

Parallelism and Generation Order in Masked Diffusion Language Models: Limits Today, Potential Tomorrow

Yangyang Zhong^{1,2}, Yanmei Gu², Zhengqing Zang^{1,2}, Xiaomeng Li²,
Yuqi Ding^{2,3}, Xibei Jia^{1,2}, Yuting Shen^{2,4}, Zhenzhong Lan^{2,5}, Liwang Zhu²,
Weiping Liu², Junlin Zhou², Haisheng Liu², Zhong Xin Yu², Pengxin Luo¹,
Donglian Qi¹, Yunfeng Yan^{1†}, Junbo Zhao^{2,1†}

¹Zhejiang University, ²Ant Group, ⁴Shanghai Jiao Tong University
³University of Chinese Academy of Social Sciences, ⁵Westlake University
{yyff, j.zhao}@zju.edu.cn

Abstract

Masked Diffusion Language Models (MDLMs) promise parallel token generation and arbitrary-order decoding, yet it remains unclear to what extent current models truly realize these capabilities. We characterize MDLM behavior along two dimensions—parallelism strength and generation order—using Average Finalization Parallelism (AFP) and Kendall’s τ . We evaluate eight mainstream MDLMs (up to 100B parameters) on 58 benchmarks spanning knowledge, reasoning, and programming. The results show that MDLMs still lag behind comparably sized autoregressive models, mainly because parallel probabilistic modeling weakens inter-token dependencies. Meanwhile, MDLMs exhibit adaptive decoding behavior: their parallelism and generation order vary significantly with the task domain, the stage of reasoning, and whether the output is correct. On tasks that require “backward information” (e.g., Sudoku), MDLMs adopt a solution order that tends to fill easier Sudoku blanks first, highlighting their advantages. Finally, we provide theoretical motivation and design insights supporting a *Generate-then-Edit* paradigm, which mitigates dependency loss while retaining the efficiency of parallel decoding.

1 Introduction

Autoregressive (AR) language models dominate modern natural language processing (NLP) due to their strong likelihood-based training objectives and reliable left-to-right decoding. However, the strictly sequential nature of AR decoding entails two fundamental limitations: (i) high inference latency and constrained generation throughput; and (ii) for tasks requiring global constraints or non-monotonic planning, a fixed chronological order may not constitute the most natural solution path. As an important branch of discrete diffusion models, masked diffusion language models (MDLMs) address these gaps by iteratively denoising masked

sequences, enabling parallel token prediction within a single step and, in principle, allowing for more flexible, non-monotonic generation orders (Li et al., 2025; Sahoo et al., 2024). For KV-cache reuse and efficiency, most practical systems adopt a block-diffusion architecture—executing autoregressively across blocks while employing diffusion within each block (Arriola et al., 2025). Recent high-performing MDLMs (e.g., LLaDA 2.0 (Bie et al., 2025), Trado (Wang et al., 2025), DiRL (Zhu et al., 2025d), SDAR (Cheng et al., 2025), and OpenPangu-Diffusion (Tian et al., 2025)) are typically initialized from the architecture and weights of strong AR models and then transferred to the diffusion paradigm by modifying the decoding mechanism, achieving competitive results.

Despite rapid progress, it remains unclear whether MDLMs truly exploit their latent non-monotonic potential during inference, or how they navigate the trade-off between parallelism and generation quality. Existing studies primarily focus on inference strategies or parameter tuning, reporting their impact on a limited set of benchmarks (Wu et al., 2025; Ye et al., 2024; Kang et al., 2025; Feng et al., 2025), yet a systematic characterization of the model’s decoding trajectories and dynamic behaviors is missing. While some recent works have begun examining decoding order (Chen et al., 2025; Gong et al., 2025), they often fail to explicitly decouple “generation order” from “parallelism intensity”; for instance, (Gong et al., 2025) conflates order metrics with variations in parallelism. Furthermore, speed evaluations frequently rely on hardware-dependent metrics such as throughput (tokens/s), which are sensitive to implementation details and hinder fair cross-model comparisons. Consequently, the community still lacks unified, interpretable, and hardware-agnostic metrics to disentangle the parallelism and non-monotonicity of MDLMs, limiting our understanding of their quality degradation and potential strengths.

Motivated by these gaps, we treat **parallelism** and **generation order** as two defining degrees of freedom of MDLMs and conduct a large-scale reality check via mechanism-level analyses. Our contributions are four-fold: **(1) Large-scale Evaluation:** We unify the deployment of recent MDLMs (e.g., LLaDA 2.0) and strong AR baselines on **58 benchmarks** spanning Knowledge, Math, Reasoning, and Coding. We identify a consistent accuracy gap and provide a theoretical account through the lens of *parallel factorization*: the conditional-independence approximation in parallel decoding induces an unavoidable lower bound on quality. **(2) Parallelism and Order Metrics:** To operationalize these properties, we propose **Average Finalization Parallelism (AFP)** and adopt Kendall’s τ to quantify the alignment between token finalization and surface order. Analyzing a 100B-scale MDLM, we reveal an **adaptive trade-off** where the model accelerates on structure-heavy spans but decelerates at semantic pivots. Notably, correct predictions demonstrate higher parallelism, suggesting a *scaling dividend* where enhanced model capability naturally accelerates inference. **(3) Uncovering Non-monotonic Potential:** Through Sudoku-variant benchmarks, we amplify the advantages of MDLMs in parallel and non-monotonic solving, exhibiting solution paths qualitatively distinct from AR decoding and suggesting significant any-order probabilistic modeling potential. **(4) Mitigating Parallel Factorization Loss:** We provide a theoretical perspective suggesting that a two-stage generate-then-edit paradigm could mitigate the dependency loss induced by parallel factorization.

2 Related Work

2.1 Masked Diffusion Language Models

Masked Diffusion Language Models (MDLMs) are a representative class of discrete diffusion generative models (Hoogeboom et al., 2021; Austin et al., 2021a). In the forward process, noise is injected by progressively replacing clean tokens with a special mask symbol [MASK] in a discrete space. The reverse process then iteratively denoises a masked sequence to recover the original text. Recently, MDLMs have been scaled to the regime of large pretrained language models. On the one hand, several works study *native* training of discrete diffusion language models; for example, the LLaDA 1.0 line (Nie et al., 2025; Zhu et al., 2025a) introduces diffusion-oriented masking strategies and decoding

schedules. On the other hand, a growing trend adapts strong autoregressive (AR) models to the diffusion paradigm by modifying their causal decoding while largely preserving pretrained weights (e.g., LLaDA-MoE (Zhu et al., 2025b), Dream-7B (Ye et al., 2025), LLaDA 2.0 (Bie et al., 2025), SDAR (Cheng et al., 2025), Trado (Wang et al., 2025), DiRL (Zhu et al., 2025d), and OpenPangu-Diffusion (Tian et al., 2025)). In our study, we further unify the deployment of these recent SOTA models and conduct an in-depth analysis.

2.2 Analyses of Diffusion LM Mechanisms

Prior work has analyzed dLLM decoding and parallelization. (Wu et al., 2025) link quality drops in parallel decoding to the conditional-independence assumption and propose threshold-/factor-based unmasking, but evaluate on limited leaderboards and use hardware-dependent throughput (tokens/s), which hinders fair cross-setting comparison. (Gong et al., 2025) study *decoding order* via AR-ness, yet it depends on a hyperparameter k and is sensitive to decoding setups; moreover, decoding order is often conflated with *parallelism* (tokens per step). (Ye et al., 2024) highlight dLLM benefits for reasoning/planning, but provide limited instance-level decoding dynamics. In contrast, we study 58 benchmarks, explicitly separate decoding order from parallelism, use robust order statistics, and measure parallelism by average tokens per step to avoid hardware confounds.

3 Preliminaries

In this section, we review inference in autoregressive and masked diffusion language models.

Auto-regressive Modeling. Let $x_{1:T} = (x_1, \dots, x_T)$ be a token sequence. Autoregressive language models factorize its probability as

$$p_{\text{AR}}(x_{1:T}) = \prod_{t=1}^T p_{\theta}(x_t | x_{<t}), \quad (1)$$

where $x_{<t} = (x_1, \dots, x_{t-1})$ and θ are model parameters. Inference is strictly left-to-right: at each step t , the model predicts x_t conditioned on the full prefix $x_{<t}$.

Masked-diffusion Inference. This paradigm underpins recent state-of-the-art models such as LLaDA (Bie et al., 2025; Nie et al., 2025; Zhu et al., 2025a,b) and Dream-7b (Ye et al., 2025). Let $\tilde{x} \sim q(\cdot | x)$ denote a masked variant of x , where positions in a masked index set $M(\tilde{x}) = \{i : \tilde{x}_i = [m]\}$

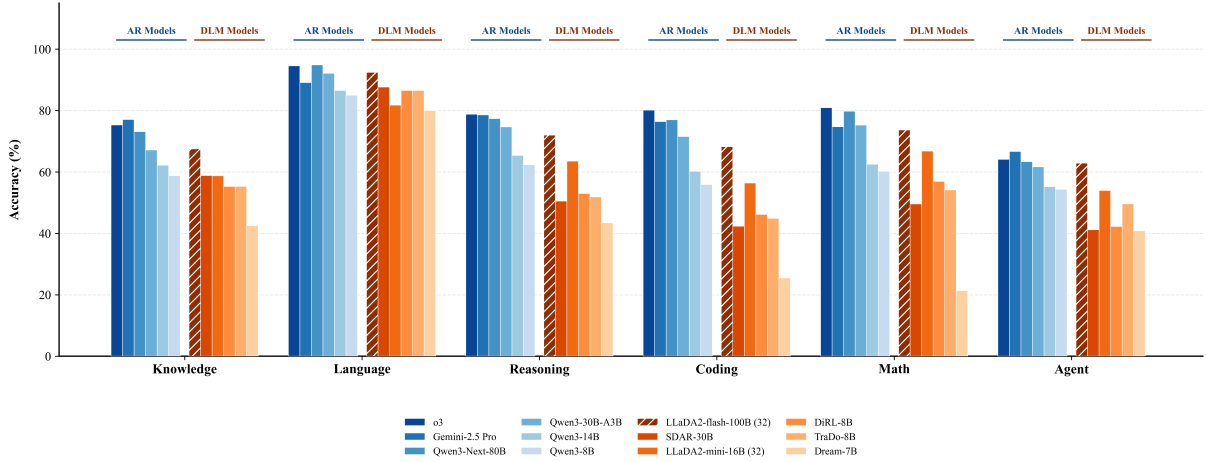


Figure 1: Overall performance comparison between AR and DLM models across six evaluation dimensions.

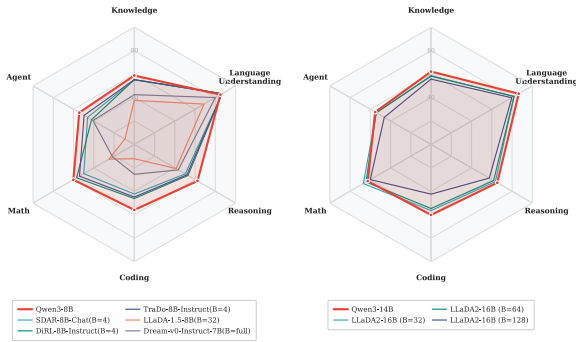


Figure 2: Comparison at the same parameter scale. Smaller block size B yields superior performance.

are replaced by $[m]$. In one parallel denoising iteration, we choose an update set $K \subseteq M(\tilde{x})$ and predict all tokens in K in a single forward pass using the (mean-field) approximation

$$p_{\theta}(x_K | \tilde{x}) \approx \prod_{i \in K} p_{\theta}(x_i | \tilde{x}), \quad (2)$$

s.t. $\tilde{x}_i = [m], \forall i \in K.$

where $x_K = \{x_i\}_{i \in K}$. Eq. (2) makes explicit that within-step parallelism assumes conditional independence among $\{x_i\}_{i \in K}$ given \tilde{x} , i.e., tokens updated in the same step cannot condition on each other.

To enable variable-length generation and efficient KV caching, these models adopt a blockwise framework that applies the above inference mechanism autoregressively. Specifically, a sequence x is partitioned into B contiguous blocks $\{x^{(1)}, \dots, x^{(B)}\}$. The generation follows a block-level autoregressive factorization:

$$\log p_{\theta}(x) = \sum_{b=1}^B \log p_{\theta}(x^{(b)} | x^{(<b)}). \quad (3)$$

For each block b , the distribution $p_{\theta}(x^{(b)} | x^{(<b)})$ is modeled by the masked diffusion process described in Eq. (2), where the denoising is conditioned on the fixed, previously generated blocks $x^{(<b)}$.

4 Reality Check: MDLMs Still Trail AR LMs at Scale

Despite the potential of MDLMs to challenge the AR paradigm, existing studies often rely on small-scale datasets, leaving their performance at scale largely unexplored. To provide a transparent and systematic evaluation, we conduct a large-scale benchmark of eight representative MDLMs against state-of-the-art AR models. Using a unified platform and 58 diverse benchmarks, this section serves as a "reality check" to quantify the current progress of the field and identify the remaining performance gaps between these two architectural paradigms.

4.1 Experimental Protocol: Models, 58 Benchmarks, and Unified Inference

Models and Benchmarks. We conduct a large-scale evaluation of 8 representative state-of-the-art MDLMs alongside frontier Autoregressive (AR) models, including OpenAI o3 and Gemini-2.5 Pro. To ensure a systematic assessment, we curate a comprehensive suite of **58 benchmarks** spanning six core dimensions: *Knowledge*, *Mathematics*, *Reasoning*, *Language Understanding*, *Agentic capabilities*, and *Coding*. We report mean accuracy over all test instances for each benchmark (one inference per instance). Due to space constraints, exhaustive model specifications and benchmark descriptions are provided in Appendix A.1.

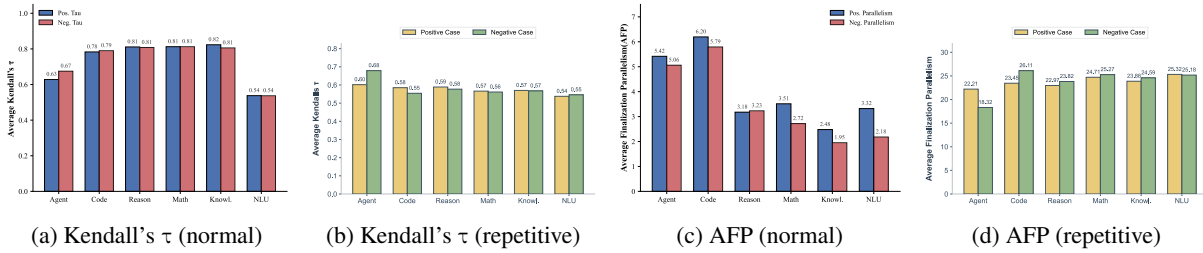


Figure 3: **Comparison of intra-chunk parallelism and sequential ordering patterns across different task domains.** Repeating samples are analyzed separately because, despite their limited count, each fills the entire 32k context window and disproportionately impacts overall statistical values.

Implementation and Fairness. All experiments are performed on a distributed cluster equipped with **512 NVIDIA GPUs** using a unified inference pipeline. Unless otherwise specified in the Appendix, each model is evaluated using its optimal default configurations. Notably, as we employ a standardized and more stringent evaluation protocol (e.g., unified prompt templates and rigorous answer-matching logic), the absolute scores for certain models may be lower than those reported in their respective technical reports. However, this controlled environment eliminates confounding variables and ensures a rigorous *apples-to-apples* comparison across disparate model architectures. Detailed implementation settings are deferred to Appendix A.1.

4.2 Empirical Trade-off: More Parallelism, Lower Accuracy

Detailed experimental results across 58 benchmarks are reported in Tables tables 1 to 4. Aggregated evaluations across six capability dimensions (as illustrated in Figure 1) indicate that a performance gap persists between current masked diffusion language models (MDLMs) and their autoregressive (AR) counterparts. A comparative analysis of models at a similar parameter scale in Figure 2 reveals that MDLMs achieving near-AR performance typically employ restricted block sizes. Notably, to attain peak precision, OpenPangu-7B-Diffusion utilizes a block size of 4 and constrains each step to decode only the token with the highest confidence. These findings collectively suggest an empirical conclusion: in contemporary architectures, excessive decoding parallelism serves as a primary bottleneck that compromises predictive accuracy.

4.3 Why This Gap Persists: The Parallel Factorization Limit

The fundamental limitation of Masked Discrete Diffusion Models (MDLMs) stems from the **con-**

ditional independence assumption (Liu et al., 2025a; Kang et al., 2025). This assumption necessitates factorizing the complex joint distribution of multiple tokens into a product of independent marginals during parallel decoding, thereby introducing significant and unavoidable approximation errors.

The Factorization Gap. Autoregressive (AR) models strictly adhere to the probabilistic chain rule, $P_{AR}(x_{1:n}|c) = \prod_{i=1}^n P(x_i|x_{<i}, c)$, to ensure global coherence. In contrast, an MDLM with a block size B approximates the joint distribution of a token block Y as the product of independent marginal distributions:

$$P_{\text{MDLM}}(Y|X) = \prod_{y_i \in Y} P_{\theta}(y_i|X, S_{<t}) \quad (4)$$

where $S_{<t}$ denotes tokens finalized in previous steps. When tokens exhibit strong syntactic, semantic, or logical coupling, this independence assumption leads to a pronounced structural bias. Following the framework of (Kang et al., 2025), even if P_{θ} achieves the optimal marginal distribution at each position, the KL divergence between this factorized form and the true data distribution $P_{data}(Y|X)$ is lower-bounded by the **Conditional Total Correlation (CTC)**:

$$\begin{aligned} \min_{\theta} D_{KL}(P_{data}||P_{\theta}) &\geq C(Y|X) \\ &= \sum_{y_i \in Y} H(y_i|X) - H(Y|X) \end{aligned} \quad (5)$$

Here, $C(Y|X)$ quantifies the statistical dependency strength among tokens in Y given the context X . Stronger dependencies result in a higher theoretical floor for quality loss, represented by the CTC bound. **Elucidating Empirical Observations.** The aforementioned framework clarifies the performance

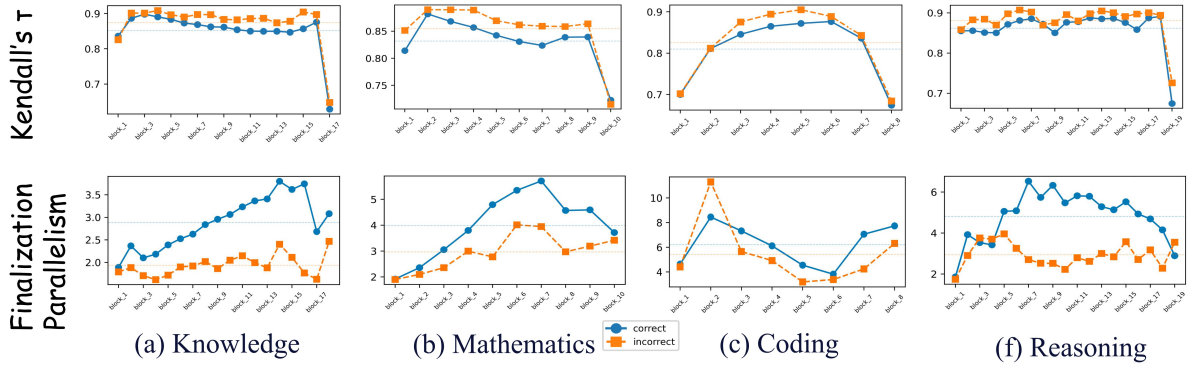


Figure 4: **Comparison of inter-chunk parallelism and sequential ordering patterns across different task domains.** Refer to Appendix A.3.1 for more groupings of τ , and Appendix A.3.2 for parallel AFP.

decay observed in Section 3.2: as the block size B (parallelism granularity) increases, the complexity of inter-token dependencies within Y grows, forcing the model to factorize under tighter intra-group constraints. This leads to a monotonic increase in the CTC-induced error bound. Our analysis suggests that the performance gap between MDLMs and AR models in high-parallelism settings is not merely a consequence of insufficient model capacity, but rather an intrinsic architectural limitation in representing high-dependency structures. This further explains why state-of-the-art methods (e.g., Trado, SDAR, openPangu (Tian et al., 2025)) often adopt highly conservative block sizes; they effectively sacrifice their defining architectural advantage—parallelism—to circumvent the theoretical precision ceiling imposed by independent factorization.

