TEST-TIME SCALING IN DIFFUSION LLMS VIA HIDDEN SEMI-AUTOREGRESSIVE EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion-based large language models (dLLMs) are trained to model extreme flexibility/dependence in the data-distribution; however, how to best utilize this at inference time remains an open problem. In this work, we uncover an interesting property of these models: dLLMs trained on textual data implicitly learn a mixture of semi-autoregressive experts, where different generation orders reveal different specialized behaviors. We show that committing to any single, fixed inference time schedule, a common practice, collapses performance by failing to leverage this latent ensemble. To address this, we introduce HEX (Hidden semi-autoregressive EXperts for test-time scaling), a training-free inference method that ensembles across heterogeneous block schedules. By doing a majority vote over diverse block-sized generation paths, HEX robustly avoids failure modes associated with any single fixed schedule. On reasoning benchmarks such as GSM8K, it boosts accuracy by up to 3.56× (from 24.72% to 88.10%), outperforming top-K margin inference and specialized fine-tuned methods like GRPO, without additional training. HEX even yields significant gains on MATH benchmark from 16.40% to 40.00%, scientific reasoning on ARC-C from 54.18% to 87.80%, and TruthfulQA from 28.36% to 57.46%. Our results establish test-time scaling as a powerful principle for dLLMs, showing that the sequence in which masking is done can play a significant role in test-time scaling/inferencing of dLLMs.

1 INTRODUCTION

Diffusion-based large language models (dLLMs) are rapidly emerging as a promising alternative to traditional autoregressive LLMs generalizing beyond the next token prediction (Nie et al., 2025). Unlike autoregressive models, dLLMs generate text via an iterative mask-and-unmask process, allowing them to decode tokens in essentially arbitrary order (Kim et al., 2025). This fundamental change in the generation mechanism during training grants dLLMs remarkable flexibility at inference time. In fact, recent dLLMs have already demonstrated competitive (and sometimes superior) performance compared to their autoregressive counterparts on a similar scale (Zhao et al., 2025). These early successes indicate that the masking strategy during inference plays a crucial role.

Gaps in our understanding about dLLMs. The freedom to choose the generation order, *the masking strategy*, is the central advantage of dLLMs. Recent works (Kim et al., 2025; Nie et al., 2025) have tried to harness this flexibility by relying on prediction confidence, progressively unmasking high-confidence tokens (top-K margin in Figure 1). However, such an approach often leads to inherently biased solutions as they overlook the crucial sequential structure present in the language training data, which induces implicit biases in the learned masking strategies. As a result, these methods might perform worse than random unmasking, as shown in Figure 1. Why this happens and what we can do to avoid such a failure remains an open question, which we address in this work.

Our key finding: hidden semi-autoregressive experts. We uncover a new dimension of test-time scaling for diffusion LLMs, centered on the masking strategy. We observe that the central degree of freedom during inference lies in choosing the *masking strategy*, which is unique to dLLMs and critically shapes the generation distribution. The sequential nature of language data causes dLLMs to implicitly learn a mixture of semi-autoregressive experts during training. Each of these ”experts” is naturally biased towards distinct masking distributions at test time, with a natural preference

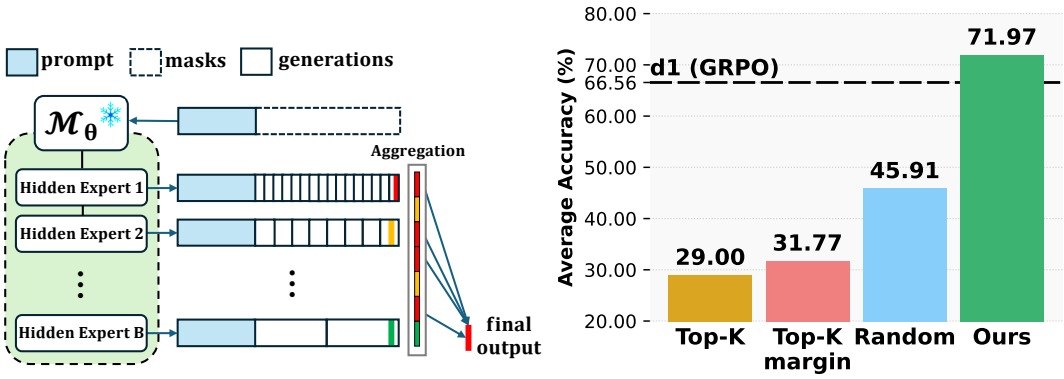

Figure 1: Overview of our proposed HEX framework. **Left**: HEX leverages multiple semi-autoregressive hidden experts, guided by different masking schedules, to produce concatenated outputs and a final answer. **Right**: HEX outperforms Top-K, Top-K margin (Kim et al., 2025) and Random expert selection strategies (Nie et al., 2025) on reasoning tasks (GSM8K, MATH, ARC-C), surpassing the training-based GRPO baseline (d1) (Zhao et al., 2025).

toward semi-autoregressive generation. For the first time, we demonstrate that this latent mixture can be deliberately accessed during inference. By varying the block size used in semi-autoregressive decoding, we can activate different experts, mirroring the conditions the model saw during training. This insight unlocks a novel method for test-time scaling. By marginalizing across these block schedules, we can exploit the latent ensemble of experts, resulting in significantly more robust and optimal inference.

Hence, we propose HEX (Hidden semi-autoregressive EXperts), a training-free inference method that uncovers a new dimension of test-time scaling for dLLMs. HEX marginalizes across block schedules, treating block size and order as latent variables that define an additional scaling dimension, and aggregates predictions via majority voting. In doing so, it robustly avoids the pitfalls of committing to any single decoding path, turning dLLMs' hidden flexibility into a principled mechanism for test-time scaling. We summarize our contributions as follows.

(i) New dimension of test-time scaling in dLLMs. We show that dLLMs implicitly learn a mixture of semi-autoregressive experts, and that block scheduling helps to uncover this latent structure. (Section 3).

(ii) HEX: Hidden semi-autoregressive EXperts for test-time scaling. We introduce HEX, a training-free inference algorithm ensembling over semi-autoregressive schedules with majority-vote aggregation (Algorithm 2), turning ordering into a reliable test-time scaling dimension. (Section 4).

(iii) Comprehensive experimental analysis: matching GRPO-level performance. HEX achieves GRPO-level results on GSM8K, MATH, ARC-C, and TruthfulQA, without retraining, establishing test-time scaling as a powerful new paradigm for diffusion LLMs. HEX outperforms existing state-of-the-art inference methods (Kim et al., 2025) on reasoning tasks, boosting accuracy by up to 3.56× (from 24.72% to 88.10%.) HEX even produces massive gains on more challenging tasks, including MATH (Lightman et al., 2023) (from 16.40% to 40.00%), ARC-C (Clark et al., 2018) (from 54.18% to 87.80%), and TruthfulQA (Lin et al., 2021) (from 28.36% to 57.46%). (Section 5).

