LED: IMPROVING MASKED DIFFUSION LANGUAGE MODELS BY LOOKAHEAD EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion large language models (dLLMs) have emerged as a promising alternative to autoregressive language models, enabling non-autoregressive text generation through parallel decoding. Existing dLLMs can be broadly categorized into two families: masked diffusion models (MDMs) and uniform diffusion models (UDMs) In this work, we theoretically show that MDMs are easier to optimize due to their mask-based transition schedule but have lower computational expressivity, whereas UDMs enable iterative state revision and greater expressivity but are substantially harder to train. Motivated by this analysis, we propose Lookahead Editing (LED), which augments masked diffusion decoding with a rewritable proposal state. At each decoding step, the model performs several lookahead updates to iteratively refine future predictions in this proposal state before committing tokens to the masked sequence. This design introduces a rewritable proposal for exploring and refining future predictions while preserving the simple mask-based transition schedule of MDMs. We fine-tune Qwen2.5-7B and Qwen3-8B to obtain LED-7B and LED-8B, respectively. Across eight benchmarks, LED SFT yields average gains of $+11.4$ and $+7.7$ points over standard MDM SFT. Moreover, using SFT alone, LED-8B outperforms LLaDA2.1-16B Q-mode, which is trained with continued pre-training, supervised fine-tuning, and reinforcement learning.

1 INTRODUCTION

Auto-regressive models (ARMs) (Achiam et al., 2023; Liu et al., 2024; Dubey et al., 2024) have achieved remarkable performance across a wide range of natural language tasks. However, ARMs are inherently limited by sequential decoding. Since tokens are generated one-by-one in a strict left-to-right order, they cannot fully exploit parallel computation during generation. Diffusion-based language models (dLLMs) (Nie et al., 2025; Zhu et al., 2025; Khanna et al., 2025) provide an alternative formulation of sequence generation that relaxes this fixed ordering and enables parallel denoising over discrete states.

Existing dLLMs can be broadly categorized into two families: masked diffusion models (MDMs) (Nie et al., 2025; Ye et al., 2025) and uniform diffusion models (UDMs) (Lou et al., 2023; Taïga et al., 2026). MDMs introduce a dedicated mask token and generate sequences by progressively replacing masked positions with predicted tokens, while UDMs define transitions over the entire vocabulary, allowing each token to be repeatedly replaced by other vocabulary tokens during denoising. Despite sharing the same diffusion-based formulation, MDMs and UDMs exhibit fundamentally different properties in terms of training difficulty and computational expressivity. Following the path-space ELBO decomposition of Amin et al. (2025) for dLLMs, which separates the objective into a transition prediction loss and a schedule matching loss, we show that UDMs incur an additional model-dependent schedule matching loss because their transition schedule depends on the denoiser. In contrast, the transition schedule in standard MDMs is independent of the denoiser and subsequently the schedule matching term is zero. This simplifies the learning objective and leads to a smaller excess loss than UDMs.

On the other hand, the training advantage of MDMs comes with a limitation in computational expressivity. We first prove that MDMs remain within TC_0 (Vollmer, 1999), a parallel computation model class in which the computational depth remains $O(1)$, even with arbitrary-order prediction and bidirectional attention, sharing the same complexity class with ARMs (Li et al., 2024). UDMs,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

by contrast, can repeatedly rewrite previously generated tokens. We show that this rewritable state enables them to achieve TC_0 and support NC_1 computation, which can represent hierarchical computations of logarithmic depth such as evaluating nested Boolean expressions.

Motivated by the above analysis, we propose Lookahead Editing (LED), which introduces a rewritable proposal to explore and refine future predictions while preserving the simple mask-based transition schedule of MDMs. For model learning, we first train a standard masked diffusion model, which is subsequently used to generate proposal sequences. Given a clean sequence, we randomly mask a subset of tokens to construct a noised sequence. We then concatenate the clean and noised sequences and use the pretrained masked diffusion model to generate a proposal sequence. Finally, we concatenate the clean, noised, and proposal sequences as input to LED and optimize a cross-entropy loss on the outputs corresponding to the noised branch. Since the proposal sequence generated by the pretrained MDM may contain both correct and incorrect tokens, it provides supervision for learning when to preserve correct proposal tokens and when to revise incorrect ones. At inference time, LED iteratively alternates between a proposal stage and a commit stage. The proposal stage maintains a rewritable intermediate state in the proposal branch, allowing tokens to be repeatedly generated and revised over multiple lookahead steps. The refined proposal is then used as additional condition in the commit stage, which follows the standard masked diffusion process: once a token is committed, it is fixed and cannot be rewritten in subsequent decoding steps. Across commit stages, LED preserves reliable proposal tokens while revising erroneous ones to improve subsequent proposals.

We fine-tune Qwen2.5-7B and Qwen3-8B on the same training set using standard MDM SFT and LED SFT. Across eight benchmarks, LED SFT yields average gains of $+11.4$ and $+7.7$ points over standard MDM SFT for Qwen2.5-7B and Qwen3-8B, respectively. Moreover, using SFT alone, LED-8B outperforms LLaDA2.1-16B Q-Mode, which is trained with continued pre-training, supervised fine-tuning, and reinforcement learning.

2 Preliminary

2.1 Diffusion Language Models

Diffusion models have demonstrated remarkable success in continuous data domains (Sohl-Dickstein et al., 2015; Ho et al., 2020; Karras et al., 2022). Building on these foundations, diffusion language models have been developed to train discrete diffusion models directly within the categorical vocabulary space (Sohl-Dickstein et al., 2015; Austin et al., 2021a; Lou et al., 2023; Nie et al., 2025; Ye et al., 2025). Let $\mathcal{V}$ denote a finite vocabulary and $X = (X_t)_{t\in[0,1]}$ denote a Markov trajectory, where $X_t \in \mathcal{V}$ is the state at diffusion time $t \in [0, 1]$. We denote the clean data state by $\boldsymbol{x}^0 = X_0$ and the terminal noisy state by $\boldsymbol{x}^1 = X_1$. The predefined forward noising process progressively corrupts $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$, and the learned *reverse* generative process progressively denoises $\boldsymbol{x}^1$ to $\boldsymbol{x}^0$:

$$p(X_t \mid X_s) = \mathrm{Cat}(X_t; Q_{s,t}X_s)\,, \quad q_\theta(X_s \mid X_t) = \mathrm{Cat}\big(X_s; Q^\theta_{t,s}X_t\big)\,, \quad 0 \le s < t \le 1,$$

where $Q_{s,t}$ is the forward transition kernel from time s to t, inducing the path distribution $p(X \mid \boldsymbol{x}^0)$, and $Q^\theta_{t,s}$ is the model-dependent reverse transition kernel, inducing the reverse path distribution $q_\theta(X)$, where θ is the model parameter. MDMs and UDMs differ in their choice of the forward transition kernel: MDMs use an absorbing transition toward a dedicated mask token m, whereas UDMs transition toward the uniform distribution over the vocabulary (Austin et al., 2021a),

$$Q^{\mathrm{MDM}}_{s,t} = \alpha_{s,t}\mathbf{I} + (1-\alpha_{s,t})\boldsymbol{e}_m\mathbf{1}^\top, \qquad Q^{\mathrm{UDM}}_{s,t} = \alpha_{s,t}\mathbf{I} + (1-\alpha_{s,t})\frac{1}{|\mathcal{V}|}\mathbf{1}\mathbf{1}^\top,$$

where $\boldsymbol{e}_m$ denotes the one-hot vector of the mask token, $\mathbf{I}$ is identity matrix and $\alpha_{s,t}$ controls the noise level between diffusion times s and t. The model parameters are learned by maximizing an evidence lower bound (ELBO) on the data log-likelihood,

$$\mathcal{L}_{\mathrm{ELBO}}(\theta) = \mathbb{E}_{p_{\mathrm{data}}(\boldsymbol{x}^0)}\mathbb{E}_{p(X\mid\boldsymbol{x}^0)}\left[\log\frac{q_\theta(X)}{p(X\mid\boldsymbol{x}^0)}\right] \le \mathbb{E}_{p_{\mathrm{data}}(\boldsymbol{x}^0)}\left[\log q_\theta(\boldsymbol{x}^0)\right]. \tag{1}$$

Instead of corrupting and generating tokens uniformly over the full sequence length, recent SOTA dLLMs adopt block diffusion language models (BDLMs) (Arriola et al., 2025; Wu et al., 2026; Cheng et al., 2025; Bie et al., 2026), which executes discrete diffusion by partitioning the sequence into multiple blocks. Unless otherwise stated, our discussion throughout this paper focuses on BDLMs.

2.2 Circuit Complexity

Circuit complexity (Vollmer, 1999) provides a standard framework for characterizing the computational expressivity of parallel models. A Boolean circuit is a directed acyclic graph composed of logic gates and is characterized primarily by its size, depth, fan-in, and gate types. Recently, circuit complexity has been used to characterize the expressivity of language models (Merrill & Sabharwal, 2023; Liu et al., 2023; Li et al., 2024). They consider three families of circuit classes: NC^k, AC^k, and TC^k, where $k \geq 0$. For an input of length n, NC^k consists of problems computable by polynomial-size, $O((\log n)^k)$-depth circuits with bounded-fan-in AND, OR, and NOT gates. AC^k is defined similarly but allows AND and OR gates with unbounded fan-in, while TC^k further allows unbounded-fan-in MAJORITY gates. These classes satisfy $\mathsf{NC}^k \subseteq \mathsf{AC}^k \subseteq \mathsf{TC}^k \subseteq \mathsf{NC}^{k+1}$.

With polynomially bounded hidden dimension, constant-depth autoregressive transformers without additional intermediate computation are contained in TC^0 under logarithmic numerical precision; and in AC^0 under constant numerical precision. Allowing $O(\log n)$ chain-of-thought steps does not increase these asymptotic upper bounds. The resulting models remain in TC^0 under logarithmic fixed-point precision, and in AC^0 under constant precision (Li et al., 2024).

3 Optimization-Expressivity Trade-off

MDM and UDM exhibit markedly different behaviors in both training and generation. MDM benefits from the explicit mask state, which provides a strong inductive bias for learning, but its generation process is non-rewriting: once a token is committed, it cannot be revised. UDM, in contrast, allows intermediate states to be repeatedly updated, offering a potentially richer computational mechanism, but removes the explicit masking structure that simplifies transition learning. In this Section, we extend the circuit-complexity perspective to study how the decoding dynamics of diffusion language models affect their computational expressivity.

3.1 Masking Eliminates Schedule Learning.

For any path X, let $M(X)$ be the number of state transitions along the trajectory. Define the transition schedule $S(X) = \{t_1, \ldots, t_{M(X)}\}$, where $0 < t_1 < \cdots < t_{M(X)} < 1$, and X changes state at time t_j. Following Amin et al. (2025), the ELBO in Equation (1) can be decomposed as two losses: $\mathcal{L}_{\text{where}}(\theta)$ is about the state transitions (i.e., *where to jump*), while $\mathcal{L}_{\text{when}}(\theta)$ is about the transition schedule (i.e., *when to jump*):

$$\mathcal{L}_{\text{ELBO}}(\theta) = \underbrace{-\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}^0)}\mathbb{E}_{p(X|\boldsymbol{x}^0)}\left[\log \frac{p(X \mid \boldsymbol{x}^0, \boldsymbol{x}^1, S)}{q_\theta(X \mid \boldsymbol{x}^1, S)}\right]}_{\mathcal{L}_{\text{where}}(\theta)} \underbrace{- D_{\text{KL}}\left(p(S) \,\|\, q_\theta(S)\right)}_{\mathcal{L}_{\text{when}}(\theta)} + C, \tag{2}$$

where $C = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}^0)p(S|\boldsymbol{x}^0)}\left[\log \frac{p(S)}{p(S|\boldsymbol{x}^0)}\right] - \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}^0)}\mathbb{E}_{p(S|\boldsymbol{x}^0)}\left[D_{\text{KL}}\left(p(\boldsymbol{x}^1 \mid \boldsymbol{x}^0, S) \,\|\, q(\boldsymbol{x}^1 \mid S)\right)\right]$ is a constant independent of the model parameter θ, and $q_\theta(S)$ is the pushforward distribution of $q_\theta(X)$, i.e. $q_\theta(S) = \int \mathbf{1}\{S(X) \in A\}\, q_\theta(dX) := S_{\#} q_\theta(X)$.