5 Decoding Dynamics of MDLMs: Disentangling Parallelism and Generation Order

While the preceding analysis identifies the current performance limits of MDLMs, we argue that their latent potential resides in the unique flexibility of their decoding trajectories. To characterize this behavior, we shift our perspective from macro-scale accuracy to a mechanistic deconstruction of the two defining dimensions in our title: Parallelism and Generation Order. Focusing on the state-of-the-art 100B-parameter LLaDA2-flash, we operationalize these properties via Average Finalization Parallelism (AFP, Eq. 6) and Kendall’s τ (Eq. 7) to conduct an in-depth analysis of the decoding dynamics of frontier diffusion models across diverse benchmarks.

5.1 Metrics for Decoding Dynamics: AFP and Kendall’s τ

To quantitatively analyze how MDLMs decode tokens, we introduce two key metrics.

Average Finalization Parallelism (AFP). For a generated sequence of length n , let $c_i \in \{1, \dots, T\}$ denote the decoding step at which token i is finalized. We define the number of *effective decoding steps* as $T_{\text{eff}} = |\{t \mid \exists i \text{ s.t. } c_i = t\}|$, i.e., the number of distinct steps that finalize at least one token. The *Average Finalization Parallelism* (AFP) is then

$$\text{AFP} = \frac{n}{T_{\text{eff}}} = \frac{n}{|\{t \mid \exists i \text{ s.t. } c_i = t\}|}. \quad (6)$$

Autoregressive decoding yields $\text{AFP} \approx 1$, while a fully parallel decoder that finalizes all tokens in a single step achieves $\text{AFP} = n$.

Kendall’s τ . To quantify the alignment between finalization and surface order, we adopt Kendall’s τ (Abdi, 2007), a rank correlation coefficient from statistics. For a sequence of length n , it measures the correlation between token indices i and their finalization steps c_i . A pair of indices (i, j) with $i < j$ is concordant if $c_i < c_j$ and discordant if $c_i > c_j$. The metric is defined as:

$$\tau = \frac{C - D}{n(n-1)/2}, \quad (7)$$

where C and D are the numbers of concordant and discordant pairs, respectively. While AR decoding yields $\tau = 1$ (strictly monotonic), $\tau < 1$ indicates a non-monotonic generation order where future tokens may be finalized before their predecessors.

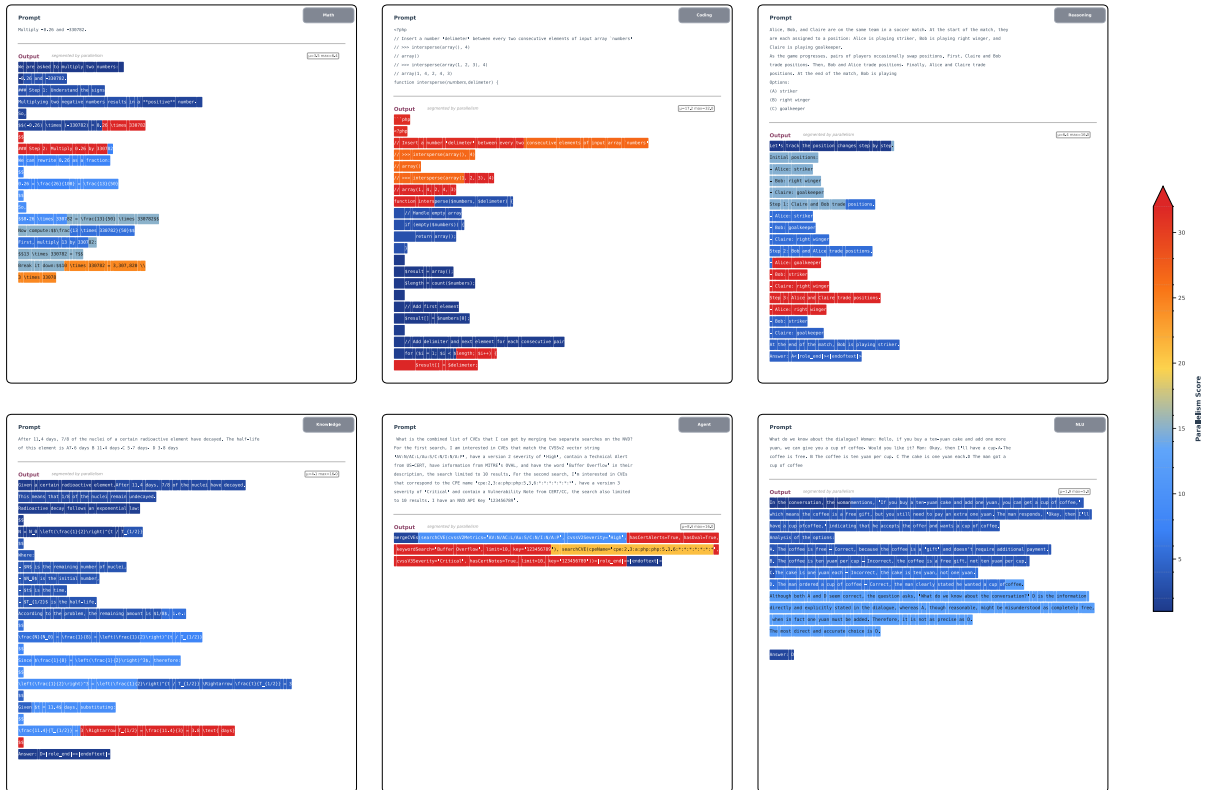


Figure 6: **Parallelism visualization plot.** Tokens are organized into blocks, each overlaid with a background color. The intensity of the color corresponds to the average number of tokens decoded per step within that block.

inates almost entirely from the global context X , rather than local interactions within the block Y . For example, the header of a for loop inherently determines the subsequent newline and indentation; since these tokens do not depend on each other’s local state, they can be generated in parallel.

Semantic Generation Order. To elucidate the underlying mechanism of generation order, we conduct a case study on the Knowledge Benchmark. We utilize Stanza (Qi et al., 2020) to perform Part-of-Speech (POS) tagging on model outputs and compute the average decoding step for various POS categories (results presented in Table 5). The results reveal a clear hierarchical pattern: the model first constructs a "structural backbone" and subsequently fills in "fine-grained details." This non-linear generation logic suggests that MDLMs do not simply construct text in a strictly left-to-right fashion. Instead, they prioritize the deployment of structural anchors by identifying global certainty, followed by the incremental refinement of complex modifiers. Further details are provided in Appendix A.4.

5.4 Case Studies: Grounding Dynamics in Semantic Anchors

Parallelism Cases. Figure 6 provides a qualitative validation of the aggregate statistical trends through representative case studies. First, we observe a distinct inverse correlation between parallel intensity and semantic novelty: parallelism peaks during the generation of deterministic content, such as numerical recapitulation, logical restatements, or formulaic templates. Second, in the Knowledge and Math domains, parallel intensity scales as the reasoning trajectory converges, manifesting a "clarification effect" where decoding accelerates once the solution path becomes certain. Finally, Coding tasks exhibit an initial surge in parallelism driven by structural boilerplate, which subsequently tapers off as the model transitions to complex core logic. These fine-grained observations are highly consistent with the statistical trajectories visualized in fig. 4.

Order Disruptions at Semantic Pivots. Beyond the high global sequentiality shown in figs. 3 and 4, we observe localized "plunges" in Kendall’s τ (fig. 7). These minima typically align with *semantic transitions* (e.g., shifting from chain-of-thought rea-

Prompt

Alice, Bob, and Claire are on the same team in a soccer match. At the start of the match, they are each assigned to a position: Alice is playing right midfielder, Bob is playing center midfielder, and Claire is playing striker. As the game progresses, pairs of players occasionally swap positions. First, Alice and Bob trade positions. Then, Alice and Claire trade positions. Finally, Bob and Claire trade positions. At the end of the match, Claire is playing

- (A) right midfielder
- (B) center midfielder
- (C) striker

Output

Let's track the positions step by step.
Initial positions:
Alice: right midfielder
Bob: center midfielder
Claire: striker
Step 1: Alice and Bob trade positions.
Alice moves to center midfielder
Bob moves to right midfielder
Claire remains striker
Current positions:
Alice: center midfielder
Bob: right midfielder
Claire: striker
Step 2: Alice and Claire trade positions.
Alice moves to striker
Claire moves to center midfielder
Bob remains right midfielder
Current positions:
Alice: striker
Bob: right midfielder
Claire: center midfielder
Step 3: Bob and Claire trade positions.
Bob moves to center midfielder
Claire moves to striker
Alice remains striker? Wait — no, Alice was at striker, but now Claire takes over.
Wait: After step 2:
Alice: striker
Bob: right midfielder
Claire: center midfielder
Now Bob and Claire trade:
Bob goes to center midfielder
Claire goes to striker
So final positions:
Alice: striker
Bob: center midfielder
Claire: striker
Wait — that can't be. Two people can't be striker.
Wait — let's double-check.
After step 2:
Alice: striker
Bob: right midfielder
Claire: center midfielder
Step 3: Bob and Claire trade.
Bob (right midfielder) swaps with Claire (center midfielder)
So:
Bob goes to center midfielder
Claire goes to right midfielder
Now final positions:
Alice: striker
Bob: center midfielder
Claire: right midfielder
So Claire is playing right midfielder.

Answer: A

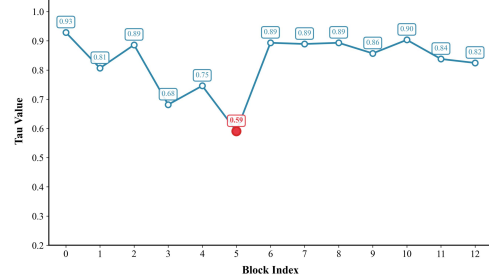


Figure 7: Case study. Highlighted blocks represent the local minima of τ . More cases in App. A.5.

soning to the final answer) or *structural boundaries* (e.g., paragraph breaks or code-to-comment transitions). At these junctions, the model momentarily diverges from a strict left-to-right trajectory. This suggests that MDLMs treat transition markers as *structural anchors* that are generated out-of-order or in parallel to "pre-plan" the layout of the subsequent thought block before fully finalizing the current one. Detailed case analyses are provided in appendix A.5.

6 Discussion: Unlocking the Potential of MDLMs

Driven by the findings in Section 5, we identify three key potentials for MDLMs: parallelism, generation order, and iterative editing.

1) Parallelism Potential: Acceleration. While MDLMs parallelize structured content effectively (Sec. 5.1), current speedup is limited by inference frameworks. Future diffusion-specific optimizations (e.g., *dinfer* (Ma et al., 2025)) will be essential to realizing their speed advantage in structured and long-form generation.

2) Sequential Potential: Non-Sequential Dependency. Unlike the unidirectional dependency of AR models, MDLMs support "any-order" inference, enabling them to handle tasks with non-linear dependency structures. **Evidence and Analysis:** In

our Sudoku experiments (Appendix A.6), as well as in comparative studies on chemical molecule design and protein design, MDLMs exhibit causal modeling capabilities that surpass those of AR models. The models spontaneously acquire an "easy-to-hard" strategy: they first fill in high-certainty "anchor points," and then leverage bidirectional attention to resolve global uncertainty. **Research Direction:** Current uniform masking strategies fail to account for the internal causal hierarchy of data, leading to an excessively large search space and numerous invalid paths during training. A key future challenge lies in introducing heuristic biases to optimize decoding path modeling by mining reverse causality.

3) Editing Potential: A Theoretical Path to Bridging the Accuracy Gap. To mitigate the loss of causal information stemming from the conditional independence assumption in parallel decoding, we explore the theoretical potential of MDLMs for parallel editing. Proofs provided in Appendix A.8 demonstrate that a "generate-then-edit" paradigm not only effectively recovers accuracy degradation but also redefines the boundaries of parallel generation. This paradigm enables the model to engage in bold probabilistic exploration during the initial stages, followed by efficient parallel editing to rectify biases in batches. This "fast sampling followed by parallel refinement" strategy theoretically allows

the model to achieve superior generation quality compared to single-pass inference within a shorter equivalent time. This signifies the transition of MDLMs from mere samplers to self-evolving reasoning systems. Furthermore, we implemented an adaptation of editing capabilities on LLaDA2-mini-16B, the preliminary experimental validation of which is detailed in Appendix A.7.

7 Conclusion

To the best of our knowledge, this work presents the first systematic and large-scale evaluation of state-of-the-art Masked Diffusion Language Models (MDLMs). By benchmarking across 58 diverse tasks and performing an in-depth mechanistic analysis of the 100B LLaDA model, we reveal the fundamental decoding dynamics that govern parallel and arbitrary-order generation. Our findings, grounded in both extensive empirical evidence and theoretical proofs, illuminate the limits of current parallel factorization while highlighting the immense potential of MDLMs for non-linear causal modeling and efficient iterative editing. This study provides a foundational blueprint for the development of next-generation non-autoregressive language models that are both logically flexible and computationally efficient.

Limitations

Potential Risks and Misinterpretation. We evaluate all models under a unified local pipeline (prompting/decoding/scoring) to control confounders and enable fair apples-to-apples relative comparisons across architectures. However, these controlled scores should not be treated as leaderboard-equivalent or directly compared to numbers from original reports, since prompts, decoding constraints, and answer-matching can differ and may lead to different absolute results. In addition, hardware-specific optimization can affect outcomes: e.g., OpenPangu is primarily optimized for Huawei Ascend, and our NVIDIA environment (without Ascend-specific kernels) may yield different practical behavior, so such differences should not be over-interpreted as purely modeling effects.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62476242). We thank Ant Group for providing the computing

resources and technical support. ChatGPT was used for language polishing.

References

- Hervé Abdi. 2007. The kendall rank correlation coefficient. *Encyclopedia of measurement and statistics*, 2:508–510.
- Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*.
- Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Matej Rosca, David Krueger, Ilya Sutskever, and 1 others. 2021b. Program synthesis with large language models. In *arXiv preprint arXiv:2108.07732*.
- Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, and 1 others. 2025. Llada2.0: Scaling up diffusion language models to 100b. *arXiv preprint arXiv:2512.15745*.
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.
- Jiahuan Cao, Yongxin Shi, Dezhi Peng, Yang Liu, and Lianwen Jin. 2024. C³ bench: A comprehensive classical chinese understanding benchmark for large language models. *arXiv preprint arXiv:2405.17732*.
- Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, and 1 others. 2022. Multiple: A scalable and extensible approach to benchmarking neural code generation. *arXiv preprint arXiv:2208.08227*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating large language models trained on code](#). *Preprint*, arXiv:2107.03374.
- Qiguang Chen, Hanjing Li, Libo Qin, Dengyun Peng, Jinhao Liu, Jiangyi Wang, Chengyue Wu, Xie Chen, Yantao Du, and Wanxiang Che. 2025. Beyond surface

- reasoning: Unveiling the true long chain-of-thought capacity of diffusion large language models. *arXiv preprint arXiv:2510.09544*.
- Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. Theoremqa: A theorem-driven question answering dataset. *arXiv preprint arXiv:2305.12524*.
- Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and 1 others. 2025. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.
- Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.
- Yin Fang, Xiaozhuan Liang, Ningyu Zhang, Kangwei Liu, Rui Huang, Zhuo Chen, Xiaohui Fan, and Hua-jun Chen. 2023. Mol-instructions: A large-scale biomolecular instruction dataset for large language models. *arXiv preprint arXiv:2306.08018*.
- Guhao Feng, Yihan Geng, Jian Guan, Wei Wu, Liwei Wang, and Di He. 2025. Theoretical benefit and limitation of diffusion language model. *arXiv preprint arXiv:2502.09622*.
- Paul Gauthier. 2023. Aider: Ai pair programming in your terminal. <https://github.com/paul-gauthier/aider>.
- Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2025. Diffucoder: Understanding and improving masked diffusion models for code generation. *arXiv preprint arXiv:2506.20639*.
- Stanford NLP Group and 1 others. 2019. SQuAD 2.0: The stanford question answering dataset.
- Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I Wang. 2024. Cruxeval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2020. [Measuring massive multitask language understanding](#). *ArXiv*, abs/2009.03300.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021a. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- HMMT Organizing Committee. 2025. Harvard-mit mathematics tournament (hmmt). <https://www.hmmt.co/>. Problems released post-contest; Accessed: 2024-12-19.
- Emiel Hoogetboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465.
- Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kübler, and Lawrence S Moss. 2020. Ocnli: Original chinese natural language inference. *arXiv preprint arXiv:2010.05444*.
- Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, and 1 others. 2023. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36:62991–63010.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. [Live-codebench: Holistic and contamination free evaluation of large language models for code](#). *Preprint*, arXiv:2403.07974.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

- Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Il Koo, Nam Ik Cho, and 1 others. 2025. Parallelbench: Understanding the trade-offs of parallel decoding in diffusion llms. *arXiv preprint arXiv:2510.04767*.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.
- Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. *KR*, 2012(13th):3.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.
- Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2024a. CMMLU: Measuring massive multitask language understanding in chinese. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11260–11285.
- Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024b. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *arXiv preprint arXiv:2402.19255*.
- Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. 2025. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*.
- Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, and 1 others. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.
- Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. 2025. ZebraLogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*.
- Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. 2024. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. *arXiv preprint arXiv:2405.12209*.
- Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yongan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. 2025a. Tidar: Think in diffusion, talk in autoregression. *arXiv preprint arXiv:2511.08923*.
- Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. 2025b. Are your llms capable of stable reasoning? In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17594–17632.
- Kaijing Ma, Xinrun Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, and 1 others. 2024. Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. *arXiv preprint arXiv:2410.06526*.
- Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, and 1 others. 2025. dinfer: An efficient inference framework for diffusion language models. *arXiv preprint arXiv:2510.08666*.
- Mathematical Association of America. 2024. American invitational mathematics examination (aime). <https://www.maa.org/math-competitions/american-invitational-mathematics-examination-aime>. Problems released annually; Accessed: 2024-12-19.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *arXiv preprint arXiv:2502.09992*.
- Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.
- Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, and 1 others. 2025. Humanity’s last exam. *arXiv preprint arXiv:2501.14249*.
- Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.
- Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Haoxu Zhang, Yi Hu, and 1 others. 2025. Phybench: Holistic evaluation of physical perception and reasoning in large language models. *arXiv preprint arXiv:2504.16074*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- James V Roggeveen, Erik Y Wang, Will Flintoft, Peter Donets, Lucy S Nathwani, Nickolas Gutierrez,

- David Ettel, Anton Marius Graf, Siddharth Dandavate, Arjun Nageswaran, and 1 others. 2025. Hardmath2: A benchmark for applied mathematics built by students as part of a graduate class. *arXiv preprint arXiv:2505.11774*.
- Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.
- Humza Sami, Samy Charas, Asav Gandhi, Pierre-Emmanuel Gaillardon, Valerio Tenace, and 1 others. 2025. Nexus: A lightweight and scalable multi-agent framework for complex tasks automation. *arXiv preprint arXiv:2502.19091*.
- Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.
- Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2023. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. *arXiv preprint arXiv:2310.16049*.
- Stanford CRFM. 2023. College-Level Math Problems dataset. <https://crfm.stanford.edu/2023/10/19/helm-math.html>. Extracted from HELM Math suite; Accessed: 2024-12-19.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.
- Minyang Tian, Luyu Gao, Shizhuo Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, and 1 others. 2024. Scicode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 37:30624–30650.
- Yuchuan Tian, Yuchen Liang, Jiacheng Sun, Shuo Zhang, Guangwen Yang, Yingte Shu, Sibofang, Tianyu Guo, Kai Han, Chao Xu, and 1 others. 2025. From next-token to next-block: A principled adaptation path for diffusion llms. *arXiv preprint arXiv:2512.06776*.
- Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Looma, Shichang Zhang, Yizhou Sun, and Wei Wang. 2023. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*.
- Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. 2025. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, and 1 others. 2024. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.
- Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. 2023. Cmath: Can your language model pass chinese elementary school math test? *arXiv preprint arXiv:2306.16636*.
- Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2025. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. *arXiv preprint arXiv:2505.22618*.
- Jin Xu, Xiaojiang Liu, Jianhao Yan, Deng Cai, Huayang Li, and Jian Li. 2022. Learning to break the loop: Analyzing and mitigating repetitions for neural text generation. *Advances in Neural Information Processing Systems*, 35:3082–3095.
- Xin Xu, Jiacheng Zhang, Tianhao Chen, Zitong Chao, Jishan Hu, and Can Yang. 2025. Ugmathbench: A diverse and dynamic benchmark for undergraduate-level mathematical reasoning with large language models. *arXiv preprint arXiv:2501.13766*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2024. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*.
- Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *Preprint*, arXiv:1809.08887.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.
- Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. Evaluating the performance of large language models on gaokao benchmark. *arXiv preprint arXiv:2305.12474*.

- Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314.
- Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and 1 others. 2025a. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*.
- Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, and 1 others. 2025b. Llada-moe: A sparse moe diffusion language model. *arXiv preprint arXiv:2509.24389*.
- Qin Zhu, Fei Huang, Runyu Peng, Keming Lu, Bowen Yu, Qinyuan Cheng, Xipeng Qiu, Xuanjing Huang, and Junyang Lin. 2025c. Autologi: Automated generation of logic puzzles for evaluating reasoning abilities of large language models. *arXiv preprint arXiv:2502.16906*.
- Ying Zhu, Jiabin Wan, Tianyi Liang, Xu Guo, Xiaoran Liu, Zengfeng Huang, Ziwei He, and Xipeng Qiu. 2025d. [Dirl: An efficient post-training framework for diffusion language models](#).
- Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kadour, Ming Xu, Zhihan Zhang, and 14 others. 2025. [Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions](#). *Preprint*, arXiv:2406.15877.
- Lucas Zimmer, Marius Lindauer, and Frank Hutter. 2021. Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE transactions on pattern analysis and machine intelligence*, 43(9):3079–3090.

A Appendix

A.1 Details of Experimental Protocol

Model Selection and Configurations. We evaluate 8 state-of-the-art MDLMs, including LLaDA, Trado (Wang et al., 2025), and SDAR (Cheng et al., 2025), with scales reaching 100B parameters (e.g., LLaDA2-flash (Bie et al., 2025)). These models span both block-wise and global diffusion decoding paradigms. For clarity, the inference block size for each model is explicitly documented in our experimental result tables to facilitate a direct comparison of parallelism granularity. As baselines, we benchmark against frontier AR models, including the open-source Qwen3 series (Yang et al., 2025) and proprietary models such as Gemini-2.5 Pro (Comanici et al., 2025) and OpenAI o3.

Benchmarks. To fill the void in systematic assessments for MDLMs, we curate **58 benchmarks** categorized into six dimensions: (i) Knowledge (e.g., MMLU); (ii) Mathematics (e.g., GSM8K, MATH); (iii) Reasoning (e.g., BBH, GPQA); (iv) Language Understanding (e.g., Hellaswag); (v) Agentic (e.g., BFCL); (vi) Coding (e.g., HumanEval). This suite quantifies the performance gap between diffusion and autoregressive paradigms across diverse domains. See Appendix A.2 for a complete list.

Implementation Details. Our large-scale evaluation is conducted on a distributed cluster of 512 NVIDIA GPUs. To maintain the architectural integrity and default configurations of each model, we perform inference using the original implementations provided in their respective official open-source repositories. For OpenPangu, we observe that it is natively optimized for the Huawei Ascend architecture and lacks optimized kernels for NVIDIA hardware. Due to the resulting excessive computational overhead, evaluating OpenPangu on the full 58-benchmark suite was infeasible; we therefore report its performance on a representative subset.

Unified evaluation protocol. All results are obtained by running each model locally. Except for the parameter configurations specified in the table, all other settings follow the best-performing configurations reported in each model’s technical report, and results are generated under a unified evaluation pipeline. While this ensures a fair, apples-to-apples comparison across models and benchmarks, the absolute scores of some models may be lower than those reported in their original papers or public leaderboards due to differences in

evaluation settings (e.g., prompts, and our stricter answer-matching/content scoring). Therefore, we recommend interpreting our results primarily as relative comparisons under a controlled protocol, rather than as leaderboard-equivalent numbers.

A.2 Details of Benchmarks

This section provides detailed descriptions of the 58 benchmarks used in our large-scale evaluation. Benchmarks are grouped into six categories: Coding, Mathematics, Knowledge, Language Understanding, Reasoning, and Agentic Agentic Capabilities.

Coding

To comprehensively assess coding capabilities, our evaluation spans from foundational synthesis to complex, domain-specific reasoning. We use **CruxEval-O** to evaluate a model’s ability to predict program outputs (Gu et al., 2024); **mbpp** to assess the synthesis of basic Python programs from natural language (Austin et al., 2021b); **MultiPL-E** to test multilingual code generation capabilities (Cassano et al., 2022); and **humaneval** for standard code generation from docstrings (Chen et al., 2021). Furthermore, we assess performance on more complex tasks with **livecodebench_v6**, using novel problems from programming contests (Jain et al., 2024); **Bigcodebench-Full**, which challenges models to invoke function calls as tools (Zhuo et al., 2025); **LCBench** for learning curve analysis (Zimmer et al., 2021); the complex text-to-SQL benchmark **spider** (Yu et al., 2019); the challenging algorithmic problems from **CodeForces** (Li et al., 2022); the scientific-focused **SciCode** benchmark (Tian et al., 2024); and **Aider** for testing interactive code editing skills (Gauthier, 2023).

Mathematics

To evaluate mathematical capabilities, we employ a diverse set of benchmarks. We begin with foundational multi-step reasoning using **GSM8K** (Cobbe et al., 2021) and its augmented version, **GSM_Plus** (Li et al., 2024b). We then assess advanced problem-solving with the challenging **MATH** competition dataset (Hendrycks et al., 2021a), a curated subset (**math500** (Hendrycks et al., 2021b)), and its Chinese-language parallel, **CMATH** (Wei et al., 2023), supplemented by **GKMathUnion**. To probe the limits of model reasoning, we use elite-level competition problems from **OlympiadBench** (He et al., 2024), recent editions of the **AIME**

Table 1: **Performance comparison across code benchmarks.** "Block Size 4 (max)" refers to the parallel computation of 4-position logits where only the single most confident token is decoded. Other configurations utilize threshold-based adaptive decoding, accepting all tokens that exceed a specific confidence level.

Model		Coding											
Model Name	Block Size	CruxEval-O	mbsp	MultiPL-E	humaneval	openai_humaneval	livecode_benchv6	Bigcode_bench	LCBench	spider	Code Forces	SciCode	Aider
qwen3-8B	/	74.25	79.80	67.14	79.73	84.76	26.60	36.14	48.73	72.94	26.14	17.71	56.95
qwen3-14B	/	79.25	83.26	69.07	84.91	87.80	29.46	38.51	55.61	76.94	27.39	21.96	68.42
qwen3-30B-A3B-Instruct-2507	/	90.31	86.86	75.64	89.41	93.29	47.36	41.14	79.22	81.60	73.56	26.22	74.25
qwen3-Next-80B-A3B-Instruct-o3	/	94.38	94.82	78.87	90.93	93.90	61.23	44.91	90.75	84.12	72.68	32.64	84.77
Gemini-2.5 Pro	/	99.38	98.59	71.23	89.94	98.17	73.79	40.35	94.97	77.69	86.49	40.97	90.23
TraDo-8B-Instruct	4	94.38	98.33	48.26	91.16	25.00	72.08	89.97	89.97	85.10	84.77	44.10	93.98
TraDo-4B-Instruct	4	58.56	79.10	34.11	44.82	81.10	20.93	34.30	40.19	72.33	15.22	10.42	48.68
DiRL-8B-Instruct	4	52.81	68.15	21.18	27.29	78.05	14.10	30.96	26.46	71.63	6.00	11.81	32.71
SDAR-30B-A3B-Chat	4	62.50	74.94	48.64	68.14	73.78	15.97	34.30	32.83	70.98	17.21	10.94	43.80
SDAR-4B-Chat	4	60.62	75.18	23.04	12.27	85.37	21.37	38.60	47.06	76.99	12.77	9.11	45.49
SDAR-8B-Chat	4	51.06	69.09	17.65	14.10	71.95	10.79	29.30	24.83	69.59	5.62	9.81	30.83
Dream-v0-Instruct-7B	full	56.81	74.88	31.48	55.11	77.44	12.17	34.47	32.41	71.03	7.69	12.33	42.29
LLaDA-8B-Instruct	32	40.00	52.55	25.74	16.16	52.44	5.40	18.95	8.63	47.28	9.14	-	5.26
LLaDA-1.5	32	20.50	21.52	11.61	16.01	19.51	1.60	2.02	3.06	28.32	1.17	-	4.51
openPangu-7B-Diffusion-Base	32	21.69	22.10	11.71	14.79	23.17	1.76	2.19	3.29	28.60	1.17	-	3.57
openPangu-7B-Diffusion-Base	4(max)	42.00	-	-	-	18.90	-	-	18.05	-	-	-	0.00
LLaDA2-mini-16B	32	41.50	-	-	-	67.68	-	-	21.26	-	-	-	-
	64	72.62	82.17	70.25	81.25	85.37	31.72	33.07	67.81	76.99	22.84	13.19	39.85
	128	69.44	81.06	69.32	81.71	85.98	28.52	31.14	64.44	74.57	17.31	12.50	39.85
	32	31.31	65.46	63.70	67.30	71.95	16.85	27.46	55.44	61.76	23.31	4.17	20.30
	64	85.44	89.14	76.01	87.65	94.51	42.51	40.88	82.91	82.49	47.72	24.65	65.04
LLaDA2-flash-100B	64	79.00	86.62	74.77	85.29	91.46	42.73	39.82	78.49	81.37	51.58	21.88	63.53
	128	42.19	42.10	72.92	67.68	71.95	18.56	25.00	68.55	46.86	47.35	7.90	32.52

(24/25) (Mathematical Association of America, 2024) and HMMT25 (HMMT Organizing Committee, 2025), and the exceptionally difficult **HARD-Math2** (Roggeveen et al., 2025). Performance on higher education material is measured using **College_Math** (Stanford CRFM, 2023), the proof-based **UGMathBench** (Xu et al., 2025), and the theorem-application test **TheoremQA** (Chen et al., 2023). Finally, for a holistic view, we include composite and dynamic benchmarks like **MathBench** (Liu et al., 2024), the **Minerva_Math** problem set (Lewkowycz et al., 2022), and the community-driven **Livemathbench** (Liu et al., 2025b).

Knowledge and Language Understanding

Our assessment of knowledge and language understanding covers a wide range, from broad factual recall to deep linguistic reasoning. For general world knowledge, we use the Massive Multitask Language Understanding (**MMLU**) benchmark (Hendrycks et al., 2020) and its more challenging expert-level variant, **MMLU-PRO** (Wang et al., 2024). We also assess knowledge in Chinese contexts through its counterpart, **CMMLU** (Li et al., 2024a), and the comprehensive **C-EVAL** suite (Huang et al., 2023). Performance on human-standardized exams is measured with **AGIEval** (Zhong et al., 2024), which includes tasks from exams like the SAT, and **GAOKAO-Bench** (Zhang et al., 2023), which is based on the Chinese college entrance exam. To test deep, specialized knowledge, we use the "Google-proof" **GPQA** benchmark (Rein et al., 2024), the scientific reasoning challenge **ARC-c** (Clark et al.,

2018), the multi-disciplinary **scibench** (Wang et al., 2023), and the high-difficulty physics benchmark **PhyBench** (Qiu et al., 2025). Open-domain question answering and fact retrieval are assessed using **TriviaQA** (Joshi et al., 2017) and the real-world query-based **NQ** (Natural Questions) (Kwiatkowski et al., 2019). Finally, deeper linguistic comprehension is evaluated through the Classical Chinese understanding benchmark **C⁸** (Cao et al., 2024), the coreference resolution task **WSC** (Winograd Schema Challenge) (Levesque et al., 2012), and **SQuAD2.0** (Group et al., 2019), which tests a model’s ability to handle unanswerable questions.

Reasoning and Agentic Capabilities

We evaluate advanced reasoning and agentic abilities across a comprehensive set of benchmarks. For *reasoning*, we include: multi-step abstract and compositional reasoning (**BBH** (Suzgun et al., 2023), **BBeH**); commonsense and physical reasoning in grounded scenarios (**HellaSwag** (Zellers et al., 2019), **PIQA** (Bisk et al., 2020)); formal logical deduction via synthetic world models (**ProntoQA** (Saparov and He, 2022)), constraint-satisfaction puzzles (**ZebraLogic** (Lin et al., 2025)), and open-ended bilingual logic (**AutoLogi** (Zhu et al., 2025c)); expert-level academic reasoning resistant to web retrieval (**HLE** (Phan et al., 2025)); multistep soft reasoning over natural language narratives (**MuSR** (Sprague et al., 2023)); Chinese textual entailment (**OCNLI** (Hu et al., 2020)); discrete numerical reasoning over passages (**DROP** (Dua et al., 2019)); and knowledge-orthogonal rule ap-

Table 2: **Performance comparison across mathematics benchmarks.** "Block Size 4 (max)" refers to the parallel computation of 4-position logits where only the single most confident token is decoded. Other configurations utilize threshold-based adaptive decoding, accepting all tokens that exceed a specific confidence level.

Model		Mathematics																
Model Name	Block Size	GSM8K	math	Olympiad Bench	CMATH	GSM-Plus	College-Math	Math Bench	Minerva-Math	math500	GKMATH Union	Live mathbench	AIME24	AIME25	HARD Math2	UGMath Bench	Theorem QA	HMMT 25
qwen3-8B	/	93.73	86.24	55.07	95.54	86.06	83.07	83.65	32.05	86.20	77.46	63.39	30.10	22.29	5.57	58.77	53.00	11.35
qwen3-14B	/	95.11	88.54	58.63	95.42	87.32	84.81	88.16	35.78	88.85	79.84	66.63	32.50	26.25	1.78	62.15	59.72	11.41
qwen3-30B-A3B-Instruct-2507	/	96.51	96.54	77.67	96.61	89.64	89.39	94.58	35.78	98.00	91.11	85.52	76.09	61.30	4.38	70.64	72.00	43.65
qwen3-Next-80B-A3B-Instruct o3	/	96.42	91.52	80.37	97.04	96.17	89.82	95.79	45.83	91.53	91.83	88.07	87.73	68.64	38.27	72.80	75.76	48.43
Gemini-2.5 Pro	/	95.45	96.30	82.81	94.72	88.83	89.67	93.00	19.79	94.05	91.87	82.48	88.59	85.62	44.31	74.55	76.81	76.95
TraDo-8B-Instruct	4	92.99	85.78	53.04	95.72	87.30	90.13	95.31	33.88	88.75	92.62	89.28	87.81	84.27	43.36	74.47	75.94	0.00
TraDo-8B-Instruct	4	92.51	78.88	44.59	93.58	83.73	77.79	81.61	38.60	81.20	63.33	45.90	16.88	16.46	4.27	49.25	48.50	3.33
TraDo-4B-Instruct	4	91.76	76.74	41.04	93.26	82.95	75.30	80.16	34.99	76.35	58.41	40.30	13.54	6.67	0.47	48.48	42.38	6.67
DiRL-8B-Instruct	4	93.35	83.70	50.11	92.03	85.70	77.71	79.83	36.76	85.05	72.70	56.56	16.88	16.88	4.38	51.45	54.62	10.42
SDAR-30B-A3B-Chat	4	91.51	77.52	35.26	92.67	82.95	65.72	79.68	27.94	78.30	56.23	41.80	6.67	10.21	2.49	46.32	44.25	3.54
SDAR-4B-Chat	4	91.26	70.46	34.59	91.26	80.63	66.47	77.12	22.73	70.25	52.54	34.02	6.46	6.67	0.47	38.66	33.88	3.33
SDAR-8B-Chat	4	90.86	74.02	40.15	92.58	82.21	73.46	78.97	33.27	74.70	56.87	38.66	16.25	3.75	1.90	44.19	43.25	3.33
Dream-v0-Instruct-7B	full	9.61	36.08	11.93	63.07	4.87	68.24	46.41	1.29	36.60	10.60	19.26	0.89	0.68	0.83	15.77	25.56	0.05
LLaDA-8B-Instruct	32	74.17	32.72	7.33	58.24	61.61	24.20	38.53	7.90	25.80	25.04	8.27	0.62	0.16	0.47	14.84	13.63	0.16
LLaDA-1.5	32	75.55	34.32	8.00	65.39	64.49	25.87	43.96	10.17	29.70	26.94	9.15	1.15	0.10	0.47	16.81	15.00	0.21
openPangu-7B-Diffusion-Base	32	26.52	-	-	-	-	-	-	14.52	48.00	-	-	-	-	-	-	-	-
openPangu-7B-Diffusion-Base 4(max)	4(max)	-	-	-	-	-	-	-	3.98	36.00	-	-	-	-	-	1.07	-	-
LLaDA2-mini-16B	32	94.24	93.18	67.56	95.54	86.29	85.42	87.61	30.70	93.70	87.18	77.32	46.67	40.83	3.79	58.09	63.75	24.38
LLaDA2-mini-16B	64	94.29	92.96	66.96	95.36	86.31	85.95	86.01	32.54	93.10	85.67	74.04	53.12	36.67	2.84	53.97	63.38	18.54
LLaDA2-mini-16B	128	89.14	85.98	63.19	89.75	80.61	83.07	78.81	31.80	89.80	82.86	69.67	48.54	38.33	1.42	35.16	59.38	21.46
LLaDA2-mini-16B	32	96.36	95.60	73.48	96.99	89.60	88.57	92.03	35.66	97.60	89.84	83.88	63.33	56.67	4.27	68.63	72.25	47.50
LLaDA2-flash-100B	64	96.13	95.14	72.30	95.72	89.65	88.86	89.64	38.24	96.40	89.05	82.51	61.67	49.58	3.79	66.73	70.88	52.92
LLaDA2-flash-100B	128	62.85	80.48	66.81	68.17	62.24	88.50	34.36	28.68	95.00	78.85	82.14	42.92	2.84	33.95	69.12	27.92	

Table 3: **Performance comparison across Knowledge and Language benchmarks.** "Block Size 4 (max)" refers to the parallel computation of 4-position logits where only the single most confident token is decoded. Other configurations utilize threshold-based adaptive decoding, accepting all tokens that exceed a specific confidence level.