1.1 Related Work

Diffusion Large Language Models. Diffusion models have achieved state-of-the-art performance in image generation (Ho et al., 2020; Song et al., 2020), and recent advances extend them to the discrete domain of language. The early approaches applied continuous diffusion to latent text representations (Austin et al., 2021; Li et al., 2022; Dieleman et al., 2022), but faced challenges with scalability and discretization. A masked diffusion paradigm soon emerged as a more tractable discrete alternative (Nie et al., 2024), with large-scale implementations such as DiffuLLaMA (Gong et al., 2024) and LLaDA (Nie et al., 2025) demonstrating that diffusion LLMs (dLLMs) can rival

similarly sized autoregressive models, even on complex reasoning (Zhao et al., 2025; Tang et al., 2025). This potential extends even to multimodal understanding (You et al., 2025; Wen et al., 2025).

Inference-Time Methods for dLLMs. In autoregressive models, inference-time scaling has been extensively studied, ranging from chain-of-thought prompting (Wei et al., 2022) and self-consistency (Wang et al., 2022) to scaling the allocation of test-time compute (Snell et al., 2024). In contrast, inference-time methods for dLLMs remain sparse. Most gains in dLLM performance so far have come from training-time improvements, such as applying GRPO (Zhao et al., 2025; Tang et al., 2025) or post-training method Temporal Consistency Reinforcement (Wang et al., 2025).

2 Problem Formulation

Masked Diffusion Language Models (MDM). Let $x = (x_1, \ldots, x_n) \in \mathcal{V}^n$ be a length-n token sequence over vocabulary $\mathcal{V}$. A masked diffusion large language model (dLLM) specifies a conditional denoiser $p_\theta(x[M] \mid x[M^c])$ for any mask $M \subseteq [n]$, where $M^c = [n] \setminus M$. The notation $x[M]$ is defined as the sequence of tokens from x on the indices of M, i.e. $x[M] = (x_i)_{i \in M}$. In the forward (corruption) process, a random subset of tokens $M \subseteq [n]$ is masked (replaced by a special symbol [MASK], and model p_θ is tasked with recovering the original tokens in M given the unmasked tokens in the complement $M^c := [n] \setminus M$. Formally, for a random mask pattern M, the model produces a conditional distribution $p_\theta(x[M] \mid x[M^c])$ on the masked tokens. Here, $x[M]$ denotes the masked tokens in x. The training objective is to maximize the likelihood of ground-truth tokens in these masked positions. Hence, the training problem can be written as

$$\begin{aligned} \theta^* \in \arg\min_\theta \mathcal{L}_{\text{mask}}(\theta) &:= \mathbb{E}_{x\sim\mathcal{D}} \mathbb{E}_{\ell\sim\text{Unif}([n])} \mathbb{E}_{M\subseteq[n],|M|=\ell} \Big[-\sum_{i\in M} \log p_\theta(x_i \mid x[M^c]) \Big] \\ &= \Big[-\sum_{M\subseteq[n],\, i\in M} \frac{1}{|M|} \frac{1}{\binom{n}{|M|}} \mathbb{E}_{x\sim\mathcal{D}} \big[\log p_\theta(x_i \mid x[M^c]) \big] \end{aligned} \tag{1}$$

where $\mathcal{D}$ is the data distribution, $\ell \sim \text{Unif}([n])$ is a uniformly sampled number of masked tokens $\ell \in \{1, \ldots, n\}$ and $M \subseteq [n]$ is the randomly selected subset of length $|M| = \ell$. The summation in (1) runs over all masked token positions $i \in M$, and the loss on each such position is the negative log-likelihood of the true token x_i given the remaining context (unmasked) $x[M^c]$. The objective in (1) trains the model to predict randomly masked-out tokens, and can be viewed as averaging next-token losses over all token permutations, i.e., an any-order objective (Nie et al., 2025).

Algorithm 1 Vanilla MDM Inference

Input: prompt x_{prompt}, output length L, steps T; mask schedule $\{M_t\}_{t=1}^T$, model $p_\theta(\cdot|\cdot)$.

Initialize: $x^{(0)} \leftarrow [\text{MASK}]^{\times L}$;
for $t = 1, 2, \ldots, T$ **do**
  Predict all masked tokens simultaneously via $\sim p_\theta\big(\cdot \mid [x_{\text{prompt}}, x^{(t-1)}]\big)$
  $x^{(t)} \leftarrow$ Fill with predicted tokens
  Fix tokens at location $i \in M_t^c$
  Mask tokens at location $i \in M_t \setminus \big(\cup_{k=1}^{t-1} M_k^c\big)$

Output: x^T

Inference as the Core Challenge. After learning θ^* from (1), generation happens step by step. For instance, for a given prompt x_{prompt}, it starts by selecting the number of tokens to be generated (say L), and then requires choosing *how* to reveal tokens. Let a *decoding trajectory* be a sequence of masks $\tau = (M_1, \ldots, M_T)$ that partition $[n]$ (such that $\bigsqcup_{t=1}^T M_t^c = [n]$), with per-step sizes $\ell_t = |M_t|$. At step t, the model predicts all masked tokens conditioned on the currently revealed context $x\big[\cup_{s=1}^{t-1} M_s^c\big]$. For a fixed trajectory τ and prompt x_{prompt}, a functional of natural log-likelihood is

$$\mathcal{J}(\tau; \theta \mid x_{\text{prompt}}) = \sum_{t=1}^{T} \sum_{i\in M_t} \log p_{\theta^*}\left(x_i \mid x_{\text{prompt}}, x\Big[\bigcup_{s=1}^{t-1} M_s^c \Big] \right). \tag{2}$$

We summarize the vanilla inference procedure in Algorithm 1. The ideal (but empirically intractable[1]) goal is to choose τ such that we obtain a sample which maximizes $\mathcal{J}(\tau; \theta \mid x_{\text{prompt}})$.

[1]Because simply making consecutive greedy choices does not guarantee that the overall J is maximized, and performing a global search would require evaluating all possible trajectories, which is practically infeasible.

Because (1) trains on *all* mask patterns, many training subproblems are intrinsically ill-posed (e.g., extremely large masks with scant context) due to the implicit sequential bias in the language training distribution. For example, some conditionals are rarely observed or provide little meaningful context, making them effectively unsolvable. As a consequence, the model ends up learning only a subset of subproblems or conditionals, while others remain poorly learned or ignored. A uniform masking objective forces the model to put equal weight on every subproblem, including those that confound the masking sequence, resulting in a suboptimal masking strategy overall. This mismatch creates a gap at inference time: the model's behavior becomes highly sensitive to the masking schedule, and strong performance depends on selecting strategies that align with the sequential biases implicitly learned during training.