The following Theorems show that the schedule-matching term vanishes for MDMs; while for UDMs it depends on the denoiser through the induced reverse jump rate.

Theorem 3.1. *For MDM, $q_\theta(S) = p(S)$, $\forall\theta$, and thus $\mathcal{L}_{\text{when}}(\theta) = D_{\text{KL}}(p(S)\|q_\theta(S)) = 0$.*

Theorem 3.2. *For UDMs, let $\tau = 1 - t$, $Y_\tau = X_{1-\tau}$, and $\mathcal{F}^S_{\tau^-} := \{\tau_i \in S : \tau_i < \tau\}$. The schedule-matching loss is $\mathcal{L}_{\text{when}}(\theta) = \mathbb{E}_{S,t}\left[\mu^*(\tau) s \log \frac{\mu^*(\tau)}{\bar{\mu}_\theta(\tau|\mathcal{F}^S_{\tau^-})} - \mu^*(\tau) + \bar{\mu}_\theta(\tau \mid \mathcal{F}^S_{\tau^-})\right]$, where $\mu^*(\tau) = \frac{V-1}{V}\beta_{1-\tau}$, $\beta_t = -\frac{d}{dt}\log\alpha_t$, $V = |\mathcal{V}|$, $\bar{\mu}_\theta(\tau \mid \mathcal{F}^S_{\tau^-}) = \mathbb{E}_{q_\theta}\left[\lambda_{f_\theta}(Y_\tau, 1-\tau) \mid \mathcal{F}^S_{\tau^-}\right]$. and $\lambda_{f_\theta}(x,t) = \frac{\beta_t}{V}\left[f_\theta(x \mid x,t)(1-\alpha_t)/(1/(V-1)+\alpha_t) + \big(1 - f_\theta(x \mid x,t)\big)(V-1+\alpha_t)/(1-\alpha_t)\right]$.*

The where loss for MDMs and UDMs are implemented using the same cross-entropy (CE) objective (Bie et al., 2026; Taïga et al., 2026), while the when loss is implemented using the following tractable schedule-matching objective $\mathbb{E}_{S,\tau,Y_\tau}\left[\mu^*(\tau)\log\frac{\mu^*(\tau)}{\lambda_{f_\theta}(Y_\tau,1-\tau)} - \mu^*(\tau) + \lambda_{f_\theta}(Y_\tau, 1-\tau)\right]$

(Campbell et al., 2022). Let MDM and UDM both use SGD with the same learning rate η, as $\theta_{k+1} = \theta_k - \eta(\nabla\mathcal{L}(\theta_k) + \xi_k)$. The following Theorem shows that with the same where loss, UDMs is harder to train in terms of admitting a smaller learning rate and resulting in a larger loss.

Theorem 3.3. *Assume that the gradient noise is unbiased (i.e., $\mathbb{E}[\xi_k \mid \theta_k] = 0$) and the model is sufficiently expressive to represent the data distribution, i.e. $q_{\theta^\star} = p_{\text{data}}$. MDM admits a larger learning rate than UDM near the optimum under second order Taylor approximation, i.e., $\eta_{\max}^{MDM} = 2/\lambda_{\max}(H_M) \geq \eta_{\max}^{UDM} = 2/\lambda_{\max}(H_U)$. Moreover, for learning rate satisfying $0 < \eta < 2/\lambda_{\max}(H_U)$, the expected excess losses satisfy $\mathbb{E}\left[\mathcal{L}_U - \mathcal{L}_U^\star\right] \geq \mathbb{E}\left[\mathcal{L}_M - \mathcal{L}_M^\star\right]$. Since $\mathcal{L}_U^\star \geq \mathcal{L}_M^\star$, it follows that $\mathbb{E}[\mathcal{L}_U] - \mathbb{E}[\mathcal{L}_M] \geq \mathcal{L}_U^\star - \mathcal{L}_M^\star \geq 0$.*

3.2 Rewriting improves Expressivity.

The optimization advantage of masking, however, comes with a structural restriction. Once a position is unmasked and committed, standard masked diffusion does not revise it in subsequent decoding steps. We next study the computational consequence of this monotone generation process.

Following Li et al. (2024), we define $\mathsf{T}_{\mathrm{ARM}}[T(n), d(n), s(n), e(n)]$ as the function class of autoregressive transformers, where the four parameters specify, respectively, the number of CoT steps, the hidden dimension, the numerical precision in bits, and the number of exponent bits. Li et al. (2024) show that autoregressive transformers with polynomial hidden dimension, $O(\log n)$-bit numerical precision, and fixed-point arithmetic remain contained in TC^0 even with $O(\log n)$ CoT steps.

Theorem 3.4 (Li et al. (2024))**.** $\mathsf{T}_{\mathrm{ARM}}[0, \mathrm{poly}(n), \log n, 0] \subseteq \mathsf{T}_{\mathrm{ARM}}[\log n, \mathrm{poly}(n), \log n, 0] \subseteq \mathsf{TC}^0$.

Denote the function class of MDMs by $\mathsf{T}_{\mathrm{MDM}}[T(n), d(n), s(n), e(n), B(n)]$. The first four parameters are the same as $\mathsf{T}_{\mathrm{ARM}}$, Compared with autoregressive models, MDMs introduce two additional sources of decoding flexibility: bidirectional attention within each block and arbitrary-order generation. The following Theorem shows that, for a constant block size, these additional mechanisms are still insufficient to move the model beyond TC^0.

Theorem 3.5. $\mathsf{T}_{\mathrm{MDM}}[0, \mathrm{poly}(n), \log n, 0, O(1)] \subseteq \mathsf{T}_{\mathrm{MDM}}[\log n, \mathrm{poly}(n), \log n, 0, O(1)] \subseteq \mathsf{TC}^0$.

$\mathsf{T}_{\mathrm{MDM}}[0, \mathrm{poly}(n), \log n, 0, O(1)]$ above isolates the effect of bidirectional attention by removing CoT, while $\mathsf{T}_{\mathrm{MDM}}[\log n, \mathrm{poly}(n), \log n, 0, O(1)]$ permits $O(\log n)$ CoT steps with arbitrary-order generation. However, they are still insufficient to provide a fundamentally stronger computational resource than autoregressive decoding.

The situation changes when previously generated states can be repeatedly rewritten. Let $\mathsf{T}_{\mathrm{UDM}}[T(n), d(n), s(n), e(n), W(n)]$ be the function class of UDMs, where the first four parameters are the same as $\mathsf{T}_{\mathrm{ARM}}$ and $\mathsf{T}_{\mathrm{MDM}}$, while the last parameter $W(n)$ is the size of the rewritable workspace (i.e., CoT length). UDMs allow polynomially many revisions even with a constant-size workspace. The following Theorem shows that with polynomially many revisions over a constant workspace, UDMs can achieve NC^1.

Theorem 3.6. $\mathsf{NC}^1 \subseteq \mathsf{T}_{\mathrm{UDM}}[\mathrm{poly}(n), \mathrm{poly}(n), \log n, 0, O(1)]$.

4 Proposed Method: Lookahead Editing

While UDMs offer strong expressivity through iterative state revision, their denoiser-dependent transition schedules hinder optimization. MDMs benefit from simple, schedule-free optimization but suffer from constrained expressivity due to non-rewritable decoding. To bridge this gap, we propose Lookahead Editing (`LED`), a simple yet effective framework that introduces a rewritable proposal state into masked diffusion decoding. We detail its training and inference procedures below.

4.1 Training

Traditional MDMs adopt a two-branch optimization framework consisting of a masked branch and a clean branch (Wu et al., 2026; Cheng et al., 2025; Fu et al., 2026). To enable editing capabilities in MDMs, we add a proposal branch that yields both correct and erroneous tokens, allowing the

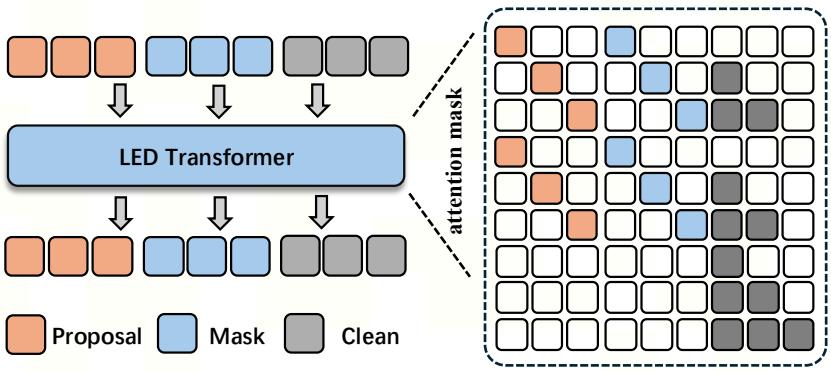

Figure 1: Training of Lookahead Editing. The proposal, masked, and clean branches are concatenated and processed by the LED Transformer in a single forward pass. The structured attention mask controls information flow across the three branches, where rows denote query tokens and columns denote key/value tokens. Within each proposal mask block pair, the two branches attend bidirectionally to each other and to all preceding clean blocks, while the clean branch follows block-causal attention.

Algorithm 1 LED Training.

Input: clean sequence $\mathtt{X}_0$, forward noise process $q(\cdot \mid \mathtt{X}_0, t)$, pretrained model $f_{\mathtt{MDM}}$, model $f_{\mathtt{LED}}$, loss $\mathcal{L}_{\mathrm{CE}}$
repeat
  Sample $t \sim \mathcal{U}[0,1]$
  $\mathtt{X}_t \sim p(\cdot \mid \mathtt{X}_0, t)$
  $\widehat{\mathtt{X}}_0 \leftarrow f_{\mathtt{MDM}}([\mathtt{X}_0, \mathtt{X}_t])$
  $\mathtt{X}_{\mathrm{logit}} \leftarrow f_{\mathtt{LED}}([\mathtt{X}_0, \mathtt{X}_t, \widehat{\mathtt{X}}_0])$
  $\mathtt{X}_{\mathrm{logit}} \leftarrow \mathtt{X}_{\mathrm{logit}}[\mathtt{X}_t]$
  Take gradient step on $\nabla_\theta \mathcal{L}_{\mathrm{CE}}(\mathtt{X}_{\mathrm{logit}}; \mathtt{X}_0)$
until converged

Algorithm 2 LED Inference.