Model		Knowledge											Language			
Model Name	Block Size	MMLU	MMLU-PRO	GPQA	AGIEval	GAOKAO-Bench	Trivia QA	NQ	ARC-c	scibench	Phy bench	CMMLU	C-EVAL	C3	WSC	squad2.0
qwen3-8B	/	82.35	66.12	47.92	71.19	86.13	52.47	30.91	92.71	3.51	11.89	79.09	81.44	92.05	77.70	85.25
qwen3-14B	/	84.67	70.49	51.42	77.21	90.17	61.19	34.07	93.69	2.97	13.49	82.37	84.64	94.08	74.70	90.89
qwen3-30B-A3B-Instruct-2507	/	87.20	74.09	54.83	84.60	94.24	65.52	39.25	95.81	4.57	31.86	86.46	87.86	96.16	90.62	89.71
qwen3-Next-80B-A3B-Instruct o3	/	96.84	81.62	73.45	87.43	91.32	72.86	58.89	96.86	1.53	32.76	88.89	95.39	97.42	95.97	91.32
Gemini-2.5 Pro	/	91.59	82.09	81.41	87.65	93.57	86.21	66.65	96.82	4.24	39.81	86.47	87.51	98.14	99.04	86.49
TraDo-8B-Instruct	4	92.87	85.59	84.34	88.77	96.91	84.29	63.27	97.20	0.81	52.63	89.00	89.67	98.14	99.94	69.38
TraDo-8B-Instruct	4	79.69	58.96	39.33	64.96	83.66	58.35	34.40	91.53	3.53	6.55	77.66	78.37	93.97	78.85	89.14
TraDo-4B-Instruct	4	75.81	53.62	35.26	59.53	77.67	48.44	28.73	89.83	2.69	8.23	73.14	74.59	91.07	57.69	87.80
DiRL-8B-Instruct	4	81.08	44.87	44.07	61.18	82.08	56.57	33.07	90.17	2.30	11.81	76.95	79.44	93.48	78.85	87.41
SDAR-30B-A3B-Chat	4	80.92	59.82	36.84	69.29	86.81	65.23	39.53	94.83	3.01	7.34	81.71	81.16	96.00	78.85	88.23
SDAR-4B-Chat	4	74.12	50.09	29.45	59.23	74.42	47.48	28.39	91.19	2.50	4.14	72.33	72.25	91.01	60.58	87.26
SDAR-8B-Chat	4	77.72	55.17	36.96	64.08	82.41	57.33	33.30	92.20	2.92	5.74	76.89	76.91	93.42	77.88	89.02
Dream-v0-Instruct-7B	full	69.28	42.10	32.04	46.36	47.43	50.33	24.63	89.53	1.56	3.30	51.66	51.80	83.62	71.33	85.12
LLaDA-8B-Instruct	32	50.78	25.31	23.30	39.41	49.39	37.56	18.37	80.59	1.09	1.54	45.76	46.54	70.03	32.75	88.17
LLaDA-1.5	32	56.00	30.19	24.53	42.15	54.30	37.48	19.14	79.45	0.40	2.23	52.42	52.98	80.49	37.56	88.48
openPangu-7B-Diffusion-Base	32	-	-	-	-	-	-	-	80.34	2.27	4.35	-	68.82	86.58	48.08	52.19
openPangu-7B-Diffusion-Base 4(max)	4(max)	-	-	-	-	-	-	-	76.27	3.31	5.59	-	72.24	83.34	51.92	-
LLaDA2-mini-16B	32	80.58	64.21	47.22	72.74	83.79	52.59	32.58	92.88	3.71	13.67	79.41	81.86	89.64	69.23	86.45
LLaDA2-mini-16B	64	80.58	62.43	45.74	72.09	83.40	52.07	32.27	92.88	4.20	13.58	79.76	79.94	89.26	70.43	86.46
LLaDA2-mini-16B	128	76.93	59.33	46.28	70.01	74.15	50.01	28.86	89.41	5.78	7.83	80.16	80.16	89.04	68.75	81.34
LLaDA2-mini-16B	32	87.91	74.84	62.25	82.01	93.29	66.81	44.79	95.93	3.74	27.71	85.05	85.93	94.19	93.27	89.88
LLaDA2-flash-100B	64	86.75	73.89	65.59	80.72	92.71	66.74	44.57	94.58	4.06	20.53	83.53	84.62	93.21	93.27	89.75
LLaDA2-flash-100B	128	42.13	21.99	46.53	46.31	57.26	65.10	40.75	31.86	7.57	22.02	26.99	34.98	50.58	24.04	76.85

plication in out-of-distribution settings (**KorBench** (Ma et al., 2024)). For *agentic* capabilities, we assess precise tool/API invocation correctness via **BFCL** (Patil et al., 2025), and multi-agent coordination, workflow orchestration, and supervisor hierarchy management using **Nexus** (Sami et al., 2025).

A.3 Details of Inter-block Patterns.

A.3.1 Dynamic variations of τ values across six domains for positive and negative samples.

See figs. 8 to 13 for details.

A.3.2 Dynamic variations of AFP values across six domains for positive and negative samples.

See figs. 12, 14 to 17 and 19 for details.

A.4 Details of Part-of-Speech(POS) Decoding Order

Taking the knowledge benchmark as an example, we collected the Part-of-Speech (POS) sequences of the output content within each generation block. The results are shown in Table 5. We observe that the model exhibits a distinct temporal order in its POS tags.

First, tags typically associated with simple or independent content—such as INTJ (interjection), X (other), NUM (number), PROP (proper noun), and SYM (symbol)—appear relatively early. This indicates that the model tends to initiate generation with simple tokens like numbers, names, and symbols. Second, structural cues also emerge early: PUNCT (punctuation) and CONJ (coordinating conjunction) generally appear at lower average steps, implying that the model quickly inserts delimiters and connectors before drafting substantive con-

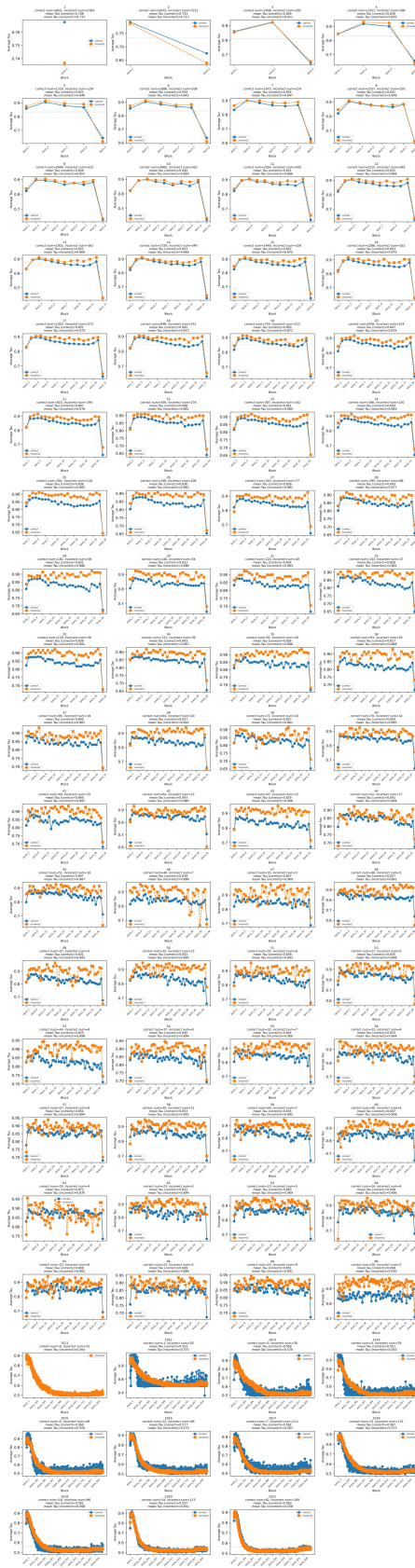


Figure 8: Evolution of τ values for grouped samples across knowledge domains.

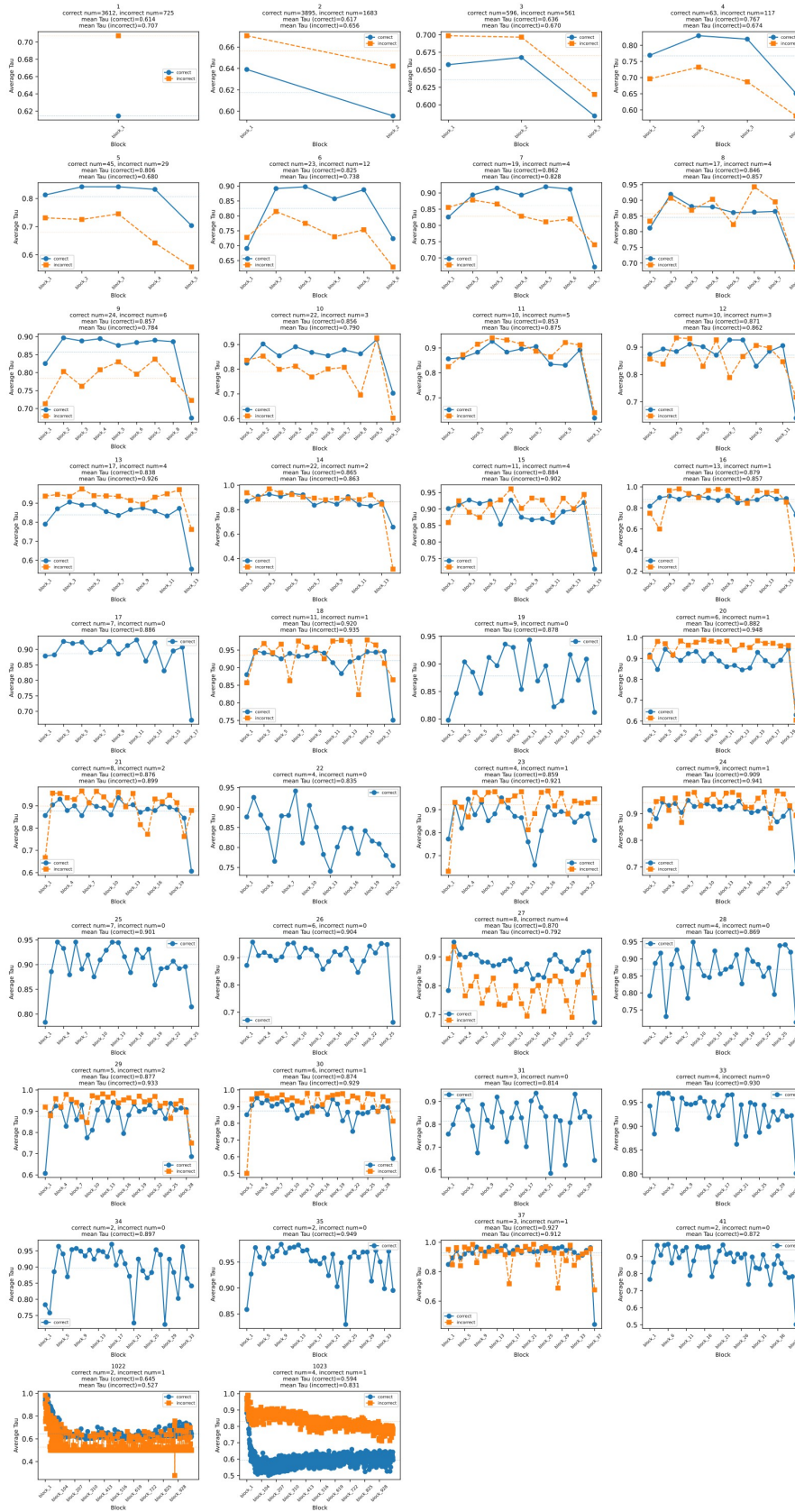


Figure 9: Evolution of τ values for grouped samples across agent domains.

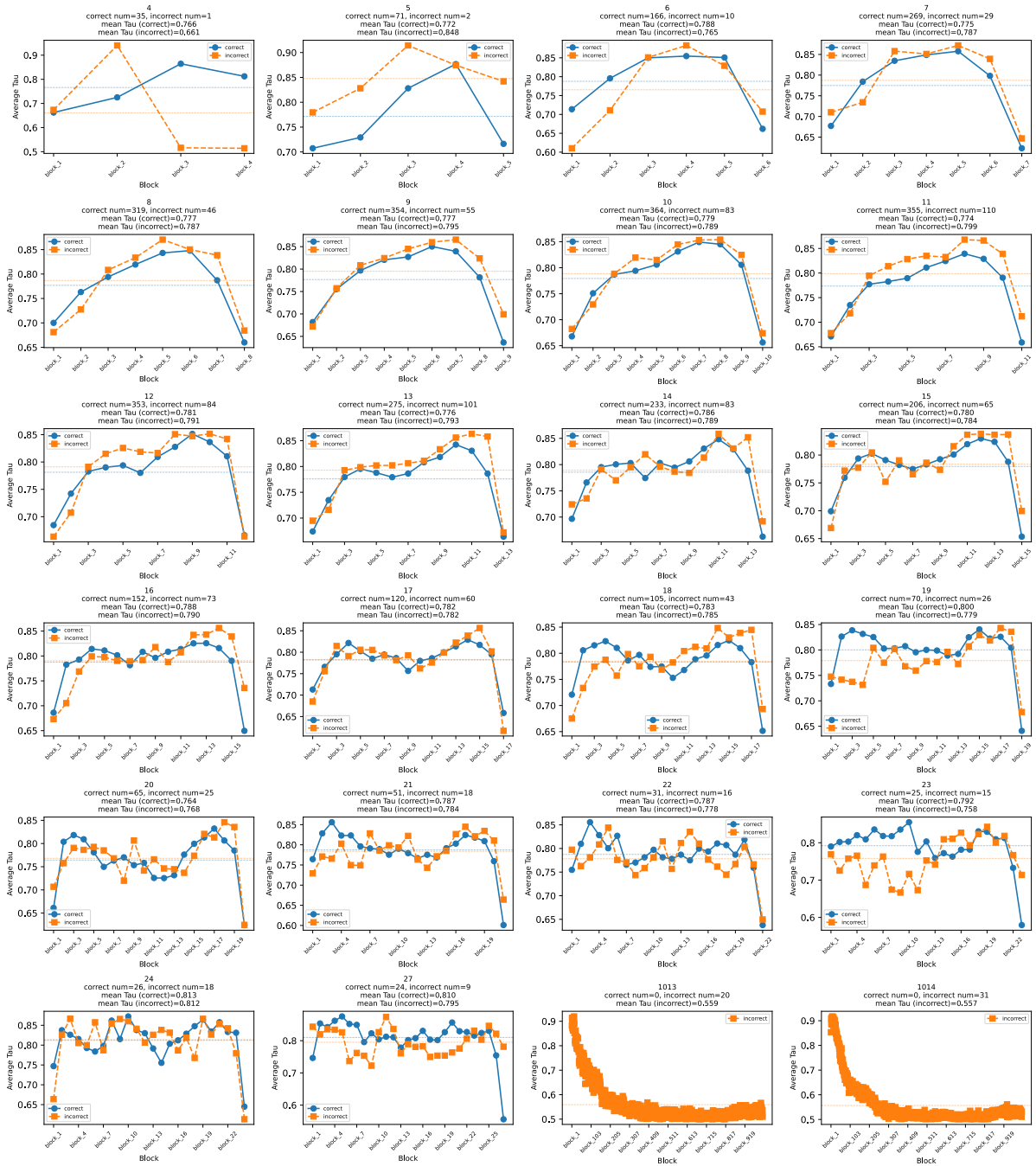


Figure 10: Evolution of τ values for grouped samples across coding domains.

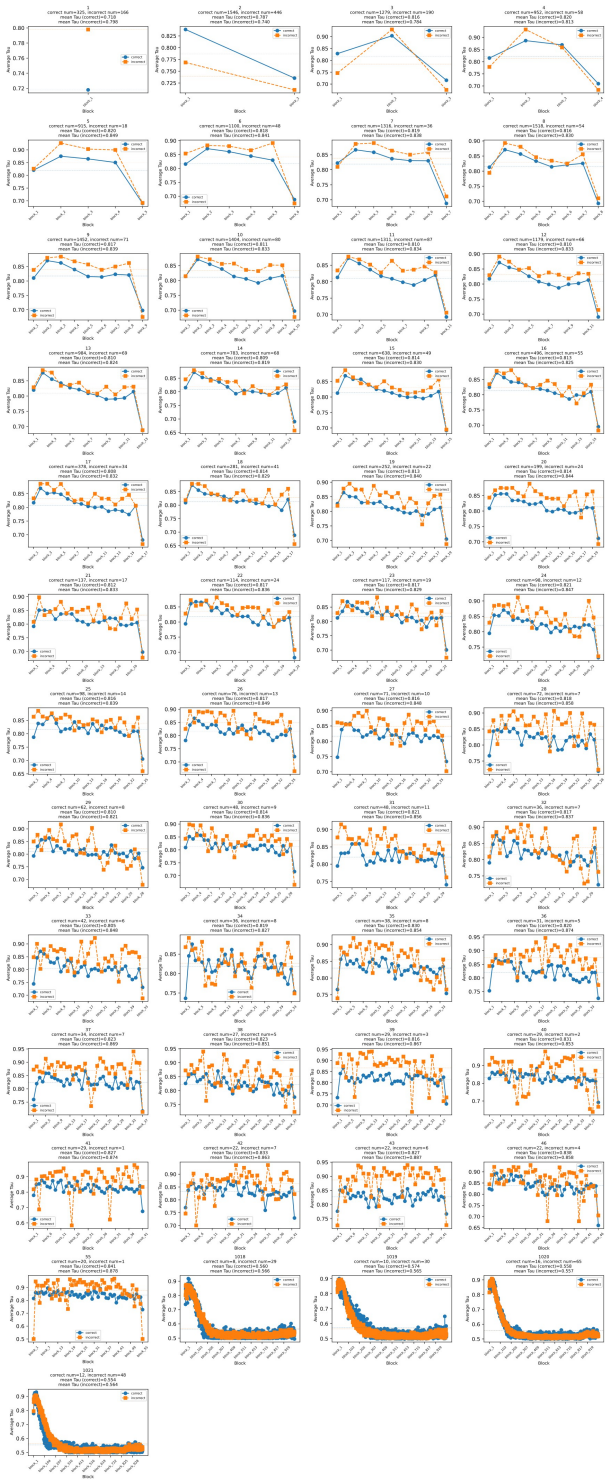


Figure 11: Evolution of τ values for grouped samples across math domains.