Key Open Question. Thus, the central question becomes: *how can we design an inference strategy that faithfully reflects what the model has learned during training, given that dLLMs leave the process fundamentally under-specified and require us to decide the optimal masking trajectory.*

3 Limitations of SoTA and Our Key Insight

Failure of existing inference methods for reasoning tasks. The existing dLLMs rely on heuristic inference-time strategies to choose which tokens to unmask at each denoising step. The common approaches include random sampling (randomly picking masked positions to predict) and confidence-based selection (choosing the token positions with the high model confidence or probability). For example, Kim et al. (2025) showed that for Sudoku puzzles a simple confidence-based top-K margin method can boost accuracy from 7% to nearly 90%. Intuitively, one might expect a similar benefit for reasoning tasks. Surprisingly, we find the opposite in reasoning benchmarks. In our experiments (see Figure 2) on GSM8K math problems, random unmasking far outperforms top-K margin (high-confidence) decoding. Instead of guiding generation, high confidence derails the unmasking trajectory into producing degenerate outputs. In Figure 2, the top-K margin strategy consistently unmasked the [AfterEoT] (⟨endoftext⟩) token at all positions, proceeding *backward from the end to the front* (See B.5 for a detailed explanation). This leads to degenerate outputs (shown in red text) in Figure 2. This surprising result challenges our intuition built through studying prior work.

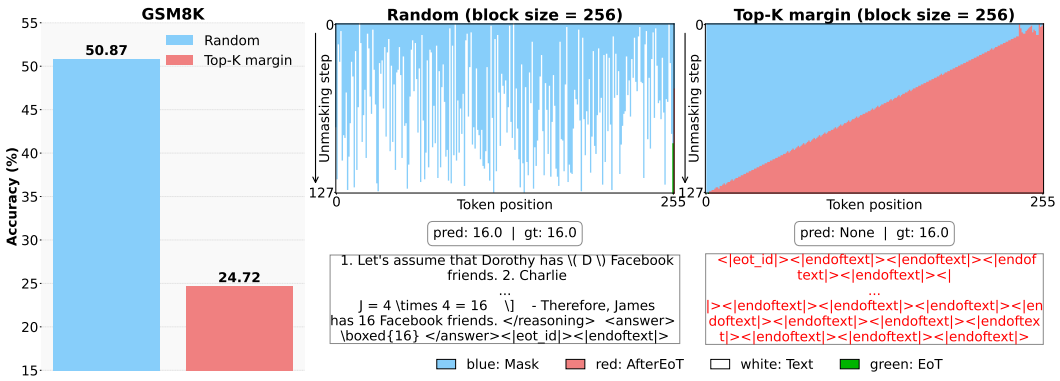

Figure 2: **Random vs. Top-K margin inference on GSM8K. Left**: Random decoding achieves 50.87% accuracy, while **Right**: Top-K margin only 24.72%. For each method, the text box shows the result at the last unmasking step. Top-K margin generates output tokens in reverse, from the end toward the beginning, and exhibits a catastrophic collapse in which all tokens are [AfterEoT] (shown in red). Over 55.5% of top-K margin runs suffered this collapse, yielding very low accuracy. These failures cast doubt on methods that rely solely on token confidence.

Unexpected reversal of intuition. Our findings highlight a key limitation of relying on common heuristics from autoregressive models and prior work (Kim et al., 2025). While one would expect that "follow the model's own highest-probability tokens" is a reliable strategy, our results show that methods relying solely on token confidence are not sufficient for strong performance in complex reasoning tasks. Our results thus raise basic questions: Why does random sampling outperform top-K margin? Why does the model overconfidently pick the [AfterEoT] so early? To answer these

questions, we propose a new perspective on the dLLM's internal structure. Our key insight is that the dLLM can be viewed as an implicit mixture of experts, which allows us to mitigate the risk of overconfidently predicting [AfterEoT] tokens too early. By aggregating predictions from experts conditioned on different subsets of tokens, effectively marginalizing over contexts, our approach avoids prematurely committing to high-confidence tokens like the [AfterEoT] token.

From failure to mechanism. The surprising failure of confidence-based schedules suggests that local token probabilities are unreliable under many masked contexts induced at inference. Because the dLLM objective (in (1)) averages over a wide variety of maskings, including ill-posed ones, some conditionals are poorly learned and disproportionately favor special tokens such as [AfterEoT]. Our view is to treat each semi-AR block schedule as querying a different conditional expert, then marginalize across experts to recover robust predictions. This replaces brittle confidence-following with consensus-seeking and is the core rationale behind HEX.

Our Key Insight: dLLM is an implicit mixture of experts. From (2), it is evident that the diffusion LLM training leads to a model with a family of masked–token conditionals

$$\left\{\, p_\theta(x_i \mid [x_{\text{prompt}}, x[U]]) : i \in [n],\ U \subseteq [n] \setminus \{i\} \,\right\},$$

which we can view as implicit "experts" indexed by the visible/unmasked set of tokens U. For a fixed target index i at test-time, the goal is to aggregate the relevant experts $\{p_\theta(x_i \mid [x_{\text{prompt}}, x[U]])\}_{U \subseteq [n]\setminus\{i\}}$ into a single prediction rule. A principled aggregation strategy is the mixture-of-experts predictor given by

$$p_{\text{mix}}(x_i = a \mid x_{\text{prompt}}) \;=\; \sum_U \pi(U \mid x_{\text{prompt}})\, p_\theta(x_i = a \mid [x_{\text{prompt}}, x[U]]), \tag{3}$$

where $\pi(U \mid x_{\text{prompt}})$ denotes the weight or trust placed on expert U for prompt x_{prompt}.

A Toy Example. In our toy example (Figure 3), we examine the model's answer to the prompt "Who invented telephone?" (ground truth: "The inventor was Bell."). We treat each latent conditional context (or expert) as a separate setting that produces a distribution to predict masked tokens. Figure 3 plots these distributions for a particular position, $i = 4$ (the token 'Bell'). As the plot makes clear, most experts concentrate probability on "Bell" (the correct token), but a few contexts produce flat or incorrect distributions that never predict "Bell".

• *Correct-Answer Contexts (Experts)*: The majority of conditionals yield a distribution that peaked at the correct answer token Bell. These contexts effectively act as "experts" on this question.

• *Non-Expert Contexts*: Some conditionals do not produce a clear peak in 'Bell'. These contexts fail to predict the correct answer.

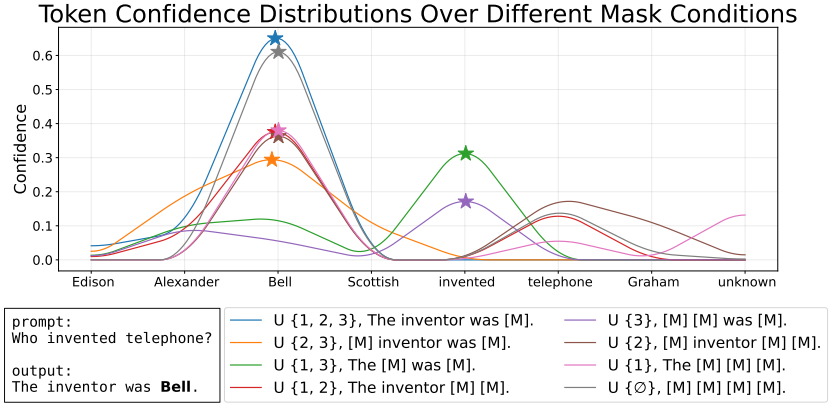

Figure 3: The distribution of the 4th token `'Bell'` in the output sequence changes significantly depending on the 2^3 masking conditions applied to the previous three tokens: `'The'`, `'inventor'`, `'was'`. The star mark indicates the highest confidence for each distribution generated by U. Some masking conditions (violet and green) produce collapsed distribution: `"Bell Bell was invented."` (ungrammatical sentence), `"The telephone was invented."` (missing target information), respectively.