Input: model $f_{\mathtt{LED}}$, lookahead steps K, sampler algorithm $\mathcal{S}$
$\mathtt{X}^{\mathtt{T}}, \mathtt{LX}^{\mathtt{t0}}, \mathtt{C}^{\mathtt{T}} \leftarrow [\mathtt{MASK}]^L$
for $t = T$ **to** 1 **do**
  $\mathtt{LC}^{\mathtt{t1}} \leftarrow f_{\mathtt{LED}}(\mathtt{p}, \mathtt{X}^t, \mathtt{C}^{\mathtt{t}})$ ▷ Step 1
  $\mathtt{LX}^{\mathtt{t1}}, \mathtt{LC}^{\mathtt{t1}} \leftarrow \mathcal{S}(\mathtt{LO}^{\mathtt{t0}}), \mathtt{LO}^{\mathtt{t0}}$
  for $k = 1$ **to** K **do** ▷ Step 2
    $\mathtt{LO}^{\mathtt{tk}} \leftarrow f_{\mathtt{LED}}(\mathtt{p}, \mathtt{LX}^{\mathtt{tk}}, \mathtt{LC}^{\mathtt{tk}})$
    $\mathtt{LX}^{\mathtt{t,k+1}}, \mathtt{LC}^{\mathtt{t,k+1}} \leftarrow \mathcal{S}(\mathtt{LO}^{\mathtt{tk}}), \mathtt{LO}^{\mathtt{tk}}$
  $\mathtt{C}^t \leftarrow f_{\mathtt{LED}}(\mathtt{p}, \mathtt{X}^t, \mathtt{LC}^{\mathtt{t,K+1}})$ ▷ Step 3
  $\mathtt{X}^{t-1}, \mathtt{C}^{t-1} \leftarrow \mathcal{S}(\mathtt{C}^t), \mathtt{C}^t$
return $\mathtt{X}^0$

Figure 2: Training and inference procedures of Lookahead Editing. During training, a pretrained MDM generates the proposal, and LED is trained on the resulting clean, noised, and proposal branches. During inference, LED performs K iterative lookahead updates before each actual decoding step while retaining the original mask-based sampling schedule.

model to identify erroneous tokens and subsequently correct them while retaining correct tokens. The pseudocode is shown in Algorithm 1. For each training sequence, we construct a three-branch input consisting of: (i) a clean branch representing the ground-truth sequence $\mathtt{X}_0$; (ii) a masked branch $\mathtt{X}_t$, obtained by applying standard masking (Arriola et al., 2025) to $\mathtt{X}_0$; and (iii) a proposal branch $\widehat{\mathtt{X}}_0$, generated by a pretrained masked block diffusion model f_{MDM} conditioned on the concatenated clean and masked branches $[\mathtt{X}_0, \mathtt{X}_t]$. We concatenate the clean branch $\mathtt{X}_0$, masked branch $\mathtt{X}_t$, and proposal branch $\widehat{\mathtt{X}}_0$ as input to $f_{\mathtt{LED}}$. The attention mask is illustrated in Figure 1. Within each mask–proposal block pair, the two blocks attend bidirectionally to each other as well as to all preceding clean blocks, whereas attention within the clean branch remains strictly block-causal. This design enables bidirectional interactions between mask and proposal blocks, while preventing information leakage from future clean tokens. The model outputs $\mathtt{X}_{\mathrm{logit}}$ corresponding to the masked branch are optimized via a cross-entropy loss against the clean target sequence $\mathtt{X}_0$. Because the proposal tokens generated by f_{MDM} are a mix of correct predictions and errors, they provide natural supervision for editing. Specifically, the loss encourages the model to selectively retain accurate proposal tokens while updating inaccurate ones.

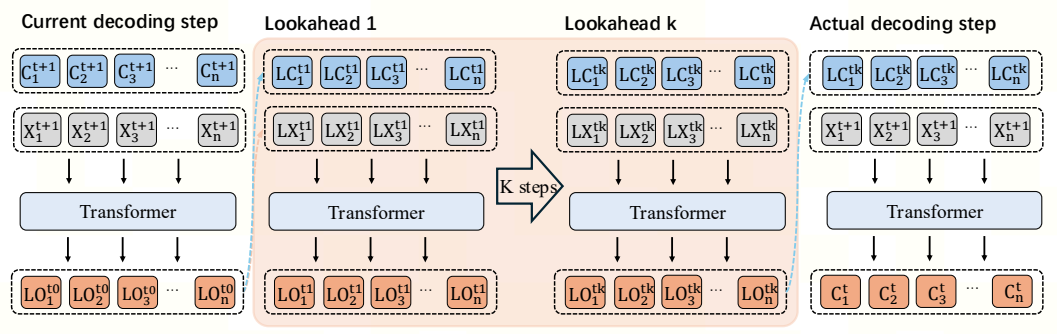

Figure 3: Lookahead Editing decoding process. Orange arrow denotes and blue arrow denotes sample. At each decoding step, LED performs K lookahead updates on the rewritable proposal state, allowing future predictions to be iteratively explored and refined. The resulting proposal is then used to perform the actual decoding update and advance the sequence to the next decoding step.

4.2 Inference

The pseudocode and schematic overview of our inference procedure are provided in Algorithm 2 and Figure 3, respectively. At decoding step $\mathtt{t}$, inference maintains three sequence components: prompt $\mathtt{p}$, a partially masked sequence $\mathtt{X}^{\mathtt{t}+1}$, and a rewritable proposal sequence $\mathtt{C}^{\mathtt{t}+1}$. Rather than immediately committing tokens from a single-step prediction, f_{LED} performs Lookahead Editing to iteratively refine future predictions prior to commitment:

1. Initial proposal generation: At the beginning of step $\mathtt{t}$, the current proposal $\mathtt{C}^{\mathtt{T}}$ and masked sequence $\mathtt{X}^{\mathtt{T}}$ are initialized as fully masked sequences and processed by f_{LED}. This yields an initial prediction $\mathtt{LO}^{\mathtt{t}_0}$, which is copied to $\mathtt{LC}^{\mathtt{t}1}$ and serves as the base proposal for the subsequent lookahead iterations.
2. Lookahead proposal refinement: As illustrated in Figure 3, for each lookahead step $\mathtt{k} \in \{1, \ldots, K\}$, the proposal block $\mathtt{LC}^{\mathtt{tk}}$ is updated based on the model's output from the preceding step. Passing $\mathtt{LX}^{\mathtt{tk}}$ and $\mathtt{LC}^{\mathtt{tk}}$ into f_{LED} yields updated predictions $\mathtt{LO}^{\mathtt{tk}}$. These predictions serve as the proposal sequence $\mathtt{LC}^{\mathtt{tk}+1}$ for the subsequent lookahead step, while high-confidence tokens are sampled by sampler $\mathcal{S}$ into the temporary $\mathtt{LX}^{\mathtt{tk}+1}$. This allows the draft proposal to be iteratively rewritten and explored across K steps.
3. Decoding commitment: After K lookahead steps, the highly refined proposal $\mathtt{LC}^{\mathtt{t},\mathtt{K}+1}$ is combined with $\mathtt{X}^{\mathtt{t}-1}$ to compute token confidence scores. The model then commits a subset of high-confidence tokens in $\mathtt{C}^{\mathtt{t}}$ to get $\mathtt{X}^{\mathtt{t}-1}$, advancing the sequence to the next decoding step.

Decoding alternates between rewritable lookahead drafting and irreversible token commitment until the current block is fully decoded, after which it is appended to $\mathtt{p}$.

5 Experiments

5.1 Experiment Setup

Training Set We construct the training corpus for SFT by combining several curated open-source post-training datasets, including the NVIDIA OpenMathReasoning (Moshkov et al., 2025), NVIDIA OpenScience (NVIDIA), NVIDIA OpenCodeReasoning2 (Ahmad et al., 2025), OpenThoughts3 (Guha et al., 2026), and Mixture-of-Thoughts (open r1).

Benchmarks We evaluate the models on a comprehensive suite of tasks covering diverse aspects of reasoning abilities. The evaluation suite includes (i) mathematical reasoning tasks: MATH (Saxton & Hill, 2019), AIME 2024 (Jia, 2024) and AIME 2025 (math ai, 2025); (ii) scientific reasoning: GPQA (Rein et al., 2024); and (iii) code generation tasks: HumanEval, HumanEval+ (Chen et al., 2021), MBPP, and MBPP+ (Austin et al., 2021b). All benchmarks are evaluated in a zero-shot setting.

Table 1: Results on mathematical reasoning, scientific reasoning, and code generation benchmarks. Models are grouped by their post-training recipes. For LED, the green superscripts indicate the absolute improvements over the corresponding MDM-SFT baseline with the same model size. Avg. denotes the average performance across all eight benchmarks. The best result in each column is shown in **bold**, and the second-best result is underlined.

Model	Math			Science	Code				Avg.
	MATH500	AIME24	AIME25	GPQA	HumanEval	HumanEval+	MBPP	MBPP+	
CPT+SFT+RL									
LLaDA 2.1 16B	**92.8**	**50.0**	36.7	50.5	84.8	80.5	**80.4**	**67.2**	67.9
CPT + SFT									
SDAR 8B	68.8	10.0	6.7	33.3	73.8	68.9	75.4	62.2	49.9
Nemotron 8B	86.6	20.0	26.7	39.4	82.3	79.3	77.0	64.6	59.5
Dream 7B	39.2	0.0	0.0	26.8	48.8	43.9	67.7	55.8	35.3
SFT									
XDLM 8B	30.2	3.3	0.0	29.8	28.0	23.8	43.4	34.4	24.1
Fast-dLLM 7B	58.0	0.0	6.7	31.9	63.4	58.5	63.0	52.3	41.7
MDM-SFT 7B	72.0	13.3	13.3	22.7	62.8	59.8	64.0	55.0	45.4
LED 7B	86.0$_{+14.0}$	26.7$_{+13.3}$	23.3$_{+10.0}$	33.8$_{+11.1}$	75.0$_{+12.2}$	70.7$_{+10.9}$	74.6$_{+10.6}$	64.0$_{+9.0}$	56.8
MDM-SFT 8B	88.8	36.7	23.3	46.5	79.9	74.4	74.6	61.9	60.8
LED 8B	92.6$_{+3.8}$	**50.0**$_{+13.3}$	**40.0**$_{+16.7}$	**53.6**$_{+7.1}$	**87.2**$_{+7.3}$	**84.1**$_{+9.7}$	75.4$_{+0.8}$	65.1$_{+3.2}$	**68.5**

We report accuracy on MATH, AIME 2024, AIME 2025, and GPQA, and Pass@1 on HumanEval, HumanEval+, MBPP, and MBPP+.

Baseline We compare LED against 7B–16B models, including MDMs such as Fast-dLLM (Wu et al., 2026), Dream (Ye et al., 2025), Nemotron (Fu et al., 2026), and SDAR (Cheng et al., 2025), as well as hybrid MDM–UDM models such as LLaDA 2.1 (Bie et al., 2026) and XDLM (Liu et al., 2026). All models use confidence-based sampler with a threshold of 0.9.

5.2 Comparison with Existing dLLMs

We initialize our models from Qwen2.5-7B and Qwen3-8B and perform LED SFT. All models are trained with a block size of 32, a global batch size of 512, a sequence length of 16,384, and a constant learning rate of 10^{-5} with Adam optimizer. Compared with SFT-only baselines, including Fast-dLLM (41.7 average accuracy) and XDLM (24.1), LED-8B achieves an average score of 68.5, outperforming them by $+26.8$ and $+44.4$ points, respectively. We further compare LED with dLLMs trained using more extensive post-training pipelines. Among models trained with continued pre-training (CPT) followed by SFT, Nemotron-8B achieves the highest average score of 59.5, while SDAR-8B and Dream-7B achieve 49.9 and 35.3, respectively. With SFT alone, LED-8B achieves an average score of 68.5, outperforming Nemotron-8B by $+9.0$ points. Moreover, LED-8B also surpasses LLaDA2.1-16B, which is trained with CPT, SFT, and RL, by $+0.6$ points on average. These show that LED is highly competitive despite using SFT alone.

