$0.780^{+0.034}_{-0.038}$	1.5	5.81	$0.690^{+0.039}_{-0.042}$
				0.5	5.17	$0.744^{+0.036}_{-0.040}$			
				0.7	5.57	$0.718^{+0.038}_{-0.041}$			
				1	5.93	$0.668^{+0.040}_{-0.042}$			

Table 9: **Frontier table, Nemotron-3B, MATH-500, 0-shot.** Nemotron-Labs-Diffusion-3B, canvas 2048, 0-shot, greedy; the Fast-dLLM lane is the model's own confidence-threshold decoder at the same geometry; PIG.EXT is the composite operating rule (the two slowest points additionally floor admission at a confidence ν, written θ ($\nu \geq c$)). Accuracy with its Wilson 95% interval; the full three-geometry grid is Figure 20.

regime	Fast-dLLM			PIG.INT			PIG.EXT		
	ν	TPF	acc	φ	TPF	acc	θ	TPF	acc
BL=32 fixed	0.9	2.80	$0.606^{+0.042}_{-0.043}$	0.03	2.72	$0.708^{+0.038}_{-0.041}$	0.125	2.24	$0.744^{+0.036}_{-0.040}$
	0.8	3.26	$0.598^{+0.042}_{-0.044}$	0.05	3.09	$0.686^{+0.039}_{-0.042}$	0.25	2.63	$0.708^{+0.038}_{-0.041}$
	0.7	3.78	$0.552^{+0.043}_{-0.044}$	0.1	3.45	$0.692^{+0.039}_{-0.042}$	0.5	3.96	$0.588^{+0.042}_{-0.044}$
	0.6	4.28	$0.498^{+0.044}_{-0.044}$	0.2	3.98	$0.614^{+0.042}_{-0.043}$	1	5.23	$0.492^{+0.044}_{-0.044}$
	0.5	4.95	$0.414^{+0.044}_{-0.042}$	0.3	4.37	$0.556^{+0.043}_{-0.044}$			
	0.4	5.61	$0.308^{+0.042}_{-0.039}$	0.5	4.87	$0.480^{+0.044}_{-0.043}$			
				0.7	5.30	$0.446^{+0.044}_{-0.043}$			
				1	5.69	$0.418^{+0.044}_{-0.042}$			
BL=32 sliding	0.9	2.70	$0.688^{+0.039}_{-0.042}$	0.03	2.86	$0.628^{+0.041}_{-0.043}$	0.125	2.22	$0.664^{+0.040}_{-0.043}$
	0.8	3.20	$0.628^{+0.041}_{-0.043}$	0.05	3.28	$0.658^{+0.040}_{-0.043}$	0.25	2.63	$0.680^{+0.039}_{-0.042}$
	0.7	3.71	$0.612^{+0.042}_{-0.043}$	0.1	3.76	$0.612^{+0.042}_{-0.043}$	0.5	3.87	$0.582^{+0.042}_{-0.044}$
	0.6	4.22	$0.476^{+0.044}_{-0.043}$	0.2	4.34	$0.552^{+0.043}_{-0.044}$	1	5.00	$0.494^{+0.044}_{-0.044}$
	0.5	4.91	$0.392^{+0.043}_{-0.042}$	0.3	4.80	$0.530^{+0.043}_{-0.044}$			
	0.4	5.66	$0.260^{+0.040}_{-0.037}$	0.5	5.28	$0.452^{+0.044}_{-0.043}$			
				0.7	5.74	$0.394^{+0.043}_{-0.042}$			
BL=32 in 64-wide window	0.9	2.84	$0.628^{+0.041}_{-0.043}$	0.03	2.87	$0.644^{+0.041}_{-0.043}$	0.125	2.19	$0.662^{+0.040}_{-0.043}$
	0.8	3.41	$0.588^{+0.042}_{-0.044}$	0.05	3.29	$0.628^{+0.041}_{-0.043}$	0.25	2.61	$0.666^{+0.040}_{-0.042}$
	0.7	4.03	$0.544^{+0.043}_{-0.044}$	0.1	3.74	$0.594^{+0.042}_{-0.044}$	0.5	3.86	$0.574^{+0.043}_{-0.044}$
	0.6	4.60	$0.474^{+0.044}_{-0.043}$	0.2	4.36	$0.572^{+0.043}_{-0.044}$	1	5.32	$0.462^{+0.044}_{-0.043}$
	0.5	5.25	$0.370^{+0.043}_{-0.041}$	0.3	4.78	$0.524^{+0.043}_{-0.044}$			
				0.5	5.30	$0.480^{+0.044}_{-0.043}$			
				0.7	5.73	$0.404^{+0.044}_{-0.042}$			