We note that even though the model as a whole can answer correctly, not all conditionals are experts. The figure shows that only a subset of experts "know" the answer, while others do not. Hence, our toy example shows that identifying which context is the right expert is difficult. We do not know a priori which conditional context (or expert) will yield the correct token. The hidden gating distribution $\pi(U)$ that governs how likely each context knows the answer is unknown. The dLLMs doesn't learn the underlying gating distribution.

4 Hidden semi-autoregressive EXperts for test-time scaling

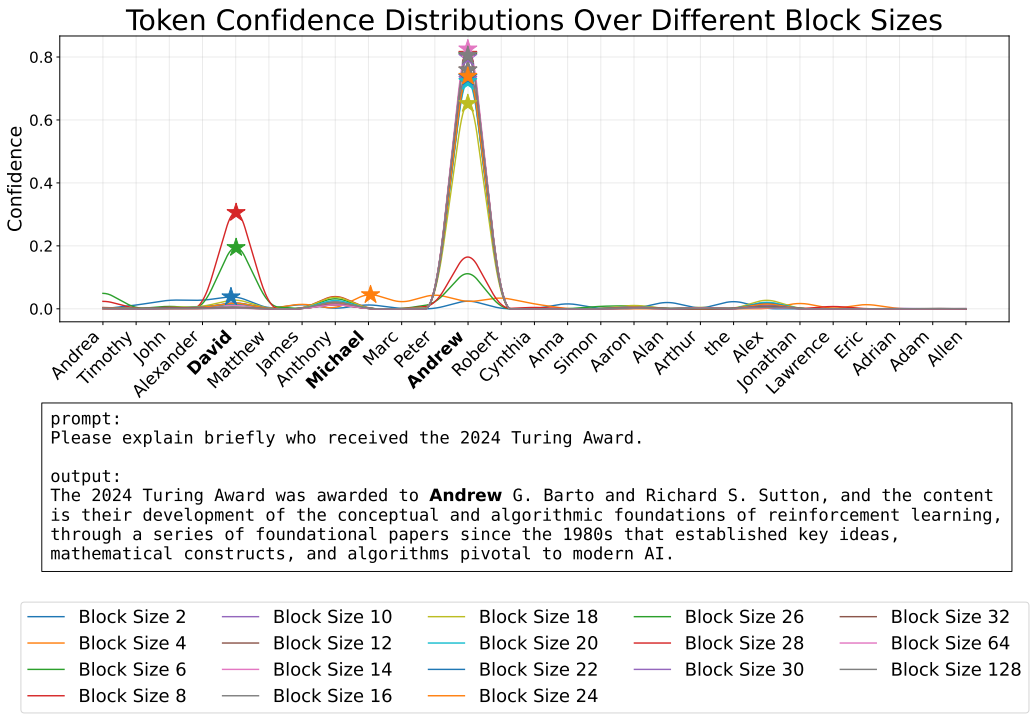

Figure 4: When asked about the 2024 Turing Award winners, names other than the actual recipients (such as Michael or David) might be generated due to different block sizes, which in turn risks producing incorrect information in the subsequent token sequence. However, if we generate outputs with various block sizes and then select the most frequently produced answer, that answer is more likely to be correct, since it was probably derived through a valid reasoning (Andrew) during the inference process.

However, in order to estimate p_{mix}, we would need to estimate the likelihood for dLLM, which is extremely challenging, as also highlighted in (Tang et al., 2025; Zhao et al., 2025; Nie et al., 2025). Ideally, we would compute the Bayes-optimal conditional probability $p_{\text{mix}}(x_i = a \mid x_{\text{prompt}})$ by fully marginalizing over all possible sequences of tokens in U. In practice, this is infeasible for diffusion LLMs, since their likelihood is intractable and must be approximated. To sidestep this, we approximate the ideal mixture by ensemble-averaging over a small set B of "semi-autoregressive," each defined by a particular block-schedule $b \in B$. Concretely, we sample several semi-AR decoding schedules b, each of which queries exactly one expert U_b, and then averaging:

$$p_{\text{mix}}(x_i = a \mid x_{\text{prompt}}) \;\approx\; \mathbb{E}_{b\sim B}\big[p_\theta(x_i = a \mid [x_{\text{prompt}}, x[U_b]])\big]. \tag{4}$$

The final prediction can be made using the Bayes rule:

$$\hat{a} = \arg\max_a \; p_{\text{mix}}(x_i = a \mid x_{\text{prompt}}). \tag{5}$$

A simple Monte Carlo approximation of this decision rule is *majority vote*: draw one sample $\hat{a}_b$ from each queried expert U_b, and return the mode of the sampled values. Thus, either by mixture averaging or majority vote, test-time aggregation amplifies the correct prediction by combining the specific conditionals the model actually learned. Based on this insight, we summarize our proposed approach in Algorithm 2.

For each schedule, we first convert the final generated token sequence $x^{(T)}$ into a natural language string representation. We then apply numeric parsing to remove LaTeX formatting, whitespace, and commas. This yields a parsed output $f(x^{(T)})$ for each schedule, which we store in a list. The final answer is chosen as the value that occurs most frequently across schedules. If there is a tie (i.e., two or more values appear with the same highest frequency), we choose the value generated from the schedule with the smallest block size.

Algorithm 2 Hidden semi-autoregressive **EX**perts for test-time scaling

Input: prompt x_{prompt}, output length L, output parser f, steps T, experts B; semi-AR mask schedule $\{\{U_{t,b}\}_{t=1}^{T}\}_{b=1}^{B}$, model $p_\theta(\cdot|\cdot)$.

Initialize `outputs` $\leftarrow$ [ ]
for $b = 1, 2, \ldots, B$ **do**
  Initialize: $x^{(0)} \leftarrow [\text{MASK}]^{\times L}$
  for $t = 1, 2, \ldots, T$ **do**
    Predict all masked tokens simultaneously via $\sim p_\theta\big(\cdot \mid [x_{\text{prompt}}, x^{(t-1)}]\big)$
    $x^{(t)} \leftarrow$ Fill with predicted tokens
    Fix tokens at location $i \in U_{t,b}^{c}$
    Mask again tokens at location $i \in U_{t,b} \setminus \left(\cup_{k=1}^{t-1} U_{k,b}^{c}\right)$
  $x^{(T)} \leftarrow x^{(T)}$ as a natural language string
  Append parsed string output $f(x^{(T)})$ to `outputs`
Output: max(`outputs`, key = lambda x: (`outputs`.count(x), -`outputs`.index(x))

Why semi-autoregressive? Diffusion LLMs allow all trajectories to reveal masked tokens, but uniformly random orders are suboptimal for language: they create unnatural partial contexts that the model was never intended to generate at test time. A practical restriction is semi-autoregressive left-to-right decoding (semi-AR): fix a block size $b \in \{1, \ldots, B_{\max}\}$ (where $B_{\max} = n$) and partition $[n]$ into consecutive blocks

$$M_t = \{(t-1)b + 1, \ldots, \min(tb, n)\},$$

for $t = 1, \ldots, T(b) = \lceil n/b \rceil$, revealing blocks left-to-right while denoising within each block via diffusion. This preserves a strong prefix structure (natural for language), yet allows parallel denoising inside a block. Empirically, we find (see Table 1) that semi-AR decoding avoids pathologies seen in fully parallel decoding. In particular, when using a single large block (i.e., non-semi-AR parallel decode), we often observe an [AfterEoT] collapse: the model erroneously floods the tail with [AfterEoT] tokens or repeats (Figure 8). By contrast, constraining to moderate block sizes (decoding left-to-right) eliminates this collapse and dramatically improves accuracy. (See Table 1: semi-AR has 0% collapse and much higher accuracy than non-AR decoding.) Intuitively, focusing first on the left part of the output prevents the model from prematurely committing to a length or drifting with high-confidence tail tokens.