Moreover, we also train LED and the standard MDM SFT (Arriola et al., 2025) on the same dataset under identical training settings for a controlled comparison. As shown in Table 1, LED consistently improves over the corresponding MDM-SFT baseline across all eight benchmarks and at both model scales. For the 7B model, LED increases the average score from 45.4 to 56.8, corresponding to an absolute improvement of $+11.4$ points. The gains are consistent across mathematical reasoning, scientific reasoning, and code generation, with the largest improvement of $+13.4$ points on AIME 2024. A similar trend is observed for the 8B model, where LED improves the average score from 60.8 to 68.5, yielding a $+7.7$-point gain. In particular, LED achieves a substantial improvement of $+16.7$ points on AIME 2025. These results demonstrate that introducing rewritable lookahead proposals consistently improves masked diffusion fine-tuning across different model scales and task domains.

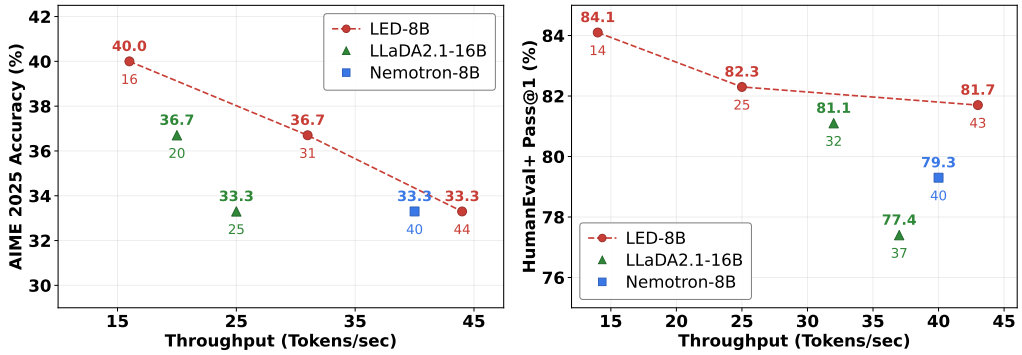

Figure 4: Speed–accuracy trade-off of LED-8B, LLaDA2.1-16B, and Nemotron-8B on AIME 2025 and HumanEval+ dataset. For LED-8B, different operating points are obtained by varying the number of lookahead steps. For LLaDA2.1-16B, we report its Q and S modes. LED-8B achieves a stronger Pareto frontier across both benchmarks.

5.3 Speed-Accuracy Trade-off

Figure 4 compares the speed-accuracy trade-off of LED-8B against LLaDA2.1-16B and Nemotron-8B on AIME 2025 and HumanEval+ datasets. For LED-8B, we vary the number of lookahead steps from 0 to 2, yielding different operating points along the accuracy–throughput Pareto front. For LLaDA2.1-16B, we report results for both its Q and S modes, which are designed to prioritize generation quality and speed, respectively. For Nemotron-8B, we report the accuracy obtained using greedy threshold decoding.

On AIME 2025 dataset, LED-8B achieves accuracy–throughput pairs of $(40.0, 16)$, $(36.7, 31)$, and $(33.3, 44)$, where throughput is measured in tokens per second. In comparison, LLaDA2.1-16B achieves $(36.7, 20)$ and $(33.3, 25)$ in its Q and S modes, respectively. At matched accuracy, LED-8B consistently achieves substantially higher throughput: at 36.7% accuracy, it reaches 31 tokens/s compared with 20 tokens/s for LLaDA2.1-16B ($1.55\times$ faster), while at 33.3% accuracy, it reaches 44 tokens/s compared with 25 tokens/s ($1.76\times$ faster). Nemotron-8B achieves its highest accuracy 33.3% at 40 tokens/s, which is also dominated by LED-8B, which attains the same accuracy at 44 tokens/s.

A similar trend is observed on HumanEval+. LED-8B achieves $(84.1, 14)$, $(82.3, 25)$, and $(81.7, 43)$, whereas LLaDA2.1-16B achieves $(81.1, 32)$ and $(77.4, 37)$. In particular, at its high-throughput operating point, LED-8B achieves both higher accuracy and higher throughput than LLaDA2.1-16B: 81.7% at 43 tokens/s versus 81.1% at 32 tokens/s. It also Pareto-dominates Nemotron-8B, which achieves 79.3% at 40 tokens/s. Overall, these results show that lookahead provides LED with a flexible mechanism for trading generation speed for accuracy, while yielding a stronger speed–accuracy Pareto frontier than the compared diffusion language models across both mathematical reasoning and code generation.

5.4 Hyper-parameter Analysis

We study the effect of two key hyper-parameters in LED: proposal threshold τ and number of lookahead steps K. Figure 5 reports the results on AIME 2025 and HumanEval+. Overall, the two hyper-parameters jointly control how aggressively the model exploits its lookahead proposals, and we find that a moderate lookahead times combined with an appropriate proposal threshold yields the best performance.

Proposal Threshold. The proposal threshold τ controls the acceptance of lookahead proposals. At a fixed number of lookahead iterations, the performance varies noticeably with τ, indicating that appropriately filtering proposal tokens is important. For $K = 2$, increasing the threshold from $\tau = 0.6$ to $\tau = 0.7$ substantially improves AIME 2025 accuracy from 33.3% to 43.3%, while HumanEval+ increases from 79.9% to 84.1%. Further increasing the threshold does not consistently improve performance: on AIME 2025, $\tau = 0.8$ and $\tau = 0.9$ both achieve approximately 40.0%, This

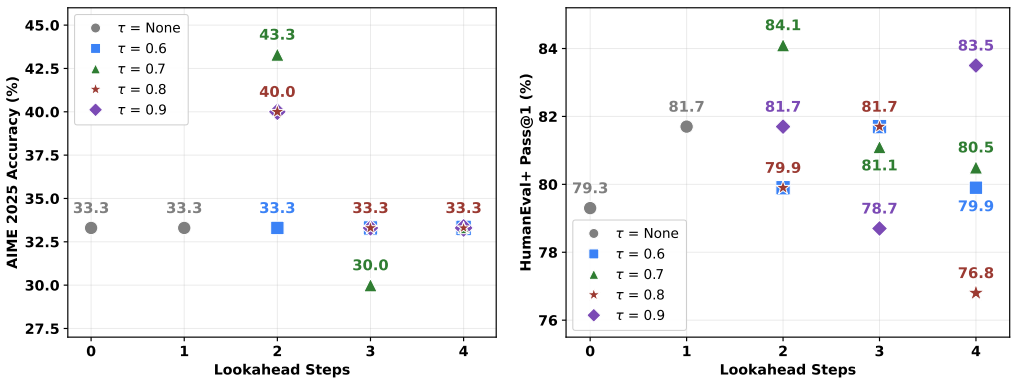

Figure 5: Hyper-parameter analysis of the number of lookahead steps K and proposal threshold τ on AIME 2025 and HumanEval+ dataset. A moderate number of lookahead steps ($K = 2$) generally yields the strongest performance, with $\tau = 0.7$ achieving the best results on AIME 2025 and HumanEval+ benchmarks.

suggests that an overly aggressive or overly conservative proposal selection criterion can both be suboptimal, and that an intermediate threshold provides a better balance between exploiting useful proposals and filtering unreliable ones. Among the configurations shown, $\tau = 0.7$ with $K = 2$ achieves the best performance on both benchmarks.

Number of Lookahead Steps K. Without sufficiently deep lookahead, the model obtains relatively limited gains: on AIME 2025, the accuracy remains at 33.3% for $K = 0$ and $K = 1$. Increasing the look ahead times to $K = 2$ leads to a substantial improvement, reaching up to 43.3% on AIME 2025 and 84.1% on HumanEval+. However, further increasing K does not consistently yield additional benefits. On AIME 2025, performance drops considerably for $K = 3$ and $K = 4$ across different threshold choices, suggesting that excessively long lookaheads may accumulate inaccurate proposals and reduce their usefulness for subsequent generation. A similar, although less pronounced, trend is observed on HumanEval+. Interestingly, with a conservative threshold of $\tau = 0.9$, $K = 4$ still achieves 83.5%, indicating that longer lookahead can remain effective when low-confidence proposals are sufficiently filtered. Overall, these results suggest that a moderate look ahead times, particularly $K = 2$, provides a robust balance between obtaining informative future context and avoiding error accumulation from excessively long proposal lookaheads.

6 Conclusion

In this work, we analyzed the trade-off between optimization simplicity and computational expressivity in diffusion language models. Our theoretical analysis reveals that: masked diffusion models are easier to optimize due to their simple mask-based transition schedules but have limited iterative revision capability, whereas uniform diffusion models achieve greater computational expressivity through iterative state revision at the cost of more challenging optimization. Motivated by this trade-off, we propose Lookahead Editing (`LED`), which augments masked diffusion decoding with a rewritable proposal state. By repeatedly refining future predictions in the proposal state before committing tokens to the masked sequence, `LED` introduces iterative revision capability while preserving the simple transition structure of masked diffusion. Across eight reasoning and code-generation benchmarks, `LED` consistently improves over standard MDM fine-tuning, with `LED-8B` further outperforming the more extensively post-trained LLaDA2.1-16B while achieving a stronger speed–accuracy Pareto front than existing dLLM baselines. Overall, our results demonstrate that `LED` effectively bridges the gap between the optimization simplicity of masked diffusion and the iterative refinement capability of more expressive diffusion processes.

AI USE STATEMENT

In this work, we used generative AI tools solely to aid and polish the writing, including improving grammar, wording, and readability. We did not use generative AI tools for research ideation or execution, retrieval or discovery of related work, drafting substantive content, generating synthetic datasets, or proving mathematical claims. All AI-assisted edits were reviewed and verified by the authors. We take full responsibility for the final content of this work.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *Preprint arXiv:2303.08774*, 2023.

Wasi Uddin Ahmad, Somshubra Majumdar, Aleksander Ficek, Sean Narenthiran, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Vahid Noroozi, and Boris Ginsburg. Opencodereasoning-ii: A simple test time scaling approach via self-critique. *Preprint arXiv:2507.09075*, 2025.

Alan Amin, Nate Gruver, and Andrew Wilson. Why masking diffusion works: Condition on the jump schedule for improved discrete diffusion. *In NeurIPS*, 2025.

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *Preprint arXiv:2503.09573*, 2025.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *In NeurIPS*, 2021a.

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

Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, et al. Llada2. 1: Speeding up text diffusion via token editing. *Preprint arXiv:2602.08676*, 2026.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *In NeurIPS*, 2022.

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, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *Preprint arXiv:2107.03374*, 2021.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *Preprint arXiv:2510.06303*, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *Preprint arXiv:2407.21783*, 2024.

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

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. In *ICLR*, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *In NeurIPS*, 2020.

Maxwell Jia. Aime problem set 2024, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *In NeurIPS*, 2022.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, et al. Mercury: Ultra-fast language models based on diffusion. *Preprint arXiv:2506.17298*, 2025.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *ICLR*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *Preprint arXiv:2412.19437*, 2024.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *In ICLR*, 2023.

Yue Liu, Yuzhong Zhao, Zheyong Xie, Qixiang Ye, Jianbin Jiao, Yao Hu, Shaosheng Cao, and Yunfan Liu. Balancing understanding and generation in discrete diffusion models. In *ICML*, 2026.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *Preprint arXiv:2310.16834*, 2023.

math ai. Aime problem set 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *In TACL*, 2023.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. *Preprint arXiv:2504.16891*, 2025.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *NeurIPS*, 2025.