Table 4: **Performance comparison across Reasoning and Agent benchmarks.** "Block Size 4 (max)" refers to the parallel computation of 4-position logits where only the single most confident token is decoded. Other configurations utilize threshold-based adaptive decoding, accepting all tokens that exceed a specific confidence level.

Model		Reasoning										Agent			
Model Name	Block Size	ocnli	drop	kor bench	bbh	bbeh	hella swag	piqa	Pronto QA	HLE	Zebra Logic	Auto Logi	MuSR	BFCL	nexus
qwen3-8B	/	60.88	84.39	58.80	78.71	17.87	79.19	88.25	93.19	3.85	37.45	76.49	69.24	70.27	38.46
qwen3-14B	/	67.22	84.78	62.48	83.09	16.96	86.52	88.68	88.06	3.20	48.98	82.13	72.41	67.23	43.17
qwen3-30B-A3B-Instruct-2507	/	71.29	89.88	74.16	85.38	37.30	86.39	91.78	96.38	5.89	91.08	88.24	78.89	73.43	49.92
qwen3-Next-80B-A3B-Instruct o3	/	73.86	91.32	78.48	86.88	47.16	89.13	93.85	97.50	7.04	95.58	89.52	78.73	76.24	50.50
o3	/	75.12	88.09	81.04	89.71	66.20	92.67	96.19	76.12	15.85	93.90	88.53	82.33	75.19	53.08
Gemini-2.5 Pro	/	73.29	87.35	81.36	88.66	66.75	92.86	96.35	96.88	7.83	91.90	77.61	82.31	75.16	58.18
TraDo-8B-Instruct	4	65.39	60.03	51.12	59.38	14.96	83.68	87.38	73.12	3.01	14.10	49.75	61.03	58.15	41.09
TraDo-4B-Instruct	4	62.98	59.77	47.36	54.98	14.25	81.77	83.13	61.12	4.12	9.22	40.91	55.61	41.69	34.90
DiRL-8B-Instruct	4	65.80	45.52	51.84	61.80	16.31	77.72	86.83	93.50	3.99	13.60	61.00	58.21	45.98	38.53
SDAR-30B-A3B-Chat	4	65.76	76.92	50.40	55.15	15.04	88.93	91.29	48.12	3.85	11.10	41.84	57.97	46.10	36.27
SDAR-4B-Chat	4	62.58	60.06	46.24	53.43	13.43	83.14	81.99	59.00	3.89	8.00	35.16	53.58	40.03	34.07
SDAR-8B-Chat	4	64.17	65.09	49.04	52.18	13.51	83.16	86.83	72.62	3.20	11.12	43.70	59.15	53.18	39.12
Dream-v0-Instruct-7B	full	49.46	74.67	33.52	43.58	13.58	69.12	87.70	66.94	4.54	4.35	26.82	46.95	52.53	29.14
LLaDA-8B-Instruct	32	24.85	58.66	34.64	38.95	10.87	68.58	71.93	51.19	3.94	-	13.17	46.74	-	11.15
LLaDA-1.5	32	35.73	63.79	35.12	42.69	10.97	67.66	73.45	56.44	4.73	-	13.43	49.79	-	9.27
openPangu-7B-Diffusion-Base	32	-	-	-	-	-	-	72.74	-	-	-	-	49.49	-	-
openPangu-7B-Diffusion-Base 4(max)	4(max)	-	-	-	-	-	-	69.21	-	-	-	-	52.65	-	-
LLaDA2-mini-16B	32	63.80	85.76	54.40	78.70	15.79	78.87	86.40	88.88	4.36	62.48	71.81	71.10	71.15	36.80
	64	63.32	83.92	53.52	79.00	16.00	78.92	86.78	83.75	4.82	47.25	68.99	74.41	71.79	36.53
	128	62.61	77.79	49.44	74.65	14.55	78.74	86.78	80.00	4.08	34.30	57.13	69.50	69.19	23.23
	32	71.86	89.45	67.76	87.00	28.27	85.08	92.93	97.00	4.91	82.05	79.53	78.23	74.90	50.82
LLaDA2-flash-100B	64	68.31	85.51	68.24	84.87	27.40	83.60	91.19	97.00	4.17	74.60	81.27	78.64	74.54	49.05
	128	6.10	30.07	56.08	33.88	21.20	58.72	52.45	53.50	4.36	50.75	72.18	54.03	64.52	27.70

tent. The core substance of the sentences is then constructed during the intermediate steps, where NOUN (noun), VERB (verb), PRON (pronoun), and AUX (auxiliary) are most prevalent. Finally, tags that add supplementary details, such as ADJ (adjective), ADV (adverb), and ADP (adposition), tend to appear later, suggesting that the model typically appends descriptions and relational context once the main content is established.

Notably, this sequence is not merely a reflection of tag frequency: some highly frequent tags (e.g., NOUN and PUNCT) are not the earliest to appear, while several less common tags emerge at the very beginning. This further supports the hypothesis that the model "builds the basic structure first, then populates the details."

Table 5: **Global POS tendency measured by average local step (knowledge benchmark).** For each generated block, we tag every token with a POS label and record its *local step* (its position index inside the block, starting from 1). We then compute the **Global Avg. Step** for each POS tag by averaging these local steps over all occurrences of this tag in the whole dataset. A smaller average step means that this POS tag is more likely to appear earlier in a block, while a larger value means it tends to appear later. **Total Count** is the total number of tokens assigned to this POS tag in our collected outputs. For clarity, the tags are sorted by **Global Avg. Step** in ascending order. Note that these numbers describe a position trend inside blocks, and they are not the same as the overall POS frequency in normal text.

POS Tag	Global Avg. Step	Total Count
INTJ	2.3418	437156
X	2.7302	1135829
NUM	3.3031	8419293
PROPN	3.6147	5188437
SYM	3.8025	2816180
CCONJ	4.5384	1894660
PUNCT	4.5618	11005503
AUX	4.7912	3089759
NOUN	4.8039	15598091
PRON	4.8151	2060850
SCONJ	4.8545	660041
PART	4.9402	1099089
DET	5.2555	3905146
ADV	5.4416	2307803
VERB	5.6032	3916307
ADP	6.0675	2759438
ADJ	6.2864	2922280

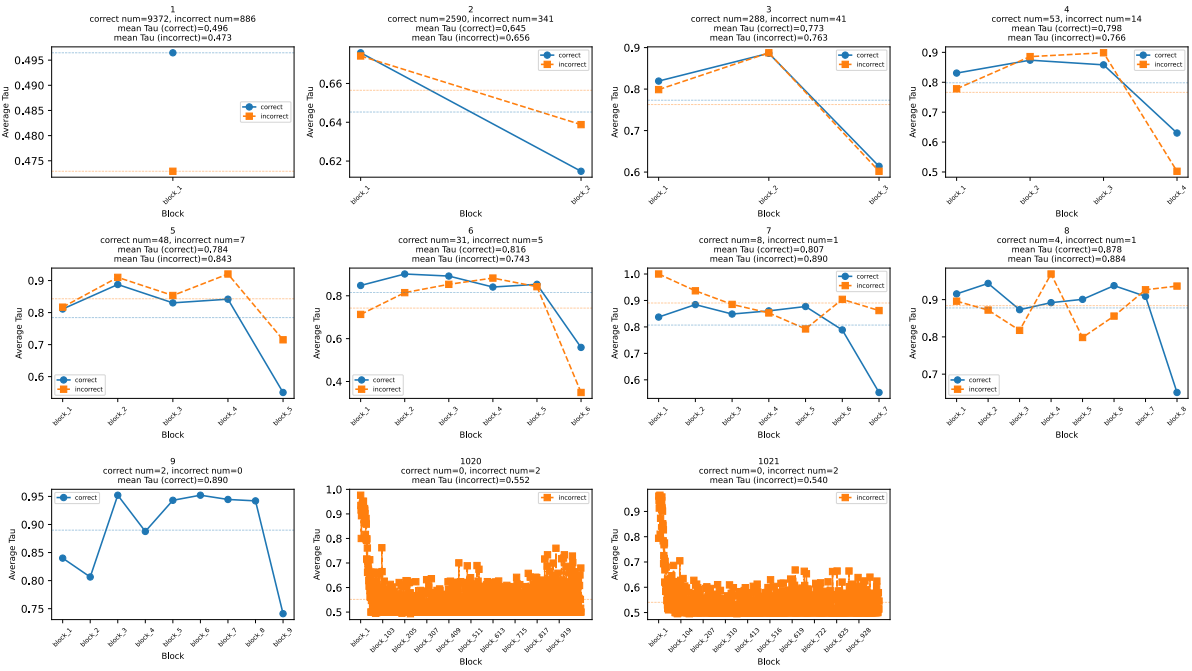


Figure 12: Evolution of τ values for grouped samples across language understanding domains.

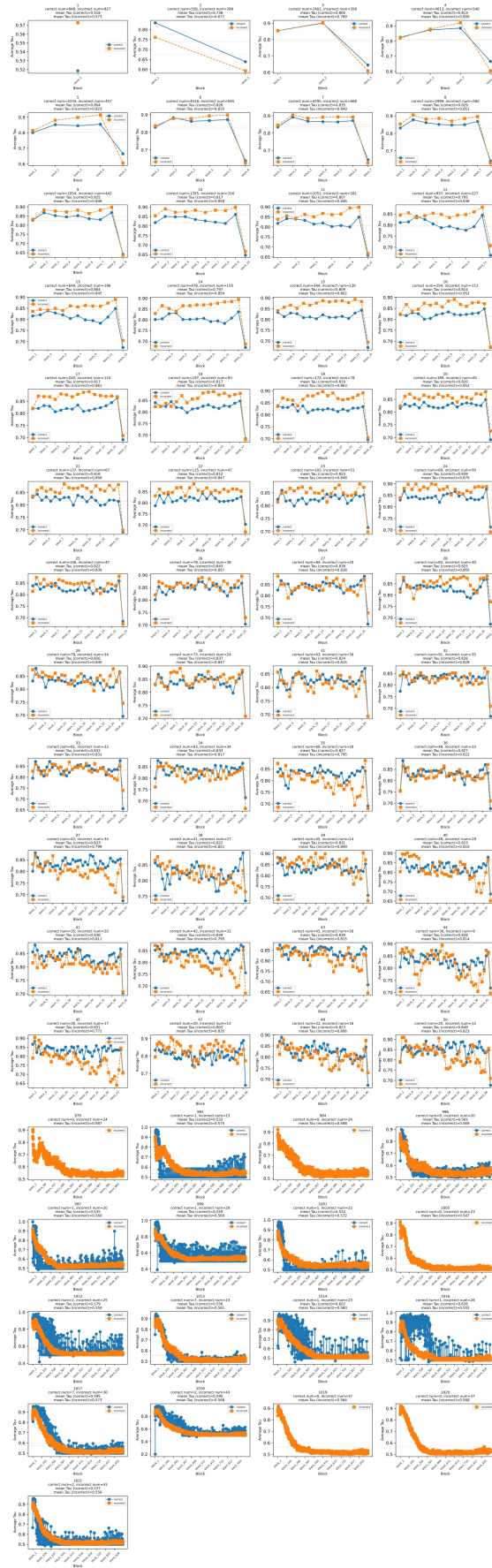


Figure 13: Evolution of τ values for grouped samples across reasoning domains.



Figure 14: Evolution of afp values for grouped samples across agent domains.

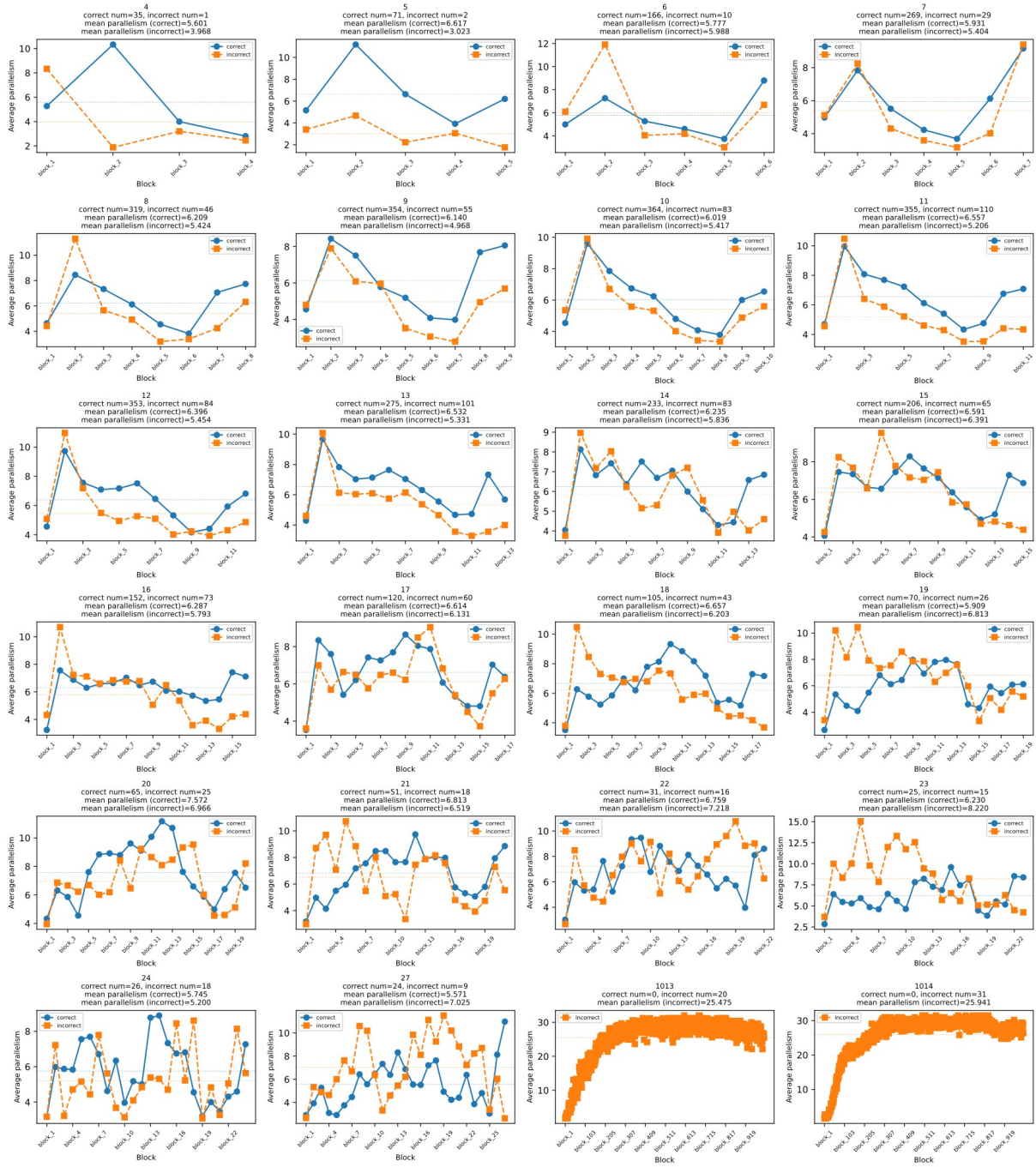


Figure 15: Evolution of afp values for grouped samples across code domains.

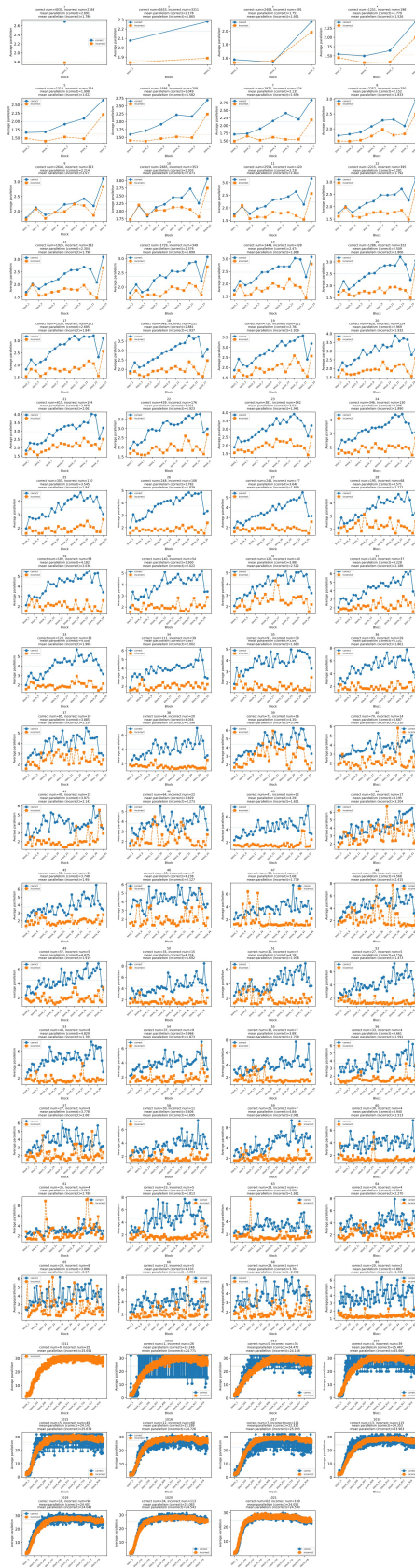


Figure 16: Evolution of afp values for grouped samples across knowledge domains.

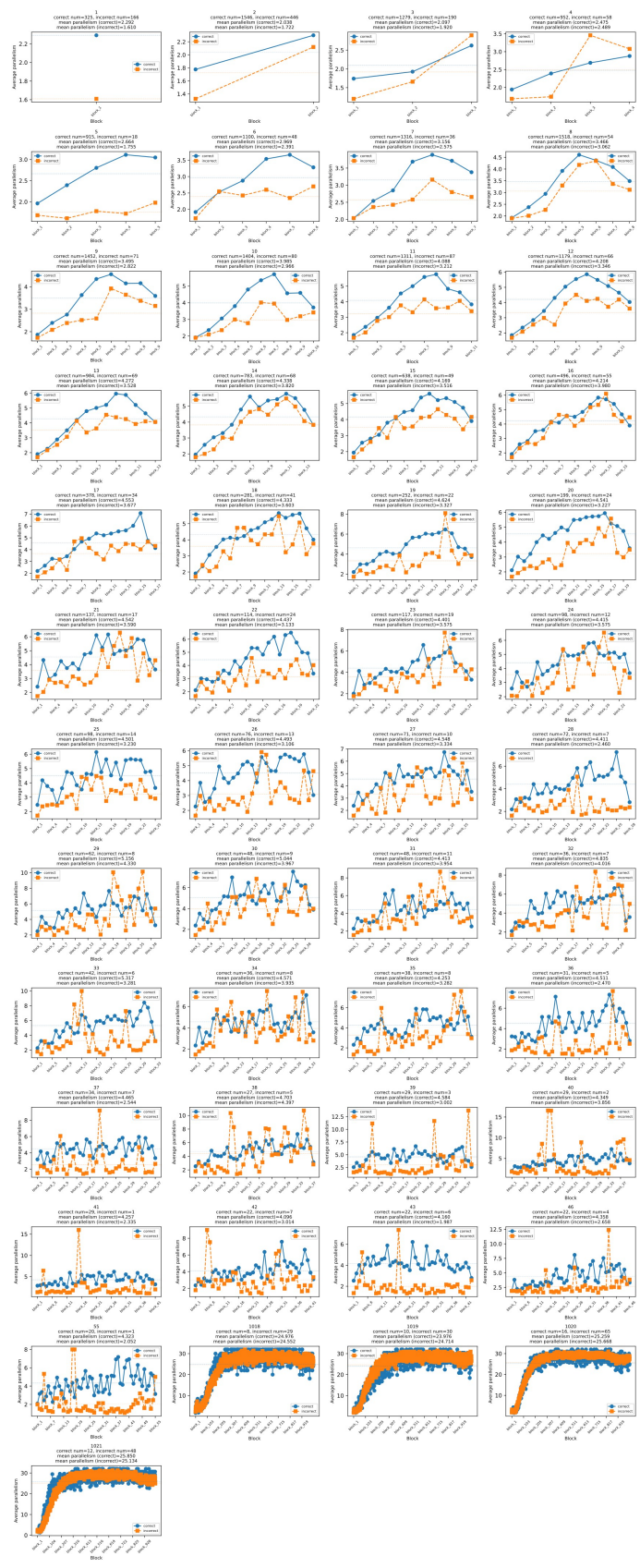


Figure 17: Evolution of afp values for grouped samples across math domains.