Table 1: Semi-AR based decoding eliminates [AfterEoT] collapse and improves accuracy.

Dataset	**Collapsed** (↓, %)	**Accuracy** (↑, %)
Baseline (non–semi-AR)		
GSM8K	55.80	22.52
MATH	29.80	16.60
Semi-AR		
GSM8K	**0.00**	**76.27**
MATH	**0.00**	**32.80**

5 Experiments

In this section, we empirically validate our claims. (i) *Effectiveness*: We first demonstrate that HEX significantly outperforms existing training-free and fine-tuned methods on a suite of reasoning benchmarks. (ii) *Scaling behavior:* We then analyze the performance-computation trade-off, showing how accuracy scales with more diverse generation paths. (iii) *Working mechanism:* Finally,

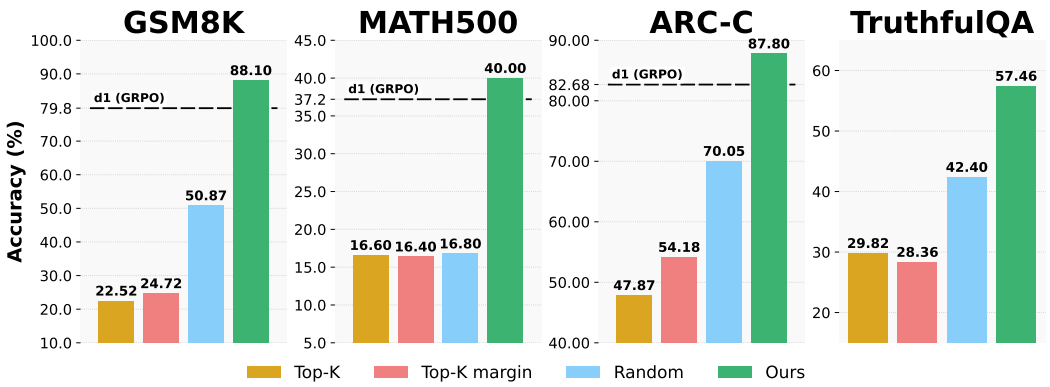

Figure 5: **HEX improves reasoning accuracy.** On LLaDA-8B-Instruct, HEX outperforms training-free baselines (Random, Top-k, Top-k-margin) on GSM8K, MATH, ARC-C, and TruthfulQA. In GSM8K, MATH, ARC-C, it even outperforms the model trained with GRPO without any training.

through a series of ablations and qualitative examples, we explore the mechanisms behind HEX's success, confirming that its gain comes from ensembling a latent mixture of semi-AR experts rather than relying on heuristics like model confidence.

5.1 Setup

Datasets and Metrics. We follow standard reasoning benchmarks: **GSM8K** (Cobbe et al., 2021) consisting of high-quality problems with diverse linguistic expressions, **MATH** (Lightman et al., 2023) is a more challenging math benchmark that includes competition-level math problems, **ARC-C** (Clark et al., 2018) is the Challenge Set from AI2's ARC dataset, consisting of science knowledge-based questions that are difficult to solve with simple keyword matching or retrieval, and **TruthfulQA** (Lin et al., 2021) which evaluates the tendency of language models to generate false information by following human misconceptions or false beliefs.[2] Primary metric is task accuracy.

Models and Baselines. All experiments with inference methods were performed using the LLaDA-8B-Instruct model (Nie et al., 2025), and the application of d1 (GPRO) (Zhao et al., 2025) is subsequently based on this model. For all methods, when the output length is 256 tokens, the number of unmasking steps is 128. At each step, two masked tokens are unmasked, and this process is repeated until all tokens are revealed. *Random* unmasks two randomly chosen masked tokens per step. Top-k margin unmasks, at each step, the two masked tokens with the highest margin defined as $(\text{top-1 confidence} - \text{top-2 confidence})$ at their positions. d1 (GRPO) row uses the reported best value (Zhao et al., 2025) for GSM8K and MATH, and for ARC-C we report a value reproduced after 1 epoch of training. TruthfulQA trained on d1 (GRPO) is excluded because there is no training data available, and neither were checkpoints released. HEX draws five samples at temperature = 0.9 for each of the block sizes $[8, 16, 32, 64, 128]$, yielding 25 samples in total. If a tie occurs for the most frequent value, the value generated with the smallest block size is selected (Algorithm 2).

5.2 Main Results: HEX Establishes a New State-of-the-Art

Overall performance. Figure 5 shows that HEX achieves the strongest results across all four reasoning benchmarks, outperforming both training-free and fine-tuned baselines. Compared to existing decoding strategies (Nie et al., 2025; Kim et al., 2025), HEX delivers large and consistent gains. In GSM8K, for example, HEX reaches 88.10% accuracy, far higher than Random decoding (50.87%) and Top-k margin (24.72%). These results show that confidence-based heuristics are unreliable in diffusion LLMs, whereas consensus-based voting in HEX is robust (Figure 7).

Comparison with GRPO fine-tuned models. Perhaps most strikingly, HEX also surpasses d1 (GRPO), which requires costly reinforcement learning fine-tuning. On GSM8K (88.10% vs.

[2]We use official evaluation scripts; numeric parsing strips LaTeX wrappers/whitespace/commas.

79.80%), MATH (40.00% vs. 37.20%), and ARC-C (87.80% vs. 82.68%), HEX sets a new state of the art without updating model parameters.

Intuitively, fixed inference scheduled in existing techniques sometimes asks the model to guess hard tokens too early, which leads to mistakes. In contrast, HEX tries several semi-autoregressive schedules and then picks the answer that many schedules agree on. In practice, answers that show up across schedules are more reliable than answers from any single schedule.

Takeaway. These results suggest that the reasoning ability of a diffusion LLM remains latent and can be unlocked at inference time through block-marginalized ensembling, without any fine-tuning.