Table 10: **Frontier table, Nemotron-3B, MBPP, 0-shot.** Nemotron-Labs-Diffusion-3B, canvas 2048, 0-shot, greedy; the Fast-dLLM lane is the model's own confidence-threshold decoder at the same geometry; PIG.EXT is the composite operating rule (the two slowest points additionally floor admission at a confidence ν, written θ ($\nu \geq c$)). Accuracy with its Wilson 95% interval; the full three-geometry grid is Figure 20.

regime	Fast-dLLM			PIG.INT			PIG.EXT		
	ν	TPF	acc	φ	TPF	acc	θ	TPF	acc
BL=32 fixed	0.9	1.91	$0.553^{+0.043}_{-0.044}$	0.03	1.92	$0.548^{+0.043}_{-0.044}$	0.125	1.64	$0.574^{+0.043}_{-0.044}$
	0.8	2.26	$0.485^{+0.044}_{-0.044}$	0.05	2.18	$0.526^{+0.043}_{-0.044}$	0.25	1.89	$0.562^{+0.043}_{-0.044}$
	0.7	2.65	$0.433^{+0.044}_{-0.043}$	0.1	2.47	$0.504^{+0.044}_{-0.044}$	0.5	3.06	$0.412^{+0.044}_{-0.042}$
	0.6	3.11	$0.343^{+0.043}_{-0.040}$	0.2	2.87	$0.448^{+0.044}_{-0.043}$	1	5.20	$0.286^{+0.041}_{-0.038}$
	0.5	3.62	$0.222^{+0.039}_{-0.034}$	0.3	3.16	$0.408^{+0.044}_{-0.042}$			
	0.4	4.23	$0.136^{+0.033}_{-0.027}$	0.5	3.60	$0.316^{+0.042}_{-0.039}$			
	0.3	5.30	$0.074^{+0.026}_{-0.020}$	0.7	3.97	$0.292^{+0.041}_{-0.038}$			
				1	4.36	$0.204^{+0.038}_{-0.033}$			
BL=32 sliding	0.9	1.81	$0.565^{+0.043}_{-0.044}$	0.03	1.95	$0.560^{+0.043}_{-0.044}$	0.125	1.60	$0.596^{+0.042}_{-0.044}$
	0.8	2.17	$0.537^{+0.043}_{-0.044}$	0.05	2.24	$0.542^{+0.043}_{-0.044}$	0.25	1.87	$0.552^{+0.043}_{-0.044}$
	0.7	2.53	$0.403^{+0.044}_{-0.042}$	0.1	2.53	$0.474^{+0.044}_{-0.043}$	0.5	2.99	$0.432^{+0.044}_{-0.043}$
	0.6	2.97	$0.335^{+0.043}_{-0.040}$	0.2	2.98	$0.398^{+0.044}_{-0.042}$	1	5.27	$0.348^{+0.043}_{-0.040}$
	0.5	3.50	$0.222^{+0.039}_{-0.034}$	0.3	3.30	$0.314^{+0.042}_{-0.039}$			
	0.4	4.08	$0.122^{+0.032}_{-0.026}$	0.5	3.71	$0.308^{+0.042}_{-0.039}$			
	0.3	5.10	$0.050^{+0.023}_{-0.016}$	0.7	4.07	$0.226^{+0.039}_{-0.034}$			
				1	4.49	$0.188^{+0.037}_{-0.032}$			
BL=32 in 64-wide window	0.9	1.84	$0.539^{+0.043}_{-0.044}$	0.03	1.94	$0.540^{+0.043}_{-0.044}$	0.125	1.60	$0.572^{+0.043}_{-0.044}$
	0.8	2.25	$0.503^{+0.044}_{-0.044}$	0.05	2.24	$0.530^{+0.043}_{-0.044}$	0.25	1.87	$0.560^{+0.043}_{-0.044}$
	0.7	2.65	$0.407^{+0.044}_{-0.042}$	0.1	2.54	$0.482^{+0.044}_{-0.043}$	0.5	3.16	$0.464^{+0.044}_{-0.043}$
	0.6	3.09	$0.297^{+0.042}_{-0.038}$	0.2	2.97	$0.392^{+0.043}_{-0.042}$	1	5.58	$0.342^{+0.043}_{-0.040}$
	0.5	3.66	$0.170^{+0.035}_{-0.030}$	0.3	3.31	$0.346^{+0.043}_{-0.040}$			
	0.4	4.29	$0.090^{+0.028}_{-0.022}$	0.5	3.73	$0.278^{+0.041}_{-0.037}$			
	0.3	5.58	$0.050^{+0.023}_{-0.016}$	0.7	4.07	$0.242^{+0.039}_{-0.035}$			
				1	4.54	$0.196^{+0.037}_{-0.032}$			