NVIDIA. Openscience: A multi-domain synthetic dataset for general-purpose reasoning. URL https://huggingface.co/datasets/nvidia/OpenScience.

open r1. Mixture-of-thoughts. URL https://huggingface.co/datasets/open-r1/Mixture-of-Thoughts.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *COLM*, 2024.

Grefenstette Saxton and Kohli Hill. Analysing mathematical reasoning abilities of neural models. *Preprint arXiv:1904.01557*, 2019.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

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

Heribert Vollmer. *Introduction to circuit complexity: a uniform approach*. Springer Science & Business Media, 1999.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion llm. In *ICLR*, 2026.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *Preprint arXiv:2508.15487*, 2025.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. Llada 1.5: Variance-reduced preference optimization for large language diffusion models. In *NeurIPS*, 2025.

APPENDIX

A PROOF OF THEOREM 3.1

Proof. For a masked diffusion model, let $S(X) := \inf\{t \in [0,1] : X_t = m\}$ denote the transition time from a clean token to the absorbing mask state m. We have

$$p(S(X) \le t) = p(X_t = m \mid X_0 = x) = 1 - \alpha_t. \tag{3}$$

Hence, the cumulative distribution function and the corresponding probability density function of the forward transition schedule are

$$F_p(t) = 1 - \alpha_t, \quad p_S(t) = \frac{d}{dt} F_p(t) = -\dot{\alpha}_t. \tag{4}$$

We next derive the transition-time distribution under the reverse process. For $0 \le s < t \le 1$, by Bayes' rule,

$$\begin{aligned} p(X_s = m \mid X_t = m, X_0 = x) &= \frac{p(X_t = m \mid X_s = m, X_0 = x) p(X_s = m \mid X_0 = x)}{p(X_t = m \mid X_0 = x)} \\ &= \frac{1 - \alpha_s}{1 - \alpha_t}, \end{aligned} \tag{5}$$

The standard MDM reverse transition is obtained by marginalizing this exact posterior over the model prediction of the clean token:

$$\begin{aligned} q_\theta(X_s = m \mid X_t = m) &= \sum_x p(X_s = m \mid X_t = m, X_0 = x) f_\theta(x \mid X_t, t) \\ &= \sum_x \frac{1 - \alpha_s}{1 - \alpha_t} f_\theta(x \mid X_t, t) \\ &= \frac{1 - \alpha_s}{1 - \alpha_t} \sum_x f_\theta(x \mid X_t, t) \\ &= \frac{1 - \alpha_s}{1 - \alpha_t}, \end{aligned} \tag{6}$$

Since $\alpha_1 = 0$, we have:

$$p(X_1 = m \mid X_0 = x) = 1 - \alpha_1 = 1. \tag{7}$$

Setting the later time in Equation (6) to 1 therefore gives

$$q_\theta(X_t = m \mid X_1 = m) = \frac{1 - \alpha_t}{1 - \alpha_1} = 1 - \alpha_t. \tag{8}$$

Under the reverse path distribution, the same deterministic map $S(X)$ associates each trajectory with its transition time. Since $S(X) \le t$ if and only if $X_t = m$, we have

$$q_\theta(S(X) \le t) = q_\theta(X_t = m \mid X_1 = m) = 1 - \alpha_t. \tag{9}$$

Thus, the cumulative distribution function and probability density function of the reverse transition schedule are

$$F_{q_\theta}(t) = 1 - \alpha_t, q_{\theta,S}(t) \quad \frac{d}{dt} F_{q_\theta}(t) = -\dot{\alpha}_t. \tag{10}$$

Comparing Equations (4) and (10), we have $p_S(t) = q_{\theta,S}(t) = -\dot{\alpha}_t, \ \forall\, t \in [0,1]$. Since the two schedule distributions have the same density, the pushforward distributions induced by $S(X)$ under the forward and reverse path distributions coincide: $p(S) = q_\theta(S)$. Consequently,

$$\mathcal{L}_{\text{when}}(\theta) = D_{\text{KL}}\left(p(S) \,\|\, q_\theta(S)\right) = 0. \tag{11}$$

□

B PROOF OF THEOREM 3.2

For a trajectory X in UDMs, define its transition schedule as

$$S(X) = \{t_1, \dots, t_M\}, \qquad 0 < t_1 < \cdots < t_M < 1,$$

where each t_i satisfies $X_{t_i^-} \neq X_{t_i}$. The UDM forward kernel is $Q_{s,t} = \alpha_{s,t} I + (1 - \alpha_{s,t}) \frac{1}{V} \mathbf{1}\mathbf{1}^\top$, where $\alpha_t := \alpha_{0,t}$, $\alpha_{s,t} = \frac{\alpha_t}{\alpha_s}$. Define the forward noise rate $\beta_t := -\frac{d}{dt} \log \alpha_t$. It follows that $\dot{\alpha}_t = -\beta_t \alpha_t$. For an infinitesimal interval of length $h > 0$,

$$\alpha_{t,t+h} = \frac{\alpha_{t+h}}{\alpha_t} = \frac{\alpha_t + \dot{\alpha}_t h + o(h)}{\alpha_t} = 1 - \beta_t h + o(h).$$

For any $y \neq x$,

$$\Pr(X_{t+h} = y \mid X_t = x) = \frac{1 - \alpha_{t,t+h}}{V} = \frac{\beta_t}{V} h + o(h).$$

Therefore, the forward jump rate is $r_t(x, y) = \frac{\beta_t}{V}, y \neq x$. Summing over all possible destination states gives the total forward jump rate

$$\lambda^*(t) = \sum_{y \neq x} r_t(x, y) = \frac{V - 1}{V} \beta_t.$$

We now derive the induced schedule density. Let $R(a, b)$ denote the probability that no transition occurs on $(a, b]$. By the definition of the total jump rate,

$$R(a, b + h) = R(a, b) \left[1 - \lambda^*(b) h + o(h)\right].$$

Therefore,

$$\frac{\partial}{\partial b} R(a, b) = -\lambda^*(b) R(a, b),$$

with initial condition $R(a, a) = 1$. Solving this differential equation gives

$$R(a, b) = \exp\left(-\int_a^b \lambda^*(u)\, du\right).$$

Consequently, for a schedule $S = \{t_1, \dots, t_M\},\ 0 < t_1 < \cdots < t_M < 1$, its density is

$$\begin{aligned} p(S) &= R(0, t_1) \lambda^*(t_1) R(t_1, t_2) \lambda^*(t_2) \cdots R(t_{M-1}, t_M) \lambda^*(t_M) R(t_M, 1) \\ &= \exp\left(-\int_0^1 \lambda^*(t)\, dt\right) \prod_{i=1}^{M} \lambda^*(t_i). \end{aligned}$$

We first derive the reverse jump rate induced by the denoiser. For $y \neq x$, Bayes' rule gives

$$\begin{aligned} &\Pr(X_{t-h} = y \mid X_t = x, X_0 = x_0) \\ &= \frac{\Pr(X_t = x \mid X_{t-h} = y, X_0 = x_0) \Pr(X_{t-h} = y \mid X_0 = x_0)}{\Pr(X_t = x \mid X_0 = x_0)}. \end{aligned}$$

By the Markov property,

$$\Pr(X_t = x \mid X_{t-h} = y, X_0 = x_0) = \Pr(X_t = x \mid X_{t-h} = y) = \frac{\beta_t}{V} h + o(h), \tag{12}$$

while

$$\Pr(X_{t-h} = y \mid X_0 = x_0) = \alpha_{t-h} \mathbf{1}\{y = x_0\} + \frac{1 - \alpha_{t-h}}{V} \tag{13}$$

$$= (\alpha_t - h\dot{\alpha}_t + o(h))\, \mathbf{1}\{y = x_0\} + \frac{1 - \alpha_t + h\dot{\alpha}_t + o(h)}{V} \tag{14}$$

$$= \Pr(X_t = y \mid X_0 = x_0) - h\dot{\alpha}_t \left(\mathbf{1}\{y = x_0\} - \frac{1}{V}\right) + o(h) \tag{15}$$

Hence

$$\Pr(X_{t-h} = y \mid X_t = x, X_0 = x_0) = \frac{\beta_t}{V} \frac{\Pr(X_t = y \mid X_0 = x_0)}{\Pr(X_t = x \mid X_0 = x_0)} h + o(h).$$

Therefore, the exact reverse rate conditioned on $X_0 = x_0$ is

$$r_t^{\leftarrow}(x, y \mid x_0) = \frac{\beta_t}{V} \frac{\Pr(X_t = y \mid X_0 = x_0)}{\Pr(X_t = x \mid X_0 = x_0)}.$$

The reverse model does not observe X_0. Let $f_\theta(x_0 \mid x, t)$ denote the denoiser prediction of the clean-state posterior given $X_t = x$. Marginalizing over x_0 gives the model reverse rate:

$$\begin{aligned} r_t^\theta(x, y) &= \sum_{x_0 \in \mathcal{V}} f_\theta(x_0 \mid x, t) r_t^{\leftarrow}(x, y \mid x_0) \\ &= \frac{\beta_t}{V} \sum_{x_0} f_\theta(x_0 \mid x, t) \frac{\Pr(X_t = y \mid X_0 = x_0)}{\Pr(X_t = x \mid X_0 = x_0)}. \end{aligned}$$

The total reverse jump rate is therefore $\lambda_{f_\theta}(x, t) := \sum_{y \neq x} r_t^\theta(x, y)$. Substituting the previous expression gives

$$\lambda_{f_\theta}(x, t) = \frac{\beta_t}{V} \sum_{x_0} f_\theta(x_0 \mid x, t) \frac{\sum_{y \neq x} \Pr(X_t = y \mid X_0 = x_0)}{\Pr(X_t = x \mid X_0 = x_0)}.$$

There are two cases.

If $x_0 = x$, then

$$\Pr(X_t = x \mid X_0 = x) = \alpha_t + \frac{1 - \alpha_t}{V} = \frac{1 + (V-1)\alpha_t}{V},$$

while

$$\sum_{y \neq x} \Pr(X_t = y \mid X_0 = x) = \frac{(V-1)(1 - \alpha_t)}{V}.$$

Thus

$$\frac{\sum_{y \neq x} \Pr(X_t = y \mid X_0 = x)}{\Pr(X_t = x \mid X_0 = x)} = \frac{1 - \alpha_t}{\frac{1}{V-1} + \alpha_t}.$$

If $x_0 \neq x$, then

$$\Pr(X_t = x \mid X_0 = x_0) = \frac{1 - \alpha_t}{V},$$

and therefore

$$\sum_{y \neq x} \Pr(X_t = y \mid X_0 = x_0) = \frac{V - 1 + \alpha_t}{V}.$$

Hence

$$\frac{\sum_{y \neq x} \Pr(X_t = y \mid X_0 = x_0)}{\Pr(X_t = x \mid X_0 = x_0)} = \frac{V - 1 + \alpha_t}{1 - \alpha_t}.$$

Finally,

$$\lambda_{f_\theta}(x, t) = \frac{\beta_t}{V} \left[f_\theta(x \mid x, t) \frac{1 - \alpha_t}{\frac{1}{V-1} + \alpha_t} + \big(1 - f_\theta(x \mid x, t)\big) \frac{V - 1 + \alpha_t}{1 - \alpha_t} \right].$$

To express the reverse process in increasing time, define $\tau := 1 - t$, $Y_\tau := X_{1-\tau}$. The reverse generative process then evolves along $\tau : 0 \to 1$. Define its state-conditioned jump rate by $\mu_\theta(x, \tau) := \lambda_{f_\theta}(x, 1 - \tau)$. Let N_τ denote the schedule counting process of reverse transitions in $[0, \tau]$, and let $\mathcal{F}_{\tau^-}^S$ denote the history states generated only by transition times strictly before τ.