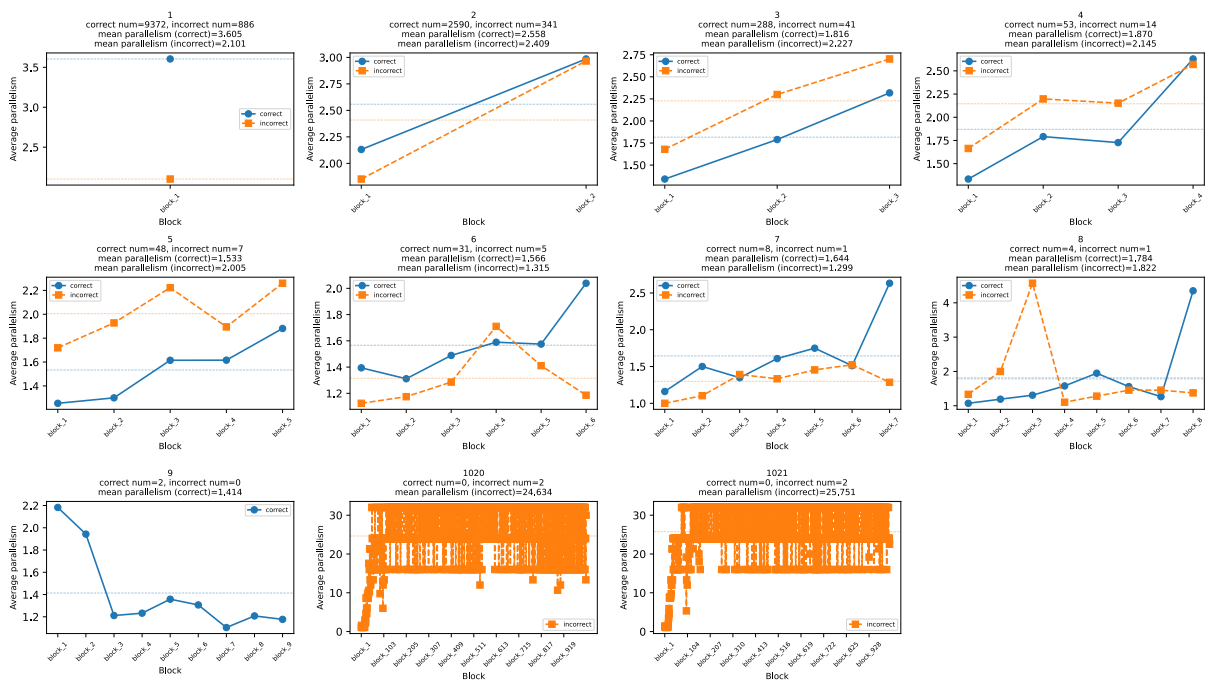


Figure 18: Evolution of afp values for grouped samples across language understanding domains.

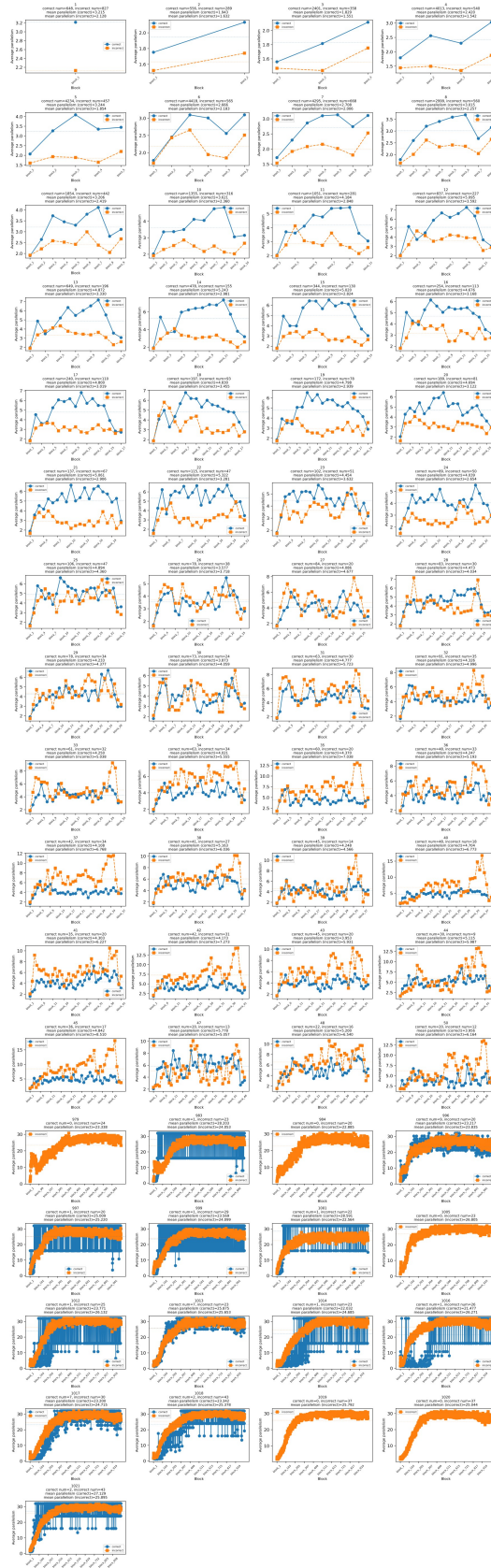


Figure 19: Evolution of afp values for grouped samples across reasoning domains.

A.5 Order Disruptions at Semantic Pivots

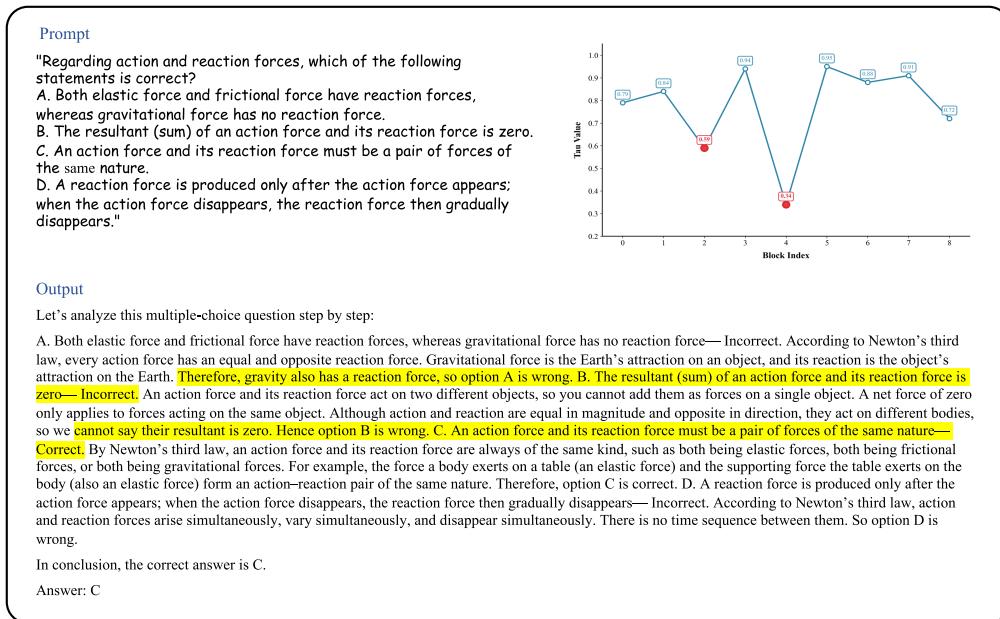


Figure 20: **case study.** This τ local-minimum output falls exactly at the boundary between “closing the previous reasoning chain” and “opening the next one.” The model first uses “cannot say their resultant is zero. Hence option B is wrong.” to simultaneously finalize B’s key rationale and its incorrect conclusion, and then immediately switches to “C. . . . Correct.” to start a new evaluation. Because this moment involves multiple constraints—summarization, switching, structural labeling, and planning the next line of argument—the demand for parallel organization is greatest, so the curve reaches its lowest value here.

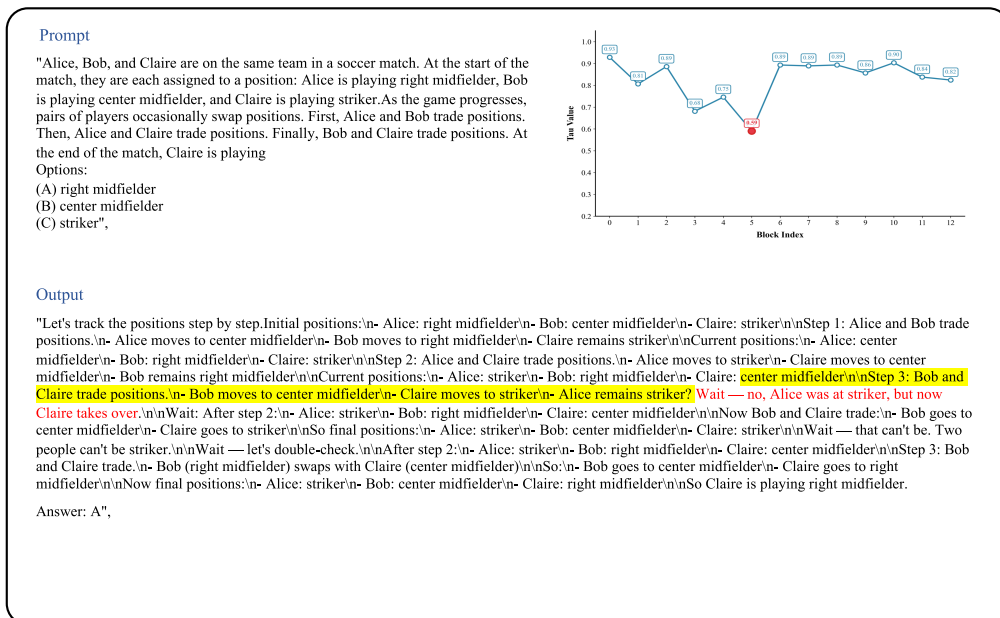


Figure 21: **case study.** The minimum τ value occurs at the critical state-update point of the "third swap," where a single operation has to change two people’s positions at the same time. Moreover, step 3 is the final swap: right after writing these two updates, the model must immediately derive the final position assignment. Models therefore tend to initiate both "updating" and "checking whether it is reasonable / whether it will conflict" mechanisms simultaneously at this point. The later output—“Wait . . . / Two people can’t be striker”—is precisely the explicit surfacing of that validation, and the minimum τ value often appears around the transition just before and after the check is triggered.

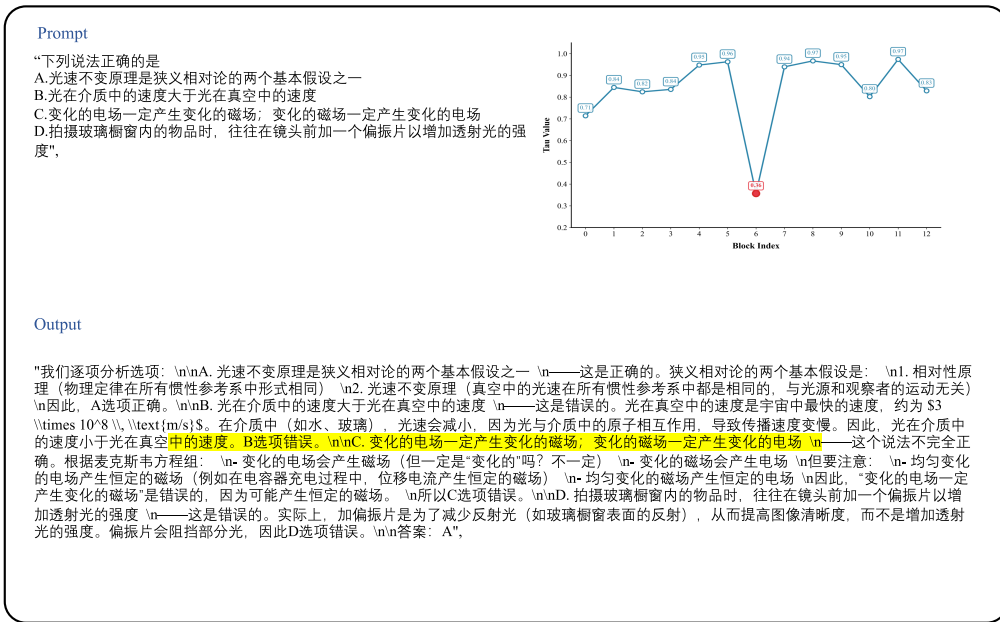


Figure 22: **case study.**At the boundary where "B ends and C begins," the model's output shows the highest degree of parallelism. At this point, the model must handle two things in parallel: on the one hand, it needs to finalize B's conclusion and bring the wording to a close; on the other hand, it needs to launch C's new argument framework (resetting the subject under discussion and the criteria for judgment). This kind of "state switching across paragraphs/across options" creates a stronger need for parallel organization.

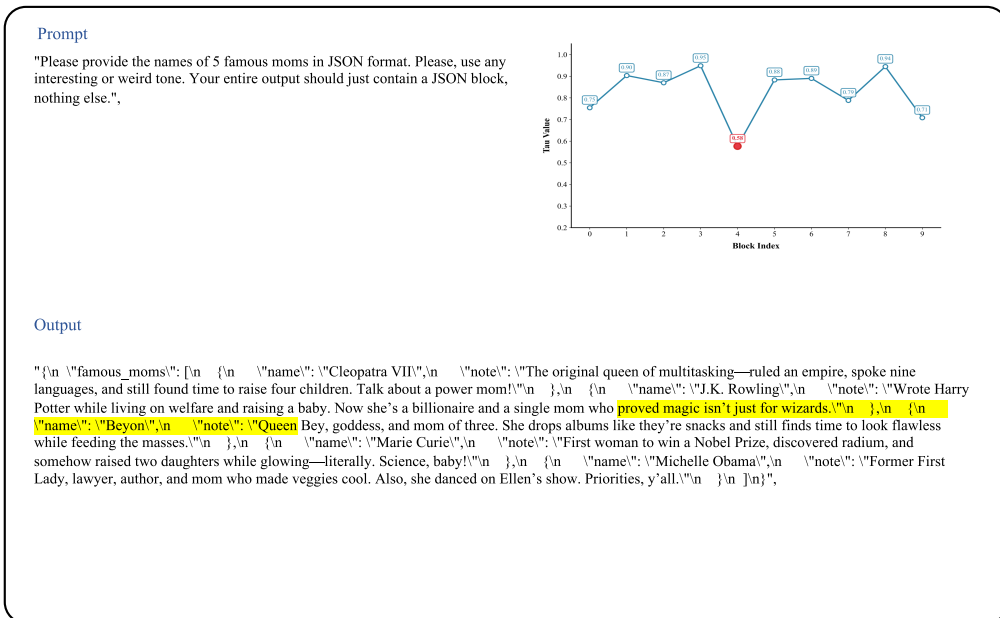


Figure 23: **case study.** The minimum τ value occurs in this segment because it sits at the boundary between "the end of the previous item + the beginning of the next item." At the same moment, the model must handle in parallel the correctness of the JSON structure, the semantic wrap-up of the previous item, the planning of the next entry, and the entity switch—moving from Rowling's concluding sentence to the start of Beyoncé's entry, which involves both an entity change and stylistic continuity. Entity switching typically requires more parallel retrieval and planning, so the degree of parallelism is higher at the switch point.

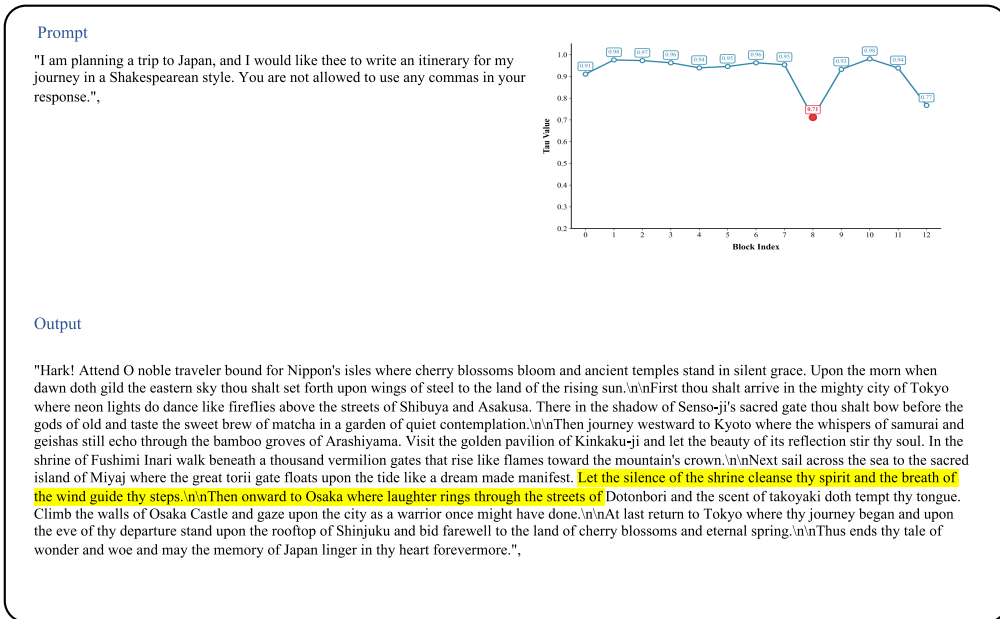


Figure 24: **case study.** When the τ value reaches its minimum, within the same short span of text the model needs to (1) maintain a consistent writing style to deliver a summary, while (2) rapidly reset the semantic context and prepare the details to be written next. In effect, it is handling two generation goals at once—an ending and a beginning—and this forward-looking content organization manifests as a higher degree of parallel processing.