Table 11: **Frontier table, Nemotron-3B, HumanEval, 0-shot.** Nemotron-Labs-Diffusion-3B, canvas 2048, 0-shot, greedy; the Fast-dLLM lane is the model's own confidence-threshold decoder at the same geometry; PIG.EXT is the composite operating rule (the two slowest points additionally floor admission at a confidence ν, written θ ($\nu \geq c$)). Accuracy with its Wilson 95% interval; the full three-geometry grid is Figure 20.

regime	Fast-dLLM			PIG.INT			PIG.EXT		
	ν	TPF	acc	φ	TPF	acc	θ	TPF	acc
BL=32 fixed	0.9	2.02	$0.750^{+0.060}_{-0.071}$	0.03	2.03	$0.756^{+0.059}_{-0.071}$	0.125	1.69	$0.793^{+0.055}_{-0.068}$
	0.8	2.41	$0.707^{+0.064}_{-0.074}$	0.05	2.30	$0.726^{+0.063}_{-0.073}$	0.25	2.00	$0.744^{+0.061}_{-0.072}$
	0.7	2.84	$0.543^{+0.074}_{-0.076}$	0.1	2.62	$0.707^{+0.064}_{-0.074}$	0.5	3.09	$0.512^{+0.075}_{-0.076}$
	0.6	3.34	$0.433^{+0.077}_{-0.073}$	0.2	3.06	$0.555^{+0.074}_{-0.076}$	1	4.59	$0.396^{+0.076}_{-0.072}$
	0.5	3.83	$0.329^{+0.075}_{-0.067}$	0.3	3.36	$0.427^{+0.077}_{-0.073}$			
	0.4	4.68	$0.146^{+0.062}_{-0.046}$	0.5	3.80	$0.348^{+0.076}_{-0.069}$			
	0.3	5.55	$0.067^{+0.049}_{-0.029}$	0.7	4.12	$0.250^{+0.071}_{-0.060}$			
				1	4.55	$0.177^{+0.066}_{-0.051}$			
BL=32 sliding	0.9	1.94	$0.713^{+0.064}_{-0.073}$	0.03	2.05	$0.726^{+0.063}_{-0.073}$	0.125	1.66	$0.744^{+0.061}_{-0.072}$
	0.8	2.32	$0.689^{+0.066}_{-0.074}$	0.05	2.36	$0.701^{+0.065}_{-0.074}$	0.25	1.96	$0.726^{+0.063}_{-0.073}$
	0.7	2.75	$0.567^{+0.073}_{-0.077}$	0.1	2.69	$0.640^{+0.069}_{-0.076}$	0.5	2.99	$0.573^{+0.073}_{-0.077}$
	0.6	3.22	$0.415^{+0.077}_{-0.073}$	0.2	3.16	$0.518^{+0.075}_{-0.076}$	1	4.24	$0.469^{+0.076}_{-0.075}$
	0.5	3.75	$0.238^{+0.071}_{-0.059}$	0.3	3.52	$0.482^{+0.076}_{-0.075}$			
	0.4	4.37	$0.104^{+0.056}_{-0.038}$	0.5	3.91	$0.348^{+0.076}_{-0.069}$			
	0.3	5.44	$0.037^{+0.041}_{-0.020}$	0.7	4.32	$0.305^{+0.074}_{-0.065}$			
				1	4.66	$0.213^{+0.069}_{-0.056}$			
BL=32 in 64-wide window	0.9	2.03	$0.701^{+0.065}_{-0.074}$	0.03	2.06	$0.689^{+0.066}_{-0.074}$	0.125	1.66	$0.720^{+0.063}_{-0.073}$
	0.8	2.46	$0.652^{+0.069}_{-0.076}$	0.05	2.36	$0.689^{+0.066}_{-0.074}$	0.25	1.94	$0.701^{+0.065}_{-0.074}$
	0.7	2.94	$0.543^{+0.074}_{-0.076}$	0.1	2.68	$0.604^{+0.072}_{-0.076}$	0.5	3.08	$0.604^{+0.072}_{-0.076}$
	0.6	3.53	$0.348^{+0.076}_{-0.069}$	0.2	3.18	$0.604^{+0.072}_{-0.076}$	1	4.36	$0.494^{+0.076}_{-0.075}$
	0.5	4.23	$0.220^{+0.069}_{-0.057}$	0.3	3.52	$0.451^{+0.076}_{-0.074}$			
	0.4	5.05	$0.116^{+0.058}_{-0.040}$	0.5	3.95	$0.329^{+0.075}_{-0.067}$			
				0.7	4.28	$0.311^{+0.074}_{-0.066}$			
				1	4.68	$0.238^{+0.071}_{-0.059}$			