Conditioned on the current state,

$$\Pr(N_{\tau+h} - N_\tau = 1 \mid Y_\tau = x, \mathcal{F}_{\tau^-}^S) = \mu_\theta(x, \tau) h + o(h).$$

Marginalizing the latent state gives

$$\Pr(N_{\tau+h} - N_\tau = 1 \mid \mathcal{F}^S_{\tau^-}) = \sum_x \Pr(N_{\tau+h} - N_\tau = 1 \mid Y_\tau = x, \mathcal{F}^S_{\tau^-}) q_\theta(Y_\tau = x \mid \mathcal{F}^S_{\tau^-})$$
$$= \left[\sum_x \mu_\theta(x,\tau) q_\theta(Y_\tau = x \mid \mathcal{F}^S_{\tau^-})\right] h + o(h).$$

Therefore, the conditional intensity of the marginal schedule process is

$$\bar{\mu}_\theta(\tau \mid \mathcal{F}^S_{\tau^-}) = \lim_{h\to 0} \frac{\left[\sum_x \mu_\theta(x,\tau) q_\theta(Y_\tau = x \mid \mathcal{F}^S_{\tau^-})\right] h + o(h)}{h} = \mathbb{E}_{q_\theta}\left[\mu_\theta(Y_\tau, \tau) \mid \mathcal{F}^S_{\tau^-}\right].$$

Equivalently,

$$\bar{\mu}_\theta(\tau \mid \mathcal{F}^S_{\tau^-}) = \mathbb{E}_{q_\theta}\left[\lambda_{f_\theta}(Y_\tau, 1-\tau) \mid \mathcal{F}^S_{\tau^-}\right].$$

Let $\Sigma = \{\tau_1, \ldots, \tau_M\},\ 0 < \tau_1 < \cdots < \tau_M < 1$, where $\tau_i = 1 - t_{M+1-i}$. Let $R_\theta(a,b)$ denote the conditional probability that no transition occurs on $(a,b]$ along the observed schedule history. By the definition of $\bar{\mu}_\theta$,

$$R_\theta(a, b+h) = R_\theta(a,b)\left[1 - \bar{\mu}_\theta(b \mid \mathcal{F}^S_{b^-})h + o(h)\right].$$

By definition of partial derivative, we have:

$$\frac{\partial}{\partial b} R_\theta(a,b) = -\bar{\mu}_\theta(b \mid \mathcal{F}^S_{b^-}) R_\theta(a,b),$$

and, since $R_\theta(a,a) = 1$,

$$R_\theta(a,b) = \exp\left[-\int_a^b \bar{\mu}_\theta(u \mid \mathcal{F}^S_{u^-})\, du\right].$$

The density of observing transitions exactly at $\tau_1, \ldots, \tau_M$ and no additional transitions is therefore

$$q_\theta(\Sigma) = R_\theta(0,\tau_1)\bar{\mu}_\theta(\tau_1 \mid \mathcal{F}^S_{\tau_1^-}) \times R_\theta(\tau_1,\tau_2)\bar{\mu}_\theta(\tau_2 \mid \mathcal{F}^S_{\tau_2^-}) \times \cdots R_\theta(\tau_M, 1)$$
$$= \exp\left[-\int_0^1 \bar{\mu}_\theta(\tau \mid \mathcal{F}^S_{\tau^-})\, d\tau\right] \prod_{i=1}^M \bar{\mu}_\theta(\tau_i \mid \mathcal{F}^S_{\tau_i^-}).$$

We now express the forward schedule in the same reverse-time coordinate. Define

$$\mu^*(\tau) := \lambda^*(1-\tau) = \frac{V-1}{V}\beta_{1-\tau}.$$

Since the transformation $t \mapsto 1 - t$ is bijective with unit absolute Jacobian,

$$D_{\mathrm{KL}}\big(p(S) \,\|\, q_\theta(S)\big) = D_{\mathrm{KL}}\big(p(\Sigma) \,\|\, q_\theta(\Sigma)\big).$$

Moreover,

$$p(\Sigma) = \exp\left[-\int_0^1 \mu^*(\tau)\, d\tau\right] \prod_{i=1}^M \mu^*(\tau_i).$$

We therefore have

$$\mathcal{L}_{\mathrm{when}}(\theta) = \mathbb{E}_{\Sigma\sim p}\left[\log \frac{p(\Sigma)}{q_\theta(\Sigma)}\right].$$

Substituting the two schedule densities gives

$$\log \frac{p(\Sigma)}{q_\theta(\Sigma)} = \sum_{i=1}^M \log \frac{\mu^*(\tau_i)}{\bar{\mu}_\theta(\tau_i \mid \mathcal{F}^S_{\tau_i^-})} + \int_0^1 \left[\bar{\mu}_\theta(\tau \mid \mathcal{F}^S_{\tau^-}) - \mu^*(\tau)\right] d\tau.$$

It remains to convert the sum over observed transition times into an integral. For any predictable function $g(\tau)$,

$$\sum_{i=1}^M g(\tau_i) = \int_0^1 g(\tau)\, dN_\tau.$$

Under the forward schedule distribution p, the counting process has intensity $\mu^*(\tau)$, and thus

$$\mathbb{E}_p[dN_\tau \mid \mathcal{F}^S_{\tau^-}] = \mu^*(\tau)\, d\tau.$$

Using the tower property,

$$\mathbb{E}_p\left[\int_0^1 g(\tau)\, dN_\tau\right] = \mathbb{E}_p\left[\int_0^1 g(\tau)\mathbb{E}_p[dN_\tau \mid \mathcal{F}^S_{\tau^-}]\right] = \mathbb{E}_p \int_0^1 \mu^*(\tau) g(\tau)\, d\tau.$$

Taking $g(\tau) = \log \frac{\mu^*(\tau)}{\bar{\mu}_\theta(\tau \mid \mathcal{F}^S_{\tau^-})}$ yields

$$\mathbb{E}_p\left[\sum_{i=1}^M \log \frac{\mu^*(\tau_i)}{\bar{\mu}_\theta(\tau_i \mid \mathcal{F}^S_{\tau_i^-})}\right] = \mathbb{E}_p \int_0^1 \mu^*(\tau) \log \frac{\mu^*(\tau)}{\bar{\mu}_\theta(\tau \mid \mathcal{F}^S_{\tau^-})}\, d\tau.$$

Combining the two terms gives the schedule-matching loss:

$$\mathcal{L}_{\text{when}}(\theta) = \mathbb{E}_{\Sigma\sim p} \int_0^1 \left[\mu^*(\tau) \log \frac{\mu^*(\tau)}{\bar{\mu}_\theta(\tau \mid \mathcal{F}^S_{\tau^-})} - \mu^*(\tau) + \bar{\mu}_\theta(\tau \mid \mathcal{F}^S_{\tau^-})\right] d\tau. \tag{16}$$

C Proof of Theorem 3.3

For each sample when loss is

$$\ell_{\text{when}}(\theta) = \mu^* \log \frac{\mu^*}{\lambda_\theta} - \mu^* + \lambda_\theta.$$

The gradient w.r.t λ_θ is

$$\frac{\partial \ell}{\partial \lambda} = 1 - \frac{\mu^*}{\lambda}.$$

Therefore, at the optimum $\theta^\star$, we have

$$\lambda_{\theta^\star} = \mu^*,\ 1 - \frac{\mu^*}{\lambda_{\theta^\star}} = 0,$$

The gradient w.r.t θ is

$$\nabla_\theta \ell_{\text{when}}(\theta) = -\mu^* \nabla_\theta \log \lambda_\theta + \nabla_\theta \lambda_\theta = \left(1 - \frac{\mu^*}{\lambda_\theta}\right) \nabla_\theta \lambda_\theta.$$

The Hessian is,

$$\nabla^2_\theta \ell_{\text{when}}(\theta) = \frac{\mu^*}{\lambda_\theta^2} \nabla_\theta \lambda_\theta \nabla_\theta \lambda_\theta^\top + \left(1 - \frac{\mu^*}{\lambda_\theta}\right) \nabla^2_\theta \lambda_\theta.$$

Therefore

$$\nabla^2_\theta \ell_{\text{when}}(\theta^\star) = \frac{1}{\mu^*} \nabla_\theta \lambda_{\theta^\star} \nabla_\theta \lambda_{\theta^\star}^\top.$$

Taking expectation,

$$H_{\text{when}} := \nabla^2 \mathcal{L}_{\text{when}}(\theta^\star) = \mathbb{E}_{S,\tau,Y_\tau}\left[\frac{1}{\mu^*(\tau)} \nabla_\theta \lambda_{f_{\theta^\star}} \nabla_\theta \lambda_{f_{\theta^\star}}^\top\right].$$

For any $v \in \mathbb{R}^d$,

$$v^\top H_{\text{when}} v = \mathbb{E}\left[\frac{1}{\mu^*} v^\top \nabla \lambda_{\theta^\star} \nabla \lambda_{\theta^\star}^\top v\right] = \mathbb{E}\left[\frac{\left(v^\top \nabla \lambda_{\theta^\star}\right)^2}{\mu^*}\right] \geq 0.$$

Therefore, $H_{\text{when}} \succeq 0$ is positive semidefinite. Since $\mathcal{L}_U = \mathcal{L}_M + \mathcal{L}_{\text{when}}$, we have $H_U = H_M + H_{\text{when}}$. Therefore, $H_U \succeq H_M$.

We next characterize the stochastic-gradient covariance. For each model $j \in \{M, U\}$, let Z denote one stochastic training sample and let $\ell_j(\theta; Z) = -\log q_\theta^j(Z)$ denote the corresponding sample negative log-likelihood. Define the sample gradient $g_j(\theta; Z) = \nabla_\theta \ell_j(\theta; Z)$, and let $s_j(\theta; Z) = \nabla_\theta \log q_\theta^j(Z)$ denote the score function. Then $g_j(\theta; Z) = -s_j(\theta; Z)$. For any θ, the score has zero mean under the model distribution:

$$\begin{aligned}
\mathbb{E}_{Z\sim q_\theta^j}\left[s_j(\theta; Z)\right] &= \int q_\theta^j(z)\nabla_\theta \log q_\theta^j(z)\, dz && (17)\\
&= \int \nabla_\theta q_\theta^j(z)\, dz && (18)\\
&= \nabla_\theta \int q_\theta^j(z)\, dz && (19)\\
&= \nabla_\theta 1 && (20)\\
&= 0. && (21)
\end{aligned}$$

Differentiating Equation (21) w.r.t θ gives

$$\begin{aligned}
0 &= \nabla_\theta \int q_\theta^j(z) s_j(\theta; z)\, dz\\
&= \int \nabla_\theta q_\theta^j(z) s_j(\theta; z)^\top\, dz + \int q_\theta^j(z)\nabla_\theta s_j(\theta; z)^\top\, dz.
\end{aligned}$$

Using $\nabla_\theta q_\theta^j(z) = q_\theta^j(z) s_j(\theta; z)$, we have

$$\mathbb{E}_{q_\theta^j}\left[s_j(\theta; Z) s_j(\theta; Z)^\top\right] = -\mathbb{E}_{q_\theta^j}\left[\nabla_\theta^2 \log q_\theta^j(Z)\right].$$

At the optimum $\theta^\star$, the single-sample stochastic-gradient covariance is

$$\text{Cov}\left[g_j(\theta^\star; Z)\right] = \mathbb{E}[g_j g_j^\top] - \mathbb{E}[g_j]\mathbb{E}[g_j]^\top = \mathbb{E}[g_j g_j^\top] = H_j.$$

For a mini-batch of B iid samples,

$$\widehat{g}_j = \frac{1}{B}\sum_{b=1}^{B} g_j^{(b)}.$$

By independence,

$$\begin{aligned}
\text{Cov}\left[\widehat{g}_j\right] &= \frac{1}{B^2}\sum_{b=1}^{B} \text{Cov}\left[g_j^{(b)}\right]\\
&= \frac{1}{B^2} B H_j\\
&= \frac{1}{B} H_j.
\end{aligned}$$

Since the loss of UDM and MDM has the form of negative log-likelihood, the local gradient-noise covariance Σ_j is $\Sigma_j = \frac{1}{B} H_j,\ j \in \{M, U\}$. Since $H_U = H_M + H_{\text{when}}$, we have

$$\begin{aligned}
\Sigma_U - \Sigma_M &= \frac{1}{B}(H_U - H_M)\\
&= \frac{1}{B} H_{\text{when}} \succeq 0.
\end{aligned}$$

Thus, $\Sigma_U \succeq \Sigma_M$.