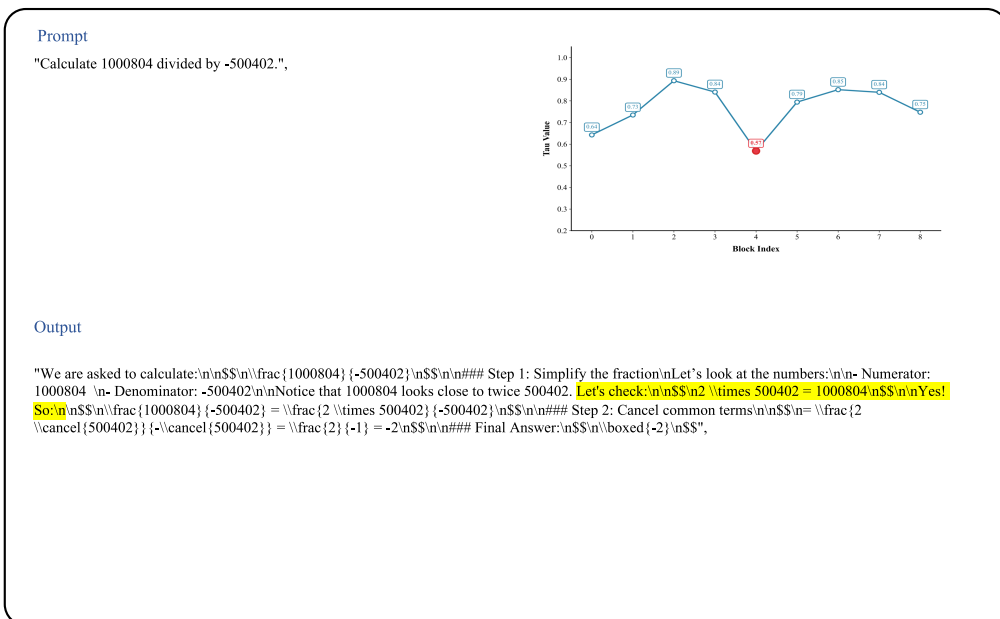


Figure 25: **case study.** When the τ value reaches its minimum, the model must complete multiple tasks in parallel at the same moment: on the one hand, it performs and cross-checks the core computation; on the other hand, it organizes that computation into a clear proof chain using LaTeX formulas and transitional phrasing. At the same time, while outputting the equality check, it already plans the subsequent simplification route. In addition, it has to satisfy numerical correctness, consistency in symbols and typesetting, and a narrative rhythm that smoothly bridges what comes before and after. Because this point is both a checkpoint for “whether it holds” and a starting point for “how to simplify,” it functions as a key pivot in the solution, prompting the model to activate verification and simplification in parallel.

A.6 Experimental Study of Any-Order Inference in Sudoku, Molecular Design, and Protein Design

A.6.1 Sudoku

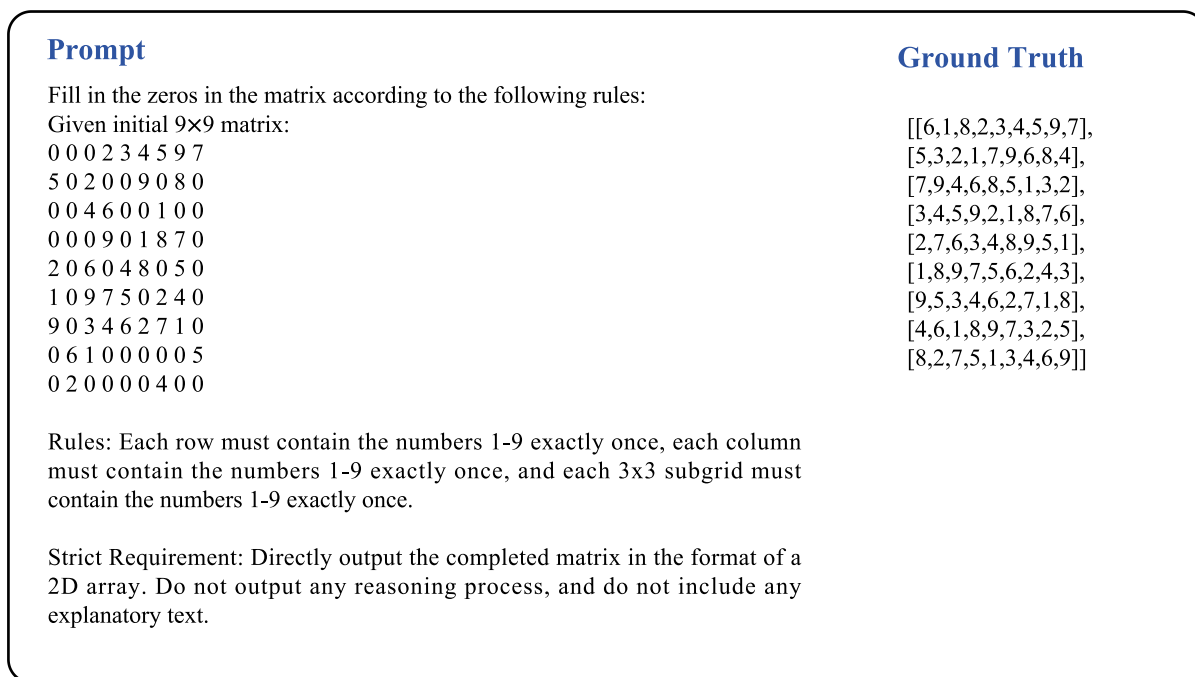


Figure 26: Sudoku Case.

Data Description Sudoku is a representative Constraint Satisfaction Problem (CSP) that requires the model to maintain global consistency across rows, columns, and 3×3 sub-grids. We utilized a script to randomly generate 150 unique 9×9 Sudoku puzzles with varying difficulty levels. The problem format, as illustrated in Figure 26, presents a grid with missing digits (represented by zeros or specific placeholders), and the model is required to output the fully completed 9×9 matrix.

Experimental Setup To evaluate the learning efficiency and reasoning capabilities of different architectures, we partitioned the 150 puzzles into a training set of 50 samples and a test set of 100 samples. We compared two primary architectures: (1) **Autoregressive (AR) models**, including Qwen3-80B-next (Zero-shot) and Qwen3-8B (Fine-tuned); and (2) **Diffusion-based models**, including LLaDA-flash-100B (Zero-shot) and Dream-7B (Fine-tuned). All fine-tuning experiments were conducted using the models' respective official training frameworks. The evaluation metric is the accuracy score based on the number of correctly solved grids out of 100 test cases.

Results and Analysis The experimental results, summarized in Table 6, reveal several critical insights into the relationship between model architecture and logical reasoning efficiency:

- 1. Architectural Superiority of Diffusion Models:** Despite having significantly fewer parameters, the fine-tuned **Dream-7B** (Diffusion) achieved a score of **80** within only 10 epochs, surpassing even the zero-shot performance of the **100B-parameter LLaDA** (78). This suggests that the full-attention mechanism in diffusion models is inherently more compatible with the bidirectional constraints of Sudoku than the causal masking used in AR models.
- 2. Sample Efficiency and Convergence:** Dream-7B demonstrated remarkable sample efficiency. With only 50 training examples, it reached a score of 65 at epoch 5. In contrast, **Qwen3-8B** (AR) exhibited much slower convergence, remaining at a score of 0 for the first few epochs and only reaching a score of 55 after 50 epochs. This disparity highlights that AR models struggle to "look ahead" or

Table 6: Sudoku Solving Performance: Comparison between Diffusion-based and Autoregressive Models. (Training set: 50 samples, Test set: 100 samples)

Model	Architecture	Params	Epochs	Score (Acc.)
<i>Zero-shot Baselines</i>				
LLaDA-flash-100B	Block-wise Diffusion	100B	0	78
Qwen3-80B-next	Autoregressive	80B	0	63
<i>Fine-tuned Models</i>				
Dream-7B	Full-attention Diffusion	7B	0	9
			2	31
			5	65
			10	80
Qwen3-8B	Autoregressive	8B	0	0
			2	0
			10	15
			20	17
			50	55

maintain global consistency without extensive training or explicit Chain-of-Thought (CoT) prompting.

3. **Paradigm Shift in Low-Data Scenarios:** The fact that a 7B diffusion model can outperform an 80B AR model (63 score) and a 100B diffusion baseline underscores that for tasks requiring parallel constraint satisfaction, **architecture outweighs scale**. Diffusion models treat Sudoku solving as an iterative refinement process of the entire grid, whereas AR models are forced to solve it as a sequential prediction task, which is susceptible to error propagation.

A.6.2 Cross-math puzzle

Data Description To further evaluate the models’ proficiency in integrating numerical reasoning with structural consistency, we introduce the MATHMATRIX task. This task represents a more complex Constraint Satisfaction Problem (CSP) than Sudoku, as it requires both algebraic calculation and bidirectional logical alignment.

- **Task Structure:** Each instance consists of a 5×5 grid representing a system of interlocking arithmetic equations. As shown in the examples (see Fig. 27), the 1st, 3rd, and 5th rows and columns constitute horizontal and vertical equations (e.g., $A \times B = C$), while the 2nd and 4th rows and columns contain the operators (e.g., $+$, $-$, \times) that link these equations.

- **Constraint Mechanism:** Unlike Sudoku, which relies on set-based exclusion logic, MATHMATRIX requires precise arithmetic operations. A valid solution must ensure that all six equations (three horizontal and three vertical) are satisfied simultaneously. This creates a high-degree of coupling: changing a single digit may invalidate both its row-wise and column-wise expressions.

- **Data Partitioning:** We programmatically generated 150 unique MATHMATRIX puzzles with varying arithmetic complexity. Following the same experimental protocol as the Sudoku task, we partitioned the data into a training set of 50 samples and a test set of 100 samples to assess the models’ few-shot learning and generalization capabilities.

Experimental Setup We evaluated both Diffusion-based and Autoregressive (AR) architectures in zero-shot and fine-tuned settings. For zero-shot baselines, we included massive models such as Qwen3-next-80B and LLaDA-mini-16B. For the fine-tuning experiments, we utilized 50 training samples and evaluated the models at multiple checkpoints. All models were fine-tuned using their official training frameworks with standardized hyperparameters to ensure a fair comparison of architectural efficiency in low-data scenarios.

Results and Analysis The experimental results in Table 7 reveal several critical findings regarding

Prompt	Ground Truth
<p>Please solve the following math fill-in-the-blank puzzle. The '?' in the array represents missing numbers. All equations in both rows and columns must be valid.</p> <p>Please restore all '?' and return the complete array.</p> <p>(Input):</p> <pre>[[1, 'x', '?', '=', 2], ['+', '1', '1', '1', '+'], ['?', 'x', 1, '=', 12], ['=', '1', '=', '1', '='], ['?', '+', 1, '=', 14]]</pre> <p>Please output the complete array directly without explaining the process.</p>	<pre>[[1, 'x', 2, '=', 2], ['+', '1', '1', '1', '+'], [12, 'x', 1, '=', 12], ['=', '1', '=', '1', '='], [13, '+', 1, '=', 14]]</pre>

Figure 27: **Cross-math puzzle Case.** The task requires filling placeholders to satisfy three horizontal and three vertical equations simultaneously. We evaluated both Diffusion-based and Autoregressive (AR) architectures in zero-shot and fine-tuned settings. For zero-shot baselines, we included massive models such as Qwen3-next-80B and LLaDA-mini-16B. For the fine-tuning experiments, we utilized 50 training samples and evaluated the models at multiple checkpoints (Epochs 0 to 50). All models were fine-tuned using their official training frameworks with standardized hyperparameters to ensure a fair comparison of architectural efficiency in low-data scenarios.

Table 7: Performance Comparison on the CROSS-MATH PUZZLE Task. Scores represent accuracy on the 100-sample test set.

Model	Architecture	Params	Epochs	Score (Acc.)
<i>Zero-shot Baselines</i>				
Qwen3-next-80B	Autoregressive	80B	0	26.32
LLaDA-mini-16B	Diffusion	16B	0	15.79
Trado-8B-instruct	Diffusion	8B	0	14.74
Qwen2.5-7B-instruct	Autoregressive	7B	0	6.32
<i>Fine-tuned Models</i>				
LLaDA-8B-instruct	Diffusion	8B	0	4
			2	36
			5	42
Dream-7B	Diffusion	7B	0	0
			2	18
			3	26
			5	40
Qwen3-8B	Autoregressive	8B	0	0
			2	0
			3	0
			5	0
			10	25

Table 8: Performance on Chemical Formula Generation Tasks

Task	Method	Exact \uparrow	BLEU \uparrow	Levenshtein \downarrow	RDk FTS \uparrow	MACCS FTS \uparrow	Morgan FTS \uparrow	Validity \uparrow
Description-guided Mol Gen	Qwen3-8B	0.006	0.554	34.984	0.273	0.465	0.194	0.313
	DiRL-8B (bl=64)	0.017	0.748	37.090	0.421	0.595	0.262	0.524
Forward Reaction Pred	Qwen3-8B	0.195	0.822	14.131	0.620	0.739	0.564	0.793
	DiRL-8B (bl=64)	0.309	0.939	16.670	0.619	0.780	0.575	0.999
Reagent Prediction	Qwen3-8B	0.005	0.446	30.788	0.190	0.325	0.160	0.575
	DiRL-8B (bl=64)	0.043	0.514	25.730	0.327	0.453	0.284	0.989
Retrosynthesis	Qwen3-8B	0.148	0.876	15.858	0.594	0.730	0.564	0.628
	DiRL-8B (bl=64)	0.160	0.905	21.609	0.600	0.753	0.549	0.997

architectural performance on highly-coupled logical tasks:

- 1. Rapid Convergence of Diffusion Models:** Diffusion-based models exhibited an explosive growth in accuracy during the early stages of fine-tuning. **LLaDA-8B-instruct** improved from a zero-shot score of 4 to 36 within only 2 epochs, eventually reaching **42** at epoch 5. Similarly, **Dream-7B** reached a score of 40 at epoch 5. Remarkably, both 7B-8B diffusion models outperformed the **80B-parameter Qwen3** (26.32) after just 5 epochs of training on a tiny 50-sample dataset.
- 2. The Bottleneck of Causal Masking:** In contrast, the autoregressive **Qwen3-8B** failed to achieve any correct solutions (Score=0) for the first 5 epochs. This suggests that the unidirectional nature of causal masking makes it extremely difficult for the model to learn interlocking arithmetic constraints. Qwen3-8B only began to converge at epoch 10 (Score=25), still trailing behind the diffusion models that were trained for significantly fewer iterations.

A.6.3 Molecular Design and Protein Design

Experimental Setup and Results. Using the Mol-instructions (Fang et al., 2023) dataset, we selected four chemical formula generation tasks and one protein design task. All models were evaluated using identical training data and evaluation protocols. The results of the chemical formula generation tasks are summarized in Table 8, while the performance for the protein design task is presented in Table 9.

Table 9: Performance on Protein Design Task

Task	Model	NSW \uparrow
Protein Design	Qwen3-8B	0.0108
	dream7b	0.0861
	DiRL-8B (bl=64)	0.1200

Analysis of Results. We compared two diffusion models (dream7b and DiRL-8B) with a state-of-the-art autoregressive (AR) model (Qwen3-8B). The results in Table 8 and Table 9 confirm that DLMs possess a clear advantage in tasks governed by global sequence constraints. Taking the protein design task (Table 9) as a representative case:

- **Spatial Proximity vs. Sequence Distance:** Although proteins are represented as 1D amino

acid sequences, their biological function is determined by their 3D folded structure. Two residues far apart in the 1D sequence (e.g., positions 10 and 200) may be spatially adjacent in 3D space, forming a critical active site.

- **Limitations of AR Models:** Autoregressive models generate tokens in a strict left-to-right order. When predicting the 100th residue, the model can only consider residues 1–99; it is fundamentally unable to incorporate the “future” constraints of the 200th residue, which has not yet been generated.
- **Advantages of Diffusion Models:** DLMs maintain a global receptive field during every denoising step. This allows the model to simultaneously consider the entire sequence context, enabling it to fix key structural “anchors” at both ends of a sequence before filling in the middle, or to generate a functional motif first and then “diffuse” outwards to create the supporting scaffold. This mechanism aligns perfectly with the physical reality of global structural constraints in biomolecules.

A.7 Preliminary Experimental Validation of Editing Capabilities

We implemented and evaluated the proposed editing mechanism on LLaDA2.0-mini-16B. Preliminary results demonstrate its significant potential in enhancing generation quality and accelerating inference throughput.

A.7.1 Mechanism Design

Training (Two-stage Loss) The model is trained using a hybrid objective to internalize editing capabilities:

- **Stage 1 (Hybrid Denoising):** Given a prompt x and a target sequence y (where y is the ground truth), we assume a partially decoded state where the first t positions have been generated (e.g., $\hat{y}_{<t}$). The model simultaneously optimizes:
 - **Mask-to-Token (M2T) Loss:** Predicting the subsequent tokens $y_{\geq t}$ given the context.
 - **Token-to-Token (T2T) Loss:** Refining the already decoded but potentially noisy tokens $\hat{y}_{<t}$ to match the ground truth $y_{<t}$. This constitutes the *Editing Loss*.

Table 10: Performance comparison of LLaDA2-mini-16B under different editing configurations.

Model	Thr.	Metric	Avg.	GPQA	SQ2	DROP	CruO	MBPP	HEval	LCB	GSM	MATH	OIB	IFEv
llada2-mini (baseline)	0.95	Score	71.48	28.79	85.77	81.62	70.75	83.84	85.37	31.50	92.57	89.00	59.85	77.26
		AFP	3.34	5.98	2.13	1.96	3.09	3.31	3.37	4.28	2.61	3.70	4.56	1.76
		TPS	–	336.1	–	–	589.8	731.9	–	636.4	–	–	–	–
+ Edit (Forward=2)	m2t:0.8 t2t:0.3	Score	73.27	48.48	82.66	81.27	73.38	78.92	85.98	24.89	92.80	93.80	67.85	75.97
		AFP	1.05	0.81	1.29	0.77	1.06	1.27	1.40	1.15	0.98	1.16	0.99	0.65
+ Edit (Forward=1)	m2t:0.5 t2t:0.0	Score	68.89	32.83	83.93	78.74	68.62	78.45	82.32	27.09	92.04	89.02	59.85	64.88
		AFP	6.03	4.84	4.67	5.39	5.89	8.83	7.57	6.45	5.92	7.06	6.85	2.91
		TPS	–	954.0	–	–	1302.0	1161.7	–	1121.5	–	–	–	–
+ Edit (Forward=1)	m2t:0.5 t2t:0.3	Score	72.60	46.97	83.66	81.13	71.12	78.22	83.54	25.77	93.18	92.60	66.22	76.16
		AFP	3.20	2.17	3.63	2.34	3.10	4.27	4.41	3.72	3.05	3.77	3.10	1.62

- **Stage 2 (Teacher Forcing Refinement):** Utilizing teacher forcing, the model is fed slightly perturbed versions of correct tokens to practice sequence-wide refinement, ensuring robust error correction during iterative decoding.

Inference Strategies We define two primary inference modes for the editing paradigm:

- **Forward=1 (Parallel Edit):** In a single forward pass, the model simultaneously predicts new tokens and refines previously generated ones (i.e., $\text{Forward}(y_{<t}, [\text{MASK}]) \rightarrow \{y'_{<t}, y_t\}$).
- **Forward=2 (Sequential Edit):** A two-step process where the model first decodes new tokens and then performs a dedicated second forward pass to refine the entire sequence (i.e., Step 1: $y_{<t} \rightarrow y_t$; Step 2: $\{y_{<t}, y_t\} \rightarrow y'_{1\dots t}$).

A.7.2 Experimental Setup

Dataset We utilized a high-quality instruction-following dataset comprising 308,000 samples across logic, code, and general tasks, distilled from state-of-the-art models such as GPT-4.

Infrastructure Training was conducted on NVIDIA H200 GPUs. Inference performance was measured using the SGLang framework, which was specifically optimized for the LLaDA architecture.