5.3 Analysis of Scaling and Compute Trade-off

Figure 6 shows that HEX's accuracy improves monotonically as the number of voting samples increases, while the tie rate, an indicator of ambiguity, steadily declines. Intuitively, different semi-AR schedules make different mistakes but tend to agree on the correct answer; adding schedules cancels schedule-specific errors and strengthens consensus, so ties resolve and accuracy improves. This trend holds consistently across all four benchmarks. Because sampling more trajectories linearly increases compute cost, HEX effectively exposes a tunable accuracy, compute knob: practitioners can trade inference cost for accuracy in a predictable way, without retraining.

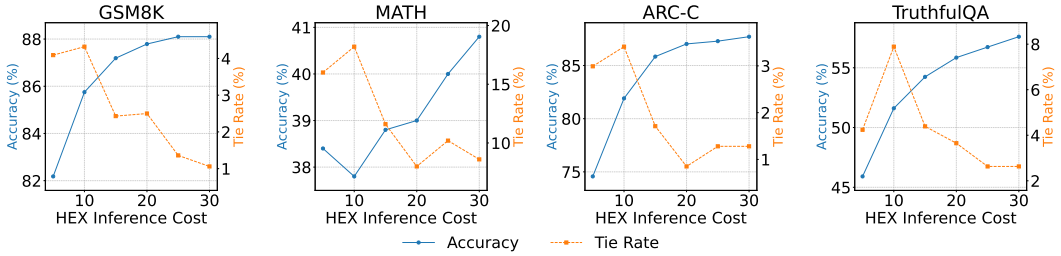

Figure 6: As the number of majority voting samples in HEX increases, accuracy improves and the tie rate decreases. The block sizes used are [8, 16, 32, 64, 128], and sampling was performed while increasing the number of seeds (1-6).

Takeaway. HEX not only establishes state-of-the-art performance but also provides a principled mechanism for test-time scaling, ensuring accuracy improves with more inference budget.

5.4 Ablation Studies

Next, we analyze the mechanisms behind the HEX improvements, focusing on two key factors: the role of block diversity and the role of likelihood versus frequency in candidate selection.

Effect of block diversity. Beyond using a fixed set of block sizes, we test whether ensembling over more varied (and even randomly generated) block schedules further boosts performance. As shown in Table 2, increasing the number of dynamic trajectories from 5 to 30 on GSM8K improves accuracy from 81.96% to 84.15% while reducing the tie rate to less than half. This reinforces our hypothesis that performance gains come from aggregating diverse "semi-AR experts." We note that diversity matters, but structured diversity (fixed block set with multiple seeds) is even stronger (as in Table 3), yielding the highest overall gains.

Size	**Accuracy** (↑ %)	**Tie** (↓ %)
5	81.96	3.87
10	82.34	3.18
15	82.49	1.59
20	82.79	1.59
25	83.47	1.52
30	**84.15**	**1.06**

Table 2: HEX dynamic block size results. Accuracy and tie rate (%) on GSM8K across dynamic block size. See Figure 9 for details.

Frequency vs. likelihood. We then examine whether HEX's gains could simply come from likelihood-based re-ranking. Table 3 shows that the selection of the lowest negative log-likelihood candidate (NLL) performs poorly, in some cases worse than Random decoding (e.g., ARC-C: 70.05% vs. 60.84%). In contrast, HEX's frequency-based majority vote achieves much higher accuracy (74.57%), confirming that

consensus among diverse trajectories is more reliable than model confidence scores. This shows that the key driver of HEX's success is ensemble agreement.

Tie break and Latency. HEX defaults to the smallest block size in tie situations, as Table 4 indicates that jointly considering frequency and log-likelihood does not bring a clear advantage. In addition, we present the wall-time latency of HEX and the baseline inference methods in Table 5.

Table 3: Ablations across datasets. **NLL** selects the candidate with the lowest NLL. HEX's tie issue diminishes as the number of samples increases. Block sizes: [8, 16, 32, 64, 128].

	GSM8K		MATH		ARC-C		TruthfulQA	
Method	Acc (↑%)	Tie (↓%)	Acc (↑%)	Tie (↓%)	Acc (↑%)	Tie (↓%)	Acc (↑%)	Tie (↓%)
Baselines								
Random	50.87	–	16.80	–	70.05	–	42.40	–
top-k	22.52	–	16.60	–	47.87	–	29.82	–
top-k margin	24.72	–	16.40	–	54.18	–	28.36	–
d1 (GRPO)	79.80	–	37.20	–	82.68	–	–	–
Likelihood-based								
NLL	76.72	4.09	34.40	16.00	60.84	2.99	28.07	4.24
HEX								
HEX	82.18	4.09	38.40	16.00	74.57	2.99	45.91	4.24
HEX ×5 seeds	**88.10**	**1.36**	**40.00**	**10.20**	**87.80**	**1.11**	**57.46**	**2.78**

6 Conclusion and Limitation

In this work, we study how diffusion-based language models (dLLMs) generate text. We found that their performance is fundamentally tied to the decoding schedule, the order in which tokens are generated. This is because dLLMs implicitly learn a "set" of semi-autoregressive experts during training. Different schedules activate different experts, and choosing the right one is crucial for getting a high-quality answer. This single insight helps explain common dLLM issues, such as why they sometimes stop generating text too early or fail even when they seem confident. Based on this insight, we introduced HEX (Hidden semi-autoregressive EXperts), a powerful inference method that requires no extra training. Instead of relying on a single schedule, HEX tries many different schedules at once and lets the experts "vote" on the best final answer. By combining the strengths of the entire hidden team, HEX turns the model's flexibility into a reliable tool for boosting performance. On challenging reasoning benchmarks, HEX doesn't just beat standard methods; it even surpasses models fine-tuned with costly techniques like reinforcement learning (GRPO).

HEX has some limitations. It requires more computation at test time, and we have mainly evaluated it on reasoning tasks. Applying this method to more creative areas like open-ended stories, image generation, or long conversations remains a promising area for future work. Further, we have not established any theoretical understanding of HEX, which is a valid scope of future work.

Ethics Statement

Our research aims to reveal the untapped potential of diffusion-based Large Language Model (dLLMs) and to enhance reasoning performance across comprehensive tasks without additional training, through test-time scaling. All datasets used in the evaluation are public and widely known, and to the best of our knowledge, we have thoroughly examined and cited research that is potentially or directly related to our work. We clarify that our use of LLMs was strictly limited to polish writing, such as grammatical correction and fluent expression, not for generating the main content of the research.

References

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022. doi: 10.48550/arXiv.2211.15089.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. *arXiv preprint arXiv:2410.17891*, 2024.

GSAI-ML. LLaDA-8B-Instruct. https://huggingface.co/GSAI-ML/LLaDA-8B-Instruct. MIT License, accessed 2025-09-18.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*, 2025.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35: 4328–4343, 2022.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*, 2024.

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.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. Weighted policy optimization for reasoning in diffusion language models. *arXiv preprint arXiv:2507.08838*, 2025.

Wen Wang, Bozhen Fang, Chenchen Jing, Yongliang Shen, Yangyi Shen, Qiuyu Wang, Hao Ouyang, Hao Chen, and Chunhua Shen. Time is a feature: Exploiting temporal dynamics in diffusion language models. *arXiv preprint arXiv:2508.09138*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yuqing Wen, Hebei Li, Kefan Gu, Yucheng Zhao, Tiancai Wang, and Xiaoyan Sun. Llada-vla: Vision language diffusion action models. *arXiv preprint arXiv:2509.06932*, 2025.

Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*, 2025.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*, 2025.

APPENDIX

A QUALITATIVE RESULTS

A.1 QUALITATIVE ANALYSIS OF BASELINES VS. HEX

Manolo bought five lollipops and four candies that cost $3.20.
If each lollipop costs $0.40, how much will 10 lollipops and 10 candies cost him?
Ground Truth: 7

Top-K margin:
First, let's determine the cost of one candy. We know that five lollipops and four candies cost $3.20, and each lollipop costs $0.40.
The total cost of five lollipops is:
\[5 \times 0.40 = 2.00 \]

This means that the total cost of the four candies is $1.20.
Therefore, the cost of one candy is:
\[\frac{1.20}{4} = 0.00 \]

Now, we need to find the cost of 10 lollipops and 10 candies. The cost of 10 candies is:
\[10 \times 0.00 = 0.00 \]

The cost of 10 lollipops is:
\[10 \times 0.40 = 4.00 \]

Therefore, the total cost for 10 lollipops and 10 candies is $4.00.
</reasoning>

<answer>
\boxed{**4.00**}
</answer><|eot_id|><|endoftext|>

Random:
First, we need to determine the cost of one candy. We know that Manolo bought five lollipops and four candies that cost $3.20, and each lollipop costs $0.40. This can be expressed as:

\[5 \times 0.400 + 4C = 3.20 \]
\[4C = 2.00 \]
\[C = \frac{2.00}{4} = 0.5000 \]

Now, we know each lollipop costs $0.40 / 100 = $0.40. Each candy costs $0.4000.

To find the total cost of 10 lollipops and 10 candies:

\[10 \times 0.4000 + 10 \times 0.5000 \]
\[10 \times 0.400 + 50000 \times 100 = 5.00 \]
</reasoning>

<answer>
\boxed{**5.00**}
</answer><|eot_id|><|endoftext|>

HEX (Ours):
First, let's determine the cost of one candy. We know that Manolo bought five lollipops and four candies for a total of $3.20, and each lollipop costs $0.40, so the total cost of five lollipops is \(5 \times 0.40 = 2.00\).

Therefore, the cost of four candies is \(3.20 - 2.00 = 1.20\).

Since four candies cost $1.20, the cost of one candy is \(1.20 \div 4 = 0.30\).

Now, we need to find the cost of 10 lollipops and ten candies. The cost of 10 lollipops is \(10 \times 0.40 = 4.00\)

and the cost of 10 candies is \(10 \times 0.30 = 3.00\).

So, the total cost is \(4.00 + 3.00 = 7.00\).
</reasoning>

<answer>
\boxed{**7.00**}
</answer><|eot_id|><|endoftext|>

Figure 7: An instance of generated text responses of different decoding strategies.

A.2 QUALITATIVE ANALYSIS OF SEMI-AR VS. NON-SEMI-AR

As shown in the right side of Figure 8 in confidence based non-semi-AR decoding, the phenomenon where [AfterEoT] tokens accumulate from the end of the output towards the front indicates that the model is assigning high confidence to [AfterEoT] token throughout the unmasking steps.

The input to the dLLM consists of the number of tokens that make up the prompt and the number of tokens in the desired output sequence, and during training it is subject to limits on the input sequence length for parallel computation. For LLaDA-8B-Instruct (GSAI-ML), this limit is 4,096 tokens. However, in the training of reasoning tasks, most of the output finishes within 256 tokens. In other words, the majority of *ground truth* tokens in the output sequence (more than 93.75%) are [AfterEoT]: Given that the training objective is to maximize the average likelihood, we can infer that the dLLM is most strongly taught to generate the [AfterEoT] token.

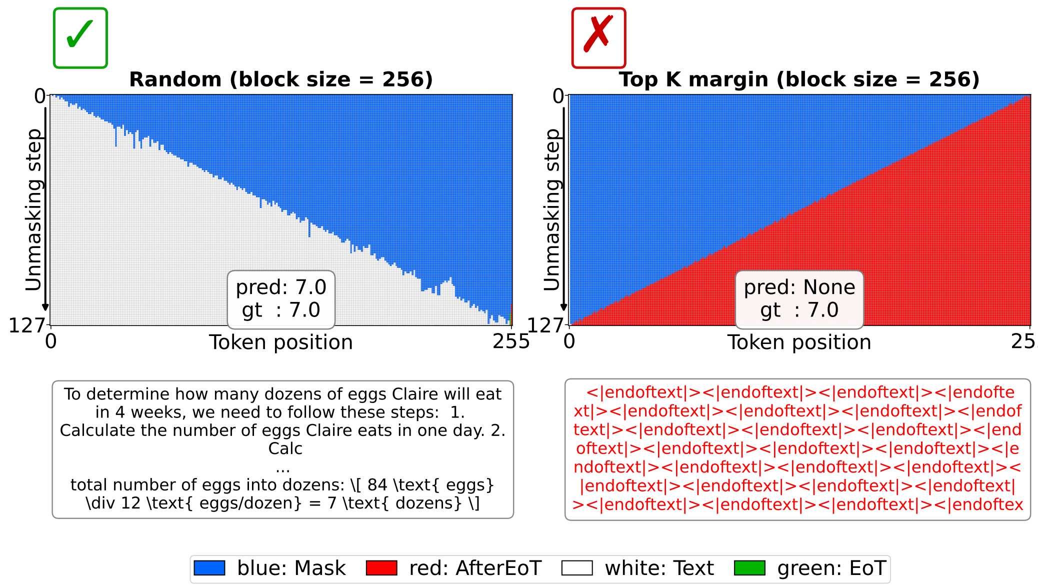

Figure 8: Blue denotes mask tokens, red denotes [AfterEoT] tokens, white denotes text tokens, and green denotes [EoT] tokens (note that in the LLaDA-8-Instruct model, [EoT] and [AfterEoT] are represented as $< |eot\_id| >$ and $< |endoftext| >$, respectively (GSAI-ML)). As unmasking proceeds, two mask tokens are unmasked at each step (output length = 256, unmasking steps = 128). Under a semi-AR regime with block size = 32, positional constraints force reasoning to progress left-to-right while still allowing diffusion-style generation within each block. By contrast, when the positional constraint is removed with block size = 256 (non-semi-AR), the model starts from the last token with the highest confidence—[AfterEoT]—and, due to the inertia of repeatedly generating the same token backward, ultimately collapses into a catastrophic output in which all tokens become [AfterEoT].

This suggests that confidence-only decoding is fundamentally limited in its ability to prevent such phenomena during inference, and highlights why the positions of tokens to be unmasked should not be selected based solely on confidence.