Consider either model $j \in \{M, U\}$ and define $e_k^j = \theta_k^j - \theta^\star$. Under the second order Taylor approximation,

$$\mathcal{L}_j(\theta) = \mathcal{L}_j^\star + \frac{1}{2}(\theta - \theta^\star)^\top H_j(\theta - \theta^\star),$$

and therefore

$$\nabla_\theta \mathcal{L}_j(\theta_k^j) = H_j e_k^j.$$

Write the stochastic gradient as

$$\widehat{g}_j(\theta_k^j) = H_j e_k^j + \xi_k^j,$$

where $\mathbb{E}\left[\xi_k^j \mid e_k^j\right] = 0$. The SGD update is

$$\theta_{k+1}^j = \theta_k^j - \eta \widehat{g}_j(\theta_k^j).$$

Subtracting $\theta^\star$ from both sides yields

$$e_{k+1}^j = (I - \eta H_j) e_k^j - \eta \xi_k^j. \tag{22}$$

Define $A_j = I - \eta H_j$. Since $H_U \succeq H_M$, we have

$$\lambda_{\max}(H_U) \geq \lambda_{\max}(H_M).$$

Therefore, the common learning-rate condition $0 < \eta < \frac{2}{\lambda_{\max}(H_U)}$ implies $0 < \eta < \frac{2}{\lambda_{\max}(H_j)}$ for both $j \in \{M, U\}$.

Diagonalize $H_j = Q_j \Lambda_j Q_j^\top$, where $\Lambda_j = \mathrm{diag}(\lambda_{j,1}, \ldots, \lambda_{j,d})$. Then

$$A_j = Q_j (I - \eta \Lambda_j) Q_j^\top.$$

For every eigenvalue $\lambda_{j,i} > 0$, $0 < \eta \lambda_{j,i} < 2$, which implies $|1 - \eta \lambda_{j,i}| < 1$. Hence $\rho(A_j) < 1$, and therefore $A_j^k \to 0$ as $k \to \infty$.

Define the second-moment matrix $P_k^j = \mathbb{E}\left[e_k^j (e_k^j)^\top\right]$. From Equation (22), we have

$$e_{k+1}^j (e_{k+1}^j)^\top = A_j e_k^j (e_k^j)^\top A_j^\top - \eta A_j e_k^j (\xi_k^j)^\top - \eta \xi_k^j (e_k^j)^\top A_j^\top + \eta^2 \xi_k^j (\xi_k^j)^\top.$$

Taking expectation, the cross terms vanish because

$$\mathbb{E}\left[e_k^j (\xi_k^j)^\top\right] = \mathbb{E}\left[e_k^j \mathbb{E}\left[(\xi_k^j)^\top \mid e_k^j\right]\right] = 0.$$

Thus

$$P_{k+1}^j = A_j P_k^j A_j^\top + \eta^2 \Sigma_j.$$

Iterating it gives

$$P_\infty^j = \eta^2 \sum_{r=0}^{\infty} A_j^r \Sigma_j (A_j^\top)^r.$$

Using $\Sigma_j = \frac{1}{B} H_j = \frac{1}{B} Q_j \Lambda_j Q_j^\top$, and $A_j = Q_j (I - \eta \Lambda_j) Q_j^\top$, we have

$$P_\infty^j = Q_j \left[\frac{\eta^2}{B} \sum_{r=0}^{\infty} (I - \eta \Lambda_j)^r \Lambda_j (I - \eta \Lambda_j)^r\right] Q_j^\top.$$

The i-th diagonal entry in the eigenbasis is

$$p_{\infty,j,i} = \frac{\eta^2 \lambda_{j,i}}{B} \sum_{r=0}^{\infty} (1 - \eta \lambda_{j,i})^{2r} = \frac{\eta^2 \lambda_{j,i} / B}{1 - (1 - \eta \lambda_{j,i})^2}.$$

Since

$$\begin{aligned}1-(1-\eta\lambda_{j,i})^2 &= 1-\left(1-2\eta\lambda_{j,i}+\eta^2\lambda_{j,i}^2\right)\\ &= 2\eta\lambda_{j,i}-\eta^2\lambda_{j,i}^2\\ &= \eta\lambda_{j,i}(2-\eta\lambda_{j,i}),\end{aligned}$$

we have

$$p_{\infty,j,i} = \frac{\eta}{B(2-\eta\lambda_{j,i})}.$$

Equivalently,

$$P_\infty^j = \frac{\eta}{B}(2I-\eta H_j)^{-1}. \tag{23}$$

Under the second order Taylor approximation,

$$\mathcal{L}_j(\theta_k^j) - \mathcal{L}_j^\star = \frac{1}{2}(e_k^j)^\top H_j e_k^j.$$

Therefore,

$$\mathbb{E}\left[\mathcal{L}_j(\theta_k^j) - \mathcal{L}_j^\star\right] = \frac{1}{2}\operatorname{Tr}\left(H_j P_k^j\right).$$

Taking $k\to\infty$ and using equation 23,

$$\mathbb{E}_{\text{ss}}\left[\mathcal{L}_j - \mathcal{L}_j^\star\right] = \frac{1}{2}\operatorname{Tr}\left(H_j P_\infty^j\right) = \frac{\eta}{2B}\operatorname{Tr}\left[H_j(2I-\eta H_j)^{-1}\right]. \tag{24}$$

We have already established $H_U \succeq H_M$. Multiplying by the negative scalar $-\eta$ reverses the Loewner order:

$$-\eta H_U \preceq -\eta H_M.$$

Adding $2I$ gives

$$2I-\eta H_U \preceq 2I-\eta H_M. \tag{25}$$

For positive-definite matrices, inversion reverses the Loewner order. Therefore, equation 25 implies

$$(2I-\eta H_U)^{-1} \succeq (2I-\eta H_M)^{-1}. \tag{26}$$

Define $F(H) = H(2I-\eta H)^{-1}$. Using $\eta H = 2I-(2I-\eta H)$, we have

$$\eta H(2I-\eta H)^{-1} = 2(2I-\eta H)^{-1} - I,$$

and hence

$$F(H) = \frac{2}{\eta}(2I-\eta H)^{-1} - \frac{1}{\eta}I. \tag{27}$$

Combining equation 26 and equation 27,

$$F(H_U) \succeq F(H_M). \tag{28}$$

Therefore,

$$\operatorname{Tr}\left[H_U(2I-\eta H_U)^{-1}\right] \geq \operatorname{Tr}\left[H_M(2I-\eta H_M)^{-1}\right]. \tag{29}$$

Using equation 24, we have

$$\mathbb{E}_{\text{ss}}\left[\mathcal{L}_U - \mathcal{L}_U^\star\right] \geq \mathbb{E}_{\text{ss}}\left[\mathcal{L}_M - \mathcal{L}_M^\star\right]. \tag{30}$$

D Proof of Theorem 3.5

We first show that replacing causal self-attention with block-causal self-attention does not increase the circuit complexity of one forward pass. Li et al. (2024) show that a constant-depth decoder-only Transformer with hidden dimension $\mathrm{poly}(n)$, numerical precision $O(\log n)$, and zero exponent bits can be simulated by a polynomial-size constant-depth threshold circuit. Their analysis considers causal self-attention, where the attention weights at position i are

$$s_i^{\mathrm{causal}} = \mathrm{softmax}_{0,s}\left(\langle q_i, k_1\rangle_{0,s}, \ldots, \langle q_i, k_i\rangle_{0,s}\right) \| 0^{n-i}.$$

Our MDM uses block-causal self-attention. For any $i \in [n]$, let $r(i) \coloneqq \min\{B(n)\lceil i/B(n)\rceil, n\}$ be the right endpoint of the block containing position i. Then the attention weights at position i are

$$s_i^{\mathrm{blk}} = \mathrm{softmax}_{0,s}\left(\langle q_i, k_1\rangle_{0,s}, \ldots, \langle q_i, k_{r(i)}\rangle_{0,s}\right) \| 0^{n-r(i)}.$$

Thus the only difference is that the causal prefix endpoint i is replaced by $r(i)$. Since $r(i) \le n$, the same finite-precision inner-product, softmax, and weighted-summation operations considered by Li et al. (2024) are applied to at most n inputs of $O(\log n)$ bits. Thus the same TC^0 simulation applies to block-causal self-attention. In particular, one forward pass of the MDM can be computed by a polynomial-size constant-depth threshold circuit.

We next adapt the parallel-enumeration argument of Li et al. (2024) to MDM decoding. Let Γ_B be the set of block-local decoding states needed to specify one decoding step, where each state specifies the token/mask configuration of the active block and its block-local generation order. Since $B = O(1)$ and $|\mathcal{V}| = O(1)$, we have $|\Gamma_B| \le (|\mathcal{V}|+1)^B B! = O(1)$.

A T-step MDM trajectory can therefore be represented by $\tau = (\gamma_1, \ldots, \gamma_T) \in \Gamma_B^T$. For $T \le c\log n$, the number of candidate trajectories is at most $|\Gamma_B|^T \le |\Gamma_B|^{c\log n} = n^{c\log|\Gamma_B|} = \mathrm{poly}(n)$.

We enumerate all candidate trajectories $\tau \in \Gamma_B^T$ in parallel. Fix a candidate $\tau = (\gamma_1, \ldots, \gamma_T)$. For every $t \in \{0, \ldots, T\}$, let $X_\tau^{(t)}(x)$ denote the MDM state obtained from x after applying the first t candidate actions $\gamma_1, \ldots, \gamma_t$, where $X_\tau^{(0)}(x) = x$. For every $t \in [T]$, define

$$E_{\tau,t}(x) \coloneqq \mathbf{1}\left[F_{\theta,n}\big(X_\tau^{(t-1)}(x), t\big) = \gamma_t\right].$$

Thus $E_{\tau,t}(x) = 1$ if and only if the t-th candidate action is consistent with the MDM decoding rule.

For a fixed τ, the actions $\gamma_1, \ldots, \gamma_T$ are fixed. Hence $X_\tau^{(t-1)}(x)$ is determined by x and $(\gamma_1, \ldots, \gamma_{t-1})$, and does not require the preceding model evaluations to be carried out sequentially. Thus all $E_{\tau,t}$ can be evaluated in parallel.

By the above single-forward-pass argument, $F_{\theta,n}(X_\tau^{(t-1)}(x), t)$ can be computed by a polynomial-size constant-depth threshold circuit. Since γ_t has constant size, testing equality with γ_t also takes constant depth. Thus $E_{\tau,t} \in \mathsf{TC}^0$ for every $\tau \in \Gamma_B^T$ and $t \in [T]$.