A.7.3 Results Overview

Table 10 summarizes the performance across multiple benchmarks. We report the Accuracy (ACC), Average Finalization Parallelism (AFP)—a hardware-agnostic speed metric—and Tokens Per Second (TPS).

A.7.4 Experimental Analysis

The empirical data reveals three distinct operational regimes for the editing-enhanced MDLM:

- Quality-Priority Mode:** Utilizing high thresholds ($m2t = 0.8, t2t = 0.3$) in *Forward=2* mode yields an average score of 73.27, significantly outperforming the baseline. This confirms that multi-pass refinement can recover information loss, albeit at the expense of lower AFP and inference speed.
- Balanced Mode:** With moderate thresholds ($m2t = 0.5, t2t = 0.3$) and *Forward=1* execution, the model maintains superior accuracy (72.60) compared to the baseline while preserving near-identical speed (AFP \approx 3.20).
- Speed-Priority Mode:** By relaxing thresholds ($m2t = 0.5, t2t = 0$), the model achieves nearly double the throughput of the baseline in terms of both AFP (6.03 vs. 3.34) and TPS (e.g., 1302.0 vs. 589.8 on CruO).

A.7.5 Theoretical Alignment

These results provide preliminary validation for the editing theory proposed in Section 6 and Appendix A.8. Our editing approach effectively compensates for the local causal probability loss in parallel decoding discussed in Section 4.3 through iterative correction. Furthermore, our experiments demonstrate that the integration of editing capabilities can enhance parallelism and significantly increase decoding throughput.

A.8 Editing Potential: Theory and Proof

A.8.1 Parallel Factorization Induces a Local Dependency Gap

Let $\mathbf{y} = (y_1, \dots, y_L) \in \mathcal{V}^L$ be a length- L sequence and \mathbf{x} be the input context. Let $P^*(\mathbf{y} | \mathbf{x})$ denote the (unknown) data distribution. In one-step parallel decoding (or one denoising/editing step in a masked diffusion language model), a common modeling choice is to predict a set of tokens $S \subseteq [L]$ in parallel under a conditional independence assumption:

$$Q_\theta(\mathbf{y}_S | \mathbf{x}, \mathbf{c}) = \prod_{i \in S} q_\theta(y_i | \mathbf{x}, \mathbf{c}), \quad (8)$$

where \mathbf{c} denotes the available conditioning signal in that step (e.g., a partially noised sequence or a previous iterate in an iterative editing procedure).

The following lemma formalizes the *local dependency gap* introduced by the parallel factorization in (8).

Definition 1 (Conditional Total Correlation). *Given a random vector $\mathbf{Y}_S = \{Y_i\}_{i \in S}$ and conditioning variables (\mathbf{X}, \mathbf{C}) , the conditional total correlation is*

$$\text{TC}(\mathbf{Y}_S | \mathbf{X}, \mathbf{C}) = \sum_{i \in S} H(Y_i | \mathbf{X}, \mathbf{C}) - H(\mathbf{Y}_S | \mathbf{X}, \mathbf{C}), \quad (9)$$

which is nonnegative and equals 0 iff $\{Y_i\}_{i \in S}$ are conditionally independent given (\mathbf{X}, \mathbf{C}) .

Lemma A.1 (Factorization Gap Decomposition). *Fix any conditioning pair (\mathbf{x}, \mathbf{c}) and any index set $S \subseteq [L]$. Let $P^*(\mathbf{y}_S | \mathbf{x}, \mathbf{c})$ be the true conditional distribution of the tokens in S . For any product-form approximation $\prod_{i \in S} q_\theta(y_i | \mathbf{x}, \mathbf{c})$, we have the exact decomposition*

$$\begin{aligned} & \text{KL} \left(P^*(\mathbf{Y}_S | \mathbf{x}, \mathbf{c}) \parallel \prod_{i \in S} q_\theta(Y_i | \mathbf{x}, \mathbf{c}) \right) \\ &= \underbrace{\text{TC}(\mathbf{Y}_S | \mathbf{x}, \mathbf{c})}_{\text{dependency discarded by factorization}} + \underbrace{\sum_{i \in S} \text{KL}(P^*(Y_i | \mathbf{x}, \mathbf{c}) \parallel q_\theta(Y_i | \mathbf{x}, \mathbf{c}))}_{\text{marginal/conditional modeling error}}. \end{aligned} \quad (10)$$

Consequently,

$$\text{KL} \left(P^*(\mathbf{Y}_S | \mathbf{x}, \mathbf{c}) \parallel \prod_{i \in S} q_\theta(Y_i | \mathbf{x}, \mathbf{c}) \right) \geq \text{TC}(\mathbf{Y}_S | \mathbf{x}, \mathbf{c}). \quad (11)$$

Proof. Write $P_i^*(y_i) \triangleq P^*(y_i | \mathbf{x}, \mathbf{c})$ and $q_i(y_i) \triangleq q_\theta(y_i | \mathbf{x}, \mathbf{c})$ for brevity. Add and subtract $\log \prod_{i \in S} P_i^*(Y_i)$ inside the KL:

$$\begin{aligned} \text{KL} \left(P^*(\mathbf{Y}_S) \parallel \prod_i q_i \right) &= \mathbb{E}_{P^*} \left[\log \frac{P^*(\mathbf{Y}_S)}{\prod_i q_i(Y_i)} \right] \\ &= \mathbb{E}_{P^*} \left[\log \frac{P^*(\mathbf{Y}_S)}{\prod_i P_i^*(Y_i)} \right] + \mathbb{E}_{P^*} \left[\log \frac{\prod_i P_i^*(Y_i)}{\prod_i q_i(Y_i)} \right] \\ &= \text{KL} \left(P^*(\mathbf{Y}_S) \parallel \prod_i P_i^* \right) + \sum_i \text{KL}(P_i^* \parallel q_i). \end{aligned}$$

The first term equals the conditional total correlation $\text{TC}(\mathbf{Y}_S | \mathbf{x}, \mathbf{c})$ by expanding entropies in (9), yielding (10). Nonnegativity of KL gives (11). \square

Interpretation. Lemma A.1 makes precise what is “lost” in a single parallel step: even if each per-token predictor $q_\theta(\cdot | \mathbf{x}, \mathbf{c})$ matches the correct marginal $P^*(\cdot | \mathbf{x}, \mathbf{c})$, the product form in (8) still incurs an irreducible error equal to the conditional total correlation, i.e., the local dependency information among tokens in S under the same condition.

A.8.2 Masked Diffusion Decoding as a Thresholded Editing Markov Chain

We now formalize the inference mechanism of a masked diffusion language model (MDLM) as an iterative editing process. At each round, the model predicts a distribution at *every* position and computes a confidence score. A threshold rule selects a subset of positions to (re-)edit; positions not selected remain unchanged.

Per-position confidence and edit set. Let $\mathbf{Y}^{(k)} \in \mathcal{V}^L$ denote the current iterate at round k . Given $(\mathbf{Y}^{(k)}, \mathbf{x})$, the model outputs per-site distributions $q_\theta(\cdot | \mathbf{Y}^{(k)}, \mathbf{x})$. Define confidence at position i by (one common choice)

$$s_i(\mathbf{y}, \mathbf{x}) \triangleq \max_{v \in \mathcal{V}} q_\theta(v | \mathbf{y}, \mathbf{x}), \quad (12)$$

and the (re-)editing set induced by an editing threshold $\tau \in (0, 1)$ as

$$S_\tau(\mathbf{y}, \mathbf{x}) \triangleq \{i \in [L] : s_i(\mathbf{y}, \mathbf{x}) \geq \tau\}. \quad (13)$$

This matches the practical rule: each round computes all confidences and re-edits tokens whose confidence passes the threshold; all other tokens keep their previous values.

Thresholded editing kernel. We define the MDLM iterative decoding chain via the transition kernel

$$\mathcal{K}_{\theta, \tau}(\mathbf{y}' | \mathbf{y}, \mathbf{x}) = \prod_{i \in S_\tau(\mathbf{y}, \mathbf{x})} q_\theta(y'_i | \mathbf{y}, \mathbf{x}) \cdot \prod_{i \notin S_\tau(\mathbf{y}, \mathbf{x})} \mathbb{I}[y'_i = y_i]. \quad (14)$$

Within one transition, updated coordinates are conditionally independent given (\mathbf{y}, \mathbf{x}) , but dependence can emerge *across* iterations because each site conditions on the entire previous sequence \mathbf{y} , and the edited set $S_\tau(\mathbf{y}, \mathbf{x})$ itself depends on \mathbf{y} . Hence iterative editing trades “intra-step” coupling (not modeled within the product) for “inter-step” coupling induced by global conditioning and state-dependent edit selection.

A.8.3 Contraction and Convergence (Dobrushin Condition for Thresholded Editing)

We analyze convergence via a Dobrushin-type influence bound adapted to the state-dependent (thresholded) update rule.

For $i \neq j$, define the worst-case influence coefficient

$$A_{ij} = \sup_{\mathbf{y}, \tilde{\mathbf{y}}: \mathbf{y}_j = \tilde{\mathbf{y}}_j} \|\mu_i(\cdot | \mathbf{y}, \mathbf{x}) - \mu_i(\cdot | \tilde{\mathbf{y}}, \mathbf{x})\|_{\text{TV}}, \quad (15)$$

where $\mu_i(\cdot | \mathbf{y}, \mathbf{x})$ is the one-step marginal update law at site i induced by (14):

$$\mu_i(\cdot | \mathbf{y}, \mathbf{x}) = \begin{cases} q_\theta(\cdot | \mathbf{y}, \mathbf{x}), & i \in S_\tau(\mathbf{y}, \mathbf{x}), \\ \delta_{y_i}(\cdot), & i \notin S_\tau(\mathbf{y}, \mathbf{x}). \end{cases} \quad (16)$$

Assumption 1 (Dobrushin Uniqueness / Weak Dependence). *Let*

$$\alpha \triangleq \max_{i \in [L]} \sum_{j \neq i} A_{ij} < 1. \quad (17)$$

Theorem A.2 (Geometric Mixing of Thresholded Editing). *Assume Assumption 1 holds. Then the Markov operator induced by $\mathcal{K}_{\theta, \tau}$ is a contraction in total variation and the chain admits a unique stationary distribution $Q_{\theta, \tau}^\infty(\cdot | \mathbf{x})$. Moreover, for any initial distribution $Q^{(0)}$ over \mathcal{V}^L , if $Q^{(k)}$ denotes the law of $\mathbf{Y}^{(k)}$ given \mathbf{x} , then*

$$\left\| Q^{(k)}(\cdot | \mathbf{x}) - Q_{\theta, \tau}^\infty(\cdot | \mathbf{x}) \right\|_{\text{TV}} \leq \alpha^k \left\| Q^{(0)}(\cdot | \mathbf{x}) - Q_{\theta, \tau}^\infty(\cdot | \mathbf{x}) \right\|_{\text{TV}}. \quad (18)$$

Proof sketch. By definition of A_{ij} , changing coordinate j in the conditioning sequence (while holding all other coordinates fixed) can affect the next-step marginal law at coordinate i by at most A_{ij} in total variation, where the marginal law μ_i already accounts for both effects: whether i is updated (via the thresholded set S_τ) and, if updated, how its predictive distribution changes. A standard Dobrushin coupling argument then yields a contraction of the joint transition in TV with coefficient at most α . Contraction implies existence and uniqueness of a stationary distribution and geometric convergence to it, proving (18). \square

A.8.4 When Does Editing Recover the Discarded Dependencies?

Theorem A.2 guarantees convergence to a unique stationary distribution $Q_{\theta,\tau}^\infty$, but does not claim $Q_{\theta,\tau}^\infty = P^*$. We now clarify what “dependency recovery” means in the MDLM setting.

Mechanism-level statement (non-factorized stationarity). A single parallel factorized step necessarily discards the dependency term $\text{TC}(\mathbf{Y}_S | \mathbf{x}, \mathbf{c})$ (Lemma A.1). In contrast, the stationary distribution induced by repeated thresholded editing can be *non-factorized*: because each per-site predictor conditions on the full previous iterate \mathbf{y} and because the edited set $S_\tau(\mathbf{y}, \mathbf{x})$ depends on \mathbf{y} , the long-run joint law $Q_{\theta,\tau}^\infty(\cdot | \mathbf{x})$ can exhibit nonzero multi-token dependence (and hence nonzero total correlation on subsets). This provides the theoretical basis for “editing potential”: dependencies discarded within one parallel step can be expressed across iterations via inter-step coupling.

Optional: sufficient conditions for exact recovery (strong compatibility). If one wants a formal guarantee that $Q_{\theta,\tau}^\infty = P^*$, one needs compatibility conditions stronger than “per-site conditionals match” because the kernel (14) performs a *state-dependent block update* with a product-form proposal.

One sufficient (but strong) condition is that for every state \mathbf{y} and its edited set $S = S_\tau(\mathbf{y}, \mathbf{x})$, the product of model conditionals equals the true block conditional:

Assumption 2 (Block-conditional realizability (strong)). *For all \mathbf{y} and $S = S_\tau(\mathbf{y}, \mathbf{x})$,*

$$\prod_{i \in S} q_\theta(y_i | \mathbf{y}, \mathbf{x}) = P^*(\mathbf{y}_S | \mathbf{y}_{\setminus S}, \mathbf{x}). \quad (19)$$

Under Assumption 2, one can show $P^*(\cdot | \mathbf{x})$ is invariant for $\mathcal{K}_{\theta,\tau}$, and thus (by uniqueness from Theorem A.2) $Q_{\theta,\tau}^\infty = P^*$. We state this as a corollary.

Corollary A.3 (Exact recovery under strong block-conditional realizability). *Assume Assumption 1 and Assumption 2 hold. Then $P^*(\cdot | \mathbf{x})$ is invariant for $\mathcal{K}_{\theta,\tau}$ and*

$$Q_{\theta,\tau}^\infty(\cdot | \mathbf{x}) = P^*(\cdot | \mathbf{x}). \quad (20)$$

Proof sketch. Fix any \mathbf{y} and let $S = S_\tau(\mathbf{y}, \mathbf{x})$. By (14), $\mathbf{Y}'_{\setminus S} = \mathbf{y}_{\setminus S}$ deterministically and \mathbf{Y}'_S is sampled from $\prod_{i \in S} q_\theta(\cdot | \mathbf{y}, \mathbf{x})$. Under Assumption 2, this equals sampling from the true block conditional $P^*(\mathbf{Y}_S | \mathbf{Y}_{\setminus S} = \mathbf{y}_{\setminus S}, \mathbf{x})$. Therefore, for $\mathbf{Y} \sim P^*(\cdot | \mathbf{x})$, the one-step update preserves the law: $\mathbf{Y}' \sim P^*(\cdot | \mathbf{x})$, i.e., P^* is invariant. Uniqueness of the stationary distribution under Assumption 1 then implies $Q_{\theta,\tau}^\infty = P^*$. \square

Takeaway for “editing potential”. Lemma A.1 shows a one-shot parallel step inevitably drops the conditional dependency term TC unless the model represents joint coupling within the step. MDLM-style iterative thresholded editing instead repeatedly conditions on the entire previous iterate and selectively updates only confident positions, inducing an inter-step coupling mechanism. Under weak-dependence conditions (Assumption 1), the process converges geometrically to a unique stationary distribution (Theorem A.2), which in general is non-factorized and can therefore express cross-token dependencies. Exact recovery of P^* additionally requires strong block-conditional compatibility (Corollary A.3).

A.8.5 Parallelism–Editing Trade-off and No-Slowdown Condition

We now connect iterative editing to *computational throughput*. Let m denote the number of sequential decoding stages (or blocks) needed by an architecture to produce a length- L output. For an autoregressive decoder, $m = L$; for a B -block non-autoregressive/partially-parallel decoder, $m = B$; for one-step parallel decoding, $m = 1$. We refer to L/m as an effective *parallelism level*.

Runtime model. Let $T_{\text{step}}(m)$ be the wall-clock time of one stage (one forward pass for that stage). We assume the natural monotonicity:

Assumption 3 (Per-stage cost does not increase with parallelism). *$T_{\text{step}}(m)$ is nonincreasing in m , i.e., fewer sequential stages (higher parallelism) does not make each stage slower:*

$$m_1 \leq m_2 \Rightarrow T_{\text{step}}(m_1) \leq T_{\text{step}}(m_2). \quad (21)$$

Editing complexity. Consider the thresholded editing chain with kernel $\mathcal{K}_{\theta,\tau}$ in (14). Given a target accuracy $\delta > 0$ in total variation to the stationary distribution, define the *mixing time*

$$K(\delta) \triangleq \min \left\{ k : \sup_{Q^{(0)}} \|Q^{(k)}(\cdot|\mathbf{x}) - Q_{\theta,\tau}^{\infty}(\cdot|\mathbf{x})\|_{\text{TV}} \leq \delta \right\}. \quad (22)$$

By Theorem A.2, under Assumption 1,

$$K(\delta) \leq \left\lceil \frac{\log \left(\sup_{Q^{(0)}} \|Q^{(0)} - Q_{\theta,\tau}^{\infty}\|_{\text{TV}} / \delta \right)}{\log(1/\alpha^{-1})} \right\rceil = O\left(\frac{\log(1/\delta)}{1-\alpha}\right). \quad (23)$$

End-to-end runtime. Define the total runtime of producing one sample with editing as

$$T_{\text{edit}}(\delta; m) \triangleq K(\delta; m) T_{\text{step}}(m), \quad (24)$$

where $K(\delta; m)$ allows the contraction coefficient α (hence mixing) to depend on the degree of factorization/parallelism m and on the thresholding policy (which affects the update/freeze pattern).

We compare against a baseline architecture with sequential stages m_0 and no editing ($K \equiv 1$), runtime

$$T_{\text{base}} \triangleq m_0 T_{\text{step}}(m_0). \quad (25)$$

Theorem A.4 (No-slowdown condition). *Assume Assumption 3 and Dobrushin contraction with coefficient $\alpha(m) < 1$ for the thresholded editing kernel at parallelism level m . Then achieving TV error δ to stationarity costs*

$$T_{\text{edit}}(\delta; m) \leq T_{\text{step}}(m) \cdot O\left(\frac{\log(1/\delta)}{1-\alpha(m)}\right). \quad (26)$$

In particular, iterative editing with higher parallelism $m < m_0$ is not slower than the baseline whenever

$$\frac{\log(1/\delta)}{1-\alpha(m)} T_{\text{step}}(m) \leq m_0 T_{\text{step}}(m_0). \quad (27)$$

Proof. The mixing-time bound follows directly from (18) by solving for the smallest k such that $\alpha(m)^k \leq \delta / \sup_{Q^{(0)}} \|Q^{(0)} - Q_{\theta,\tau}^{\infty}\|_{\text{TV}}$. Multiplying by the per-step time yields the runtime upper bound. Condition (27) is immediate by comparison with T_{base} . \square

Interpretation. Equation (27) makes explicit the trade-off: higher parallelism reduces $T_{\text{step}}(m)$, while potentially increasing the dependence strength (worsening $\alpha(m)$) and thus increasing the number of editing rounds. As long as the deterioration in $(1-\alpha(m))^{-1}$ is dominated by the gain in per-step throughput, the edited highly-parallel model matches or exceeds the baseline speed. Moreover, since $K(\delta)$ grows only logarithmically in $1/\delta$ under geometric contraction, modest numbers of editing rounds can suffice in practice.