B ADDITIONAL EXAMPLES AND RESULTS WHICH CAN BE USEFUL

B.1 HOW THE SEMI-AR SCHEDULE LEVERAGES LEARNED PREFIX-LIKE CONTEXTS

Let $x = (x_1, x_2, x_3, x_4)$. To predict x_4, the visible set is a subset of $\{1, 2, 3\}$, i.e.

$$U \in \{\varnothing, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}.$$

Suppose (due to sequential bias in the data) the model learns well only the prefix-like contexts

$$\widehat{\mathcal{S}} = \{\{1\},\ \{1, 2\},\ \{1, 2, 3\}\}. \tag{6}$$

Then left-to-right semi-autoregressive (semi-AR) schedules realize exactly these conditionals by changing the block size B:

- $B = 1, 1, 1, 1$ (blocks $\{1\}, \{2\}, \{3\}, \{4\}$): when x_4 is predicted, the visible set is $U = \{1, 2, 3\}$, so the model uses $p_\theta(x_4 \mid x_1, x_2, x_3)$.
- $B = 2, 2$ (blocks $\{1, 2\}, \{3, 4\}$): when x_4 is predicted (with x_3 in the same step parellelly), the visible context is the completed first block, $U = \{1, 2\}$, hence $p_\theta(x_4 \mid x_1, x_2)$.
- $B = 3, 1$ (blocks $\{1, 2, 3\}, \{4\}$): when x_4 is predicted, $U = \{1, 2, 3\}$ again, hence $p_\theta(x_4 \mid x_1, x_2, x_3)$.

One can also realize $U = \{1\}$, by $B = 1, 3$ (blocks $\{1\}, \{2, 3, 4\}$): when x_4 is predicted (with x_2, x_3 in the same step in parallel), the visible context is the completed first block, $U = \{1\}$, hence $p_\theta(x_4 \mid x_1)$. Furthermore, if we set additional *within*-block order constraints[3] C_{order} (e.g. descending order of confidence), another possible condition of $U = \{1\}$ can occur by

$$B = 4 \text{ (blocks } \{1, 2, 3, 4\}) \ \wedge \ C_{\text{order}}\big(x_1 \prec x_4 \prec x_2 \prec x_3 \text{ within } B\big),$$

hence the model uses $p_\theta(x_4 \mid x_1)$.

Key point: By varying the semi-AR block size and *within*-block order, decoding selects among the learned conditionals in $\widehat{\mathcal{S}}$.

B.2 Details of dynamic HEX block settings

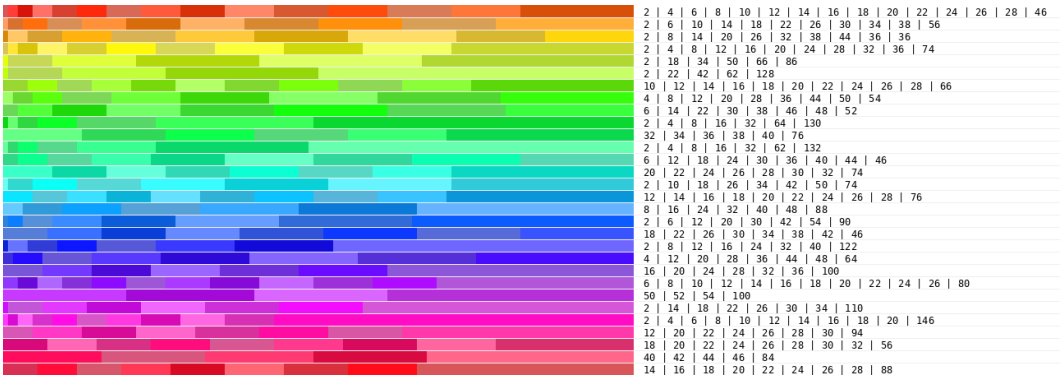

Figure 9: Examples of the block sizes and counts used in the dynamic HEX block settings. Block sizes and counts were randomly chosen and adjusted to match the total output length. The output length is 256 and the number of unmasking steps is 128, meaning that each step unmasks 2 tokens. Accordingly, all block sizes are multiples of 2, and decoding was performed in a semi-autoregressive manner.

B.3 Experimental results of HEX's tie-breaking methods

Table 4: Evaluation on tie breaking methods. If the most frequent output is in a tie situation, **TIED: NLL** selects the result with the lowest negative log-likelihood in tie situations, **TIED: first** selects the result generated from the smallest block size when tied, and **TIED: any** treats the case as correct if a correct option exists among the tied candidates. The results of **TIED: any** clearly highlight that majority voting of HEX works well across datasets.

	GSM8K		MATH		ARC-C		TruthfulQA	
Method	Acc (↑%)	Tie (↓%)	Acc (↑%)	Tie (↓%)	Acc (↑%)	Tie (↓%)	Acc (↑%)	Tie (↓%)
HEX (tie-breaking rules)								
HEX, TIED: NLL	82.18	4.09	38.00	16.00	74.49	2.99	46.20	4.24
HEX, TIED: first	82.18	4.09	38.40	16.00	74.57	2.99	45.91	4.24
HEX, TIED: any	83.09	4.09	41.00	16.00	76.11	2.99	47.66	4.24

[3] Adding this condition changes the number of unmasking steps within the block size.

B.4 Analysis of decoding latency across inference methods

Table 5: Inference efficiency (in seconds) of HEX on GSM8K, MATH, ARC-C, TruthfulQA. The numbers in parentheses indicate the number of data points. Random, top-k, and top-k margin use a single sample with a block size of 32. HEX uses five samples, where each sample is generated with block sizes of 8, 16, 32, 64, and 128. Across all samples, the output length is set to 256, with 2 tokens being unmasked at each step.

Dataset	Method	Total test set	per batch (8)	per datapoint (1)	ratio
GSM8K (1319)	random	2775.73	16.76	2.09	×1.0000
	top-k	2921.70	17.64	2.20	×1.0526
	top-k margin	3187.56	19.25	2.41	×1.1484
	HEX	14613.72	88.23	11.03	×5.2648
MATH (500)	random	1300.62	20.46	2.56	×1.0000
	top-k	1365.98	21.51	2.69	×1.0503
	top-k margin	1477.33	23.28	2.91	×1.1359
	HEX	6823.17	107.50	13.43	×5.2461
ARC-C (1172)	random	2679.99	18.15	2.27	×1.0000
	top-k	2813.69	19.05	2.38	×1.0499
	top-k margin	3048.49	20.64	2.58	×1.1375
	HEX	14062.59	95.22	11.90	×5.2473
TruthfulQA (684)	random	1532.40	17.71	2.21	×1.0000
	top-k	1608.77	18.58	2.32	×1.0498
	top-k margin	1739.78	20.12	2.52	×1.1353
	HEX	8038.26	92.90	11.61	×5.2455

B.5 Structured Patterns Inherent in AfterEoT Collapse

Despite being highly unintuitive and unpredictable, the ordering pattern observed in AfterEoT Collapse exhibits a clear structure: although there is no explicit incentive for the model to unmask padding tokens in a *right-to-left* manner during either training or inference, this regularity consistently emerges throughout the entire unmasking process (see Figure 2). This systematic behavior under collapse suggests that the underlying ordering mechanisms in diffusion LLMs — embedded within both the training objective and the inference procedure — may play a more active and influential role than previously recognized.