For each candidate trajectory τ, define $E_\tau(x) \coloneqq \bigwedge_{t=1}^T E_{\tau,t}(x)$. Since TC^0 allows unbounded-fan-in AND gates, $E_\tau \in \mathsf{TC}^0$. Under a deterministic decoding rule with fixed tie-breaking, for every input x there is a unique trajectory $\tau^\star$ such that $E_{\tau^\star}(x) = 1$.

Let $G_\tau(x)$ denote the final output corresponding to candidate trajectory τ. For every output bit j, define

$$C_{n,j}(x) \coloneqq \bigvee_{\tau \in \Gamma_B^T} \left(E_\tau(x) \wedge G_{\tau,j}(x)\right).$$

Since there are only $\mathrm{poly}(n)$ candidate trajectories and TC^0 allows unbounded-fan-in OR gates, $C_{n,j} \in \mathsf{TC}^0$. Moreover, since $\tau^\star$ is the unique consistent trajectory, $C_{n,j}(x) = G_{\tau^\star,j}(x)$. Thus C_n computes exactly the output of the original T-step MDM. This completes the proof.

E Proof of Theorem 3.6

Proof. Let $f = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n\ge 1} \in \mathsf{NC}^1$ be arbitrary. By Barrington's theorem, for every n, there exists a width-5 permutation branching program $\mathcal{B}_n = (I_1, \ldots, I_{T(n)})$ of length

$T(n) = \mathrm{poly}(n)$ that computes f_n. Each instruction is of the form $I_t = (i_t, \pi_t^0, \pi_t^1)$, where $i_t \in [n]$ and $\pi_t^0, \pi_t^1 \in S_5$. For an input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, define $\rho_t(x) \triangleq \pi_t^{x_{i_t}}$, where $\pi_t^{x_{i_t}} = \pi_t^0$ if $x_{i_t} = 0$ and $\pi_t^{x_{i_t}} = \pi_t^1$. The branching program can be chosen such that, for some fixed $\alpha \in S_5$,

$$f_n(x) = 1 \quad \Longleftrightarrow \quad \rho_1(x) \circ \cdots \circ \rho_{T(n)}(x) = \alpha.$$

We construct a constant-depth UDM that simulates one instruction of $\mathcal{B}_n$ at each decoding step. Define $\sigma_0 \triangleq e$ and $\sigma_t \triangleq \sigma_{t-1} \circ \rho_t(x)$ for $t \in [T(n)]$, where e is the identity element of S_5. It therefore suffices to maintain σ_t in a rewritable workspace.

Let the vocabulary contain the input symbols $\{0,1\}$ and one workspace symbol $\langle\sigma\rangle$ for each $\sigma \in S_5$. For every $\sigma \in S_5$, let $e_\sigma \in \{0,1\}^{120}$ denote its one-hot encoding. Since $|S_5| = 120$, all representations associated with permutations have constant dimension.

Let $k \triangleq \lceil \log_2(n+2) \rceil$ and use $\mathrm{sbin}_k(j)$ in (Li et al., 2024) to denote the signed binary representation of an address j. We reserve address $n+1$ for the workspace. Let C be a sufficiently large constant and take the hidden dimension to be $d' = 2k + C$. We partition the coordinates of each hidden representation into blocks storing, respectively, the input value, the position address, the query address, the current permutation state, the two permutations in the current instruction, the retrieved input bit, the next permutation state, a constant 1, and auxiliary coordinates.

For every input position $j \in [n]$, we choose the token and positional embeddings such that

$$h_j^0 = [x_j,\ \mathrm{sbin}_k(j),\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0]^\top,$$

where each zero denotes the zero vector of the corresponding block.

Suppose that before decoding step t, the workspace token is $\langle\sigma_{t-1}\rangle$. Its token embedding places $e_{\sigma_{t-1}}$ in the current-state block. The externally supplied timestep embedding at step t encodes the instruction $I_t = (i_t, \pi_t^0, \pi_t^1)$ by placing $\mathrm{sbin}_k(i_t), e_{\pi_t^0}$, and $e_{\pi_t^1}$ in the query-address and instruction blocks. Hence the initial hidden representation at the workspace position, denoted by $*$, is

$$h_*^0 = \left[0,\ \mathrm{sbin}_k(n+1),\ \mathrm{sbin}_k(i_t),\ e_{\sigma_{t-1}},\ e_{\pi_t^0},\ e_{\pi_t^1},\ 0,\ 0,\ 1,\ 0\right]^\top.$$

We next construct the attention layer. Following the exact-addressing construction of Li et al. (2024), we choose fixed linear projections W_Q, W_K, W_V such that W_Q reads the query-address and constant-1 blocks, W_K reads the position-address and constant-1 blocks, and W_V reads the input-value coordinate. In particular,

$$q_* \triangleq W_Q h_*^0 = \mathrm{sbin}_k(i_t) \frown \mathbf{1}_k,$$

while, for every $j \in [n]$,

$$k_j \triangleq W_K h_j^0 = B_s\left(\mathrm{sbin}_k(j) \frown (-\mathbf{1}_k)\right), \qquad v_j \triangleq W_V h_j^0 = x_j e_b,$$

where e_b denotes a designated retrieved-bit coordinate. At the workspace position,

$$k_* = W_K h_*^0 = B_s\left(\mathrm{sbin}_k(n+1) \frown (-\mathbf{1}_k)\right), \qquad v_* = 0.$$

These projections are realizable by fixed linear maps since they only select, rescale, duplicate, and relocate coordinates of the hidden representation.

By the exact-addressing lemma of Li et al. (2024),

$$\left[\exp\left(\langle q_*, k_j\rangle_s\right)\right]_s = \mathbf{1}[j = i_t]$$

for every $j \in [n]$. Moreover, $i_t \neq n+1$, so the workspace position itself receives zero attention weight. Hence the attention vector at the workspace position selects exactly position i_t, and its attention output is

$$W_O \sum_j a_j v_j = x_{i_t} e_b,$$

where W_O places the result in the retrieved-bit coordinate. After the residual connection, the workspace representation $h_*^{0.5}$ therefore contains $e_{\sigma_{t-1}}, e_{\pi_t^0}, e_{\pi_t^1}$, and x_{i_t}.

We now construct the feed-forward layer implementing the branching-program transition. For $\sigma, \mu, \nu \in S_5$ and $c \in \{0,1\}$, let u_σ, v^0_μ, and v^1_ν denote the corresponding coordinates of the current-state, π^0, and π^1 one-hot blocks of $h^{0.5}_*$. Let b denote the retrieved-bit coordinate and define $r_0(b) \triangleq 1 - b$ and $r_1(b) \triangleq b$. For every $(\sigma, \mu, \nu, c) \in S_5^3 \times \{0,1\}$, define

$$z_{\sigma,\mu,\nu,c} \triangleq \mathrm{ReLU}\left(u_\sigma + v^0_\mu + v^1_\nu + r_c(b) - 3\right).$$

Since all four quantities inside the sum are binary,

$$z_{\sigma,\mu,\nu,c} = \mathbf{1}\left[\sigma = \sigma_{t-1},\, \mu = \pi^0_t,\, \nu = \pi^1_t,\, c = x_{i_t}\right].$$

Thus exactly one such hidden unit is equal to 1.

For $c \in \{0,1\}$, define

$$P_c(\mu,\nu) \triangleq \begin{cases} \mu, & c = 0, \\ \nu, & c = 1. \end{cases}$$

For every $\tau \in S_5$, define the corresponding next-state coordinate by

$$y_\tau \triangleq \sum_{\substack{\sigma,\mu,\nu \in S_5,\, c \in \{0,1\}: \\ \sigma \circ P_c(\mu,\nu) = \tau}} z_{\sigma,\mu,\nu,c}.$$

Since exactly one $z_{\sigma,\mu,\nu,c}$ is nonzero,

$$y_\tau = \mathbf{1}\left[\tau = \sigma_{t-1} \circ \pi_t^{x_{i_t}}\right].$$

Therefore the next-state block is exactly

$$y = e_{\sigma_{t-1} \circ \pi_t^{x_{i_t}}} = e_{\sigma_t}.$$

The above mapping is realizable by a two-layer ReLU network of constant width, since $2|S_5|^3$ is independent of n. Moreover, all weights appearing in this construction belong to a fixed finite set and are therefore exactly representable under $O(\log n)$ bits of fixed-point precision.

We choose the output projection so that, for every $\tau \in S_5$, the logit of the workspace symbol $\langle \tau \rangle$ is the coordinate y_τ. Hence $\langle \sigma_t \rangle$ has logit 1, whereas every other workspace symbol has logit 0. Under deterministic argmax decoding, the model therefore outputs

$$w_t = \langle \sigma_t \rangle,$$

which rewrites the previous workspace token using the same workspace location.

We now verify the construction inductively. For every $t \in [T(n)]$, the workspace position satisfies

$$h^0_* = \left[0,\, \mathrm{sbin}_k(n+1),\, \mathrm{sbin}_k(i_t),\, e_{\sigma_{t-1}},\, e_{\pi^0_t},\, e_{\pi^1_t},\, 0,\, 0,\, 1,\, 0\right]^\top,$$

and, after the attention layer,

$$h^{0.5}_* = \left[0,\, \mathrm{sbin}_k(n+1),\, \mathrm{sbin}_k(i_t),\, e_{\sigma_{t-1}},\, e_{\pi^0_t},\, e_{\pi^1_t},\, x_{i_t},\, 0,\, 1,\, 0\right]^\top,$$

while the next-state block after the feed-forward layer is e_{σ_t}.

We claim that

$$\sigma_t = \rho_1(x) \circ \cdots \circ \rho_t(x)$$

for every $0 \le t \le T(n)$. The claim is immediate for $t = 0$ since $\sigma_0 = e$. Suppose it holds at step $t-1$. At step t, the timestep embedding supplies (i_t, π^0_t, π^1_t). By the attention construction above, the retrieved-bit coordinate is exactly x_{i_t}. The feed-forward layer therefore computes

$$\sigma_t = \sigma_{t-1} \circ \pi_t^{x_{i_t}} = \sigma_{t-1} \circ \rho_t(x).$$

Using the induction hypothesis,

$$\sigma_t = \rho_1(x) \circ \cdots \circ \rho_{t-1}(x) \circ \rho_t(x),$$

which establishes the claim.

After $T(n)$ decoding steps,

$$\sigma_{T(n)} = \rho_1(x) \circ \cdots \circ \rho_{T(n)}(x).$$

By the defining property of the branching program, $f_n(x) = 1$ if and only if $\sigma_{T(n)} = \alpha$. Since S_5 has constant size, this final equality test is realizable by a constant-size output layer.

Finally, $T(n) = \mathrm{poly}(n)$ by Barrington's theorem. The signed-binary address blocks require $k = O(\log n)$ coordinates, whereas all permutation-related blocks have constant dimension. Thus the construction in fact uses $d'(n) = O(\log n)$ hidden dimension, and hence certainly $d'(n) = \mathrm{poly}(n)$. It uses $O(\log n)$ bits of fixed-point precision and no exponent bits. Only one workspace token is preserved and rewritten between decoding steps, so $W(n) = O(1)$. Therefore

$$f \in \mathsf{T}_{\mathrm{UDM}}\left[\mathrm{poly}(n), \mathrm{poly}(n), O(\log n), 0, O(1)\right].$$

Since $f \in \mathsf{NC}^1$ was arbitrary, the result follows. □