Multi-Token Residual Prediction

Yufeng Xu[1,2] **Zishuo Bao**[1,2] **Qian Wang**[2] **Zeshen Zhang**[1] **Haoqi Zhang**[2]
Bowen Peng[3] **Ang Li**[1,4] **Rahul Chalamala**[4] **Yucheng Lu**[1,2]

[1]New York University [2]New York University Shanghai [3]Nous Research [4]Modal

{xu.yufeng, lu.yucheng}@nyu.edu

Code Models

Abstract

Diffusion Language Models (DLMs) generate text by iteratively denoising masked token sequences, offering a tradeoff between parallelism and quality compared to autoregressive models. In current practice, the number of tokens decoded per step is controlled by a confidence threshold, and quality degrades monotonically as more tokens are denoised per step. We introduce **Multi-token Residual Prediction (MRP)**, a lightweight module that enables dependency-aware multi-token denoising within a single backbone forward pass. MRP exploits a key property of the denoising process: the logit distributions at adjacent denoising steps are remarkably similar. Rather than running the backbone a second time to obtain the next-step logits, MRP predicts the residual between steps from the backbone's hidden states, effectively denoising more tokens per backbone forward at a fraction of the cost. We apply MRP across the two operating regimes of DLM decoding. In the *high-quality-low-throughput* static denoising regime, MRP serves as a drafter for speculative decoding: its proposals are verified against the backbone, yielding lossless acceleration of up to $1.4\times$ in SGLang. In the *low-quality-high-throughput* dynamic denoising regime, MRP instead drives a remasking scheme that revokes over-eager reveals, recovering most of the accuracy lost to aggressive low-threshold decoding and improving accuracy by up to 22.6 points on code generation task HumanEval and 17.7 points on reasoning task GSM8K.

1 Introduction

The design of language models involves a fundamental tension between *sequential dependency* and *parallelism*. Autoregressive models resolve this tension in favor of dependencies: each token is generated conditioned on the full history of preceding tokens, yielding high-quality text at the cost of strictly sequential decoding [Brown et al., 2020, Liu et al., 2024, Yang et al., 2025]. Diffusion Language Models (DLMs) make the opposite choice, predicting all the subsequent tokens simultaneously and refining them through iterative denoising [Sahoo et al., 2024, Lou et al., 2024, Nie et al., 2025, Hu et al., 2026]. This enables significant parallelism, but often at the cost of generation quality.

Recent efforts have sought to interpolate between these two extremes. Block diffusion [Arriola et al., 2025] applies diffusion within fixed-length blocks while maintaining autoregressive ordering across blocks. Threshold-based decoding [Cheng et al., 2025, Wu et al., 2025, 2026, Hong et al., 2025] adaptively controls how many tokens to unmask per step according to model confidence. These methods offer practical improvements, but they operate along the same quality–throughput Pareto frontier: unmasking additional tokens in a single step increases throughput at the expense of quality, precisely because each token is decoded without awareness of the other tokens resolved in the same step. The central question thus remains open: *can we decode multiple tokens per step in a DLM while preserving the inter-token dependencies that are essential for generation quality?*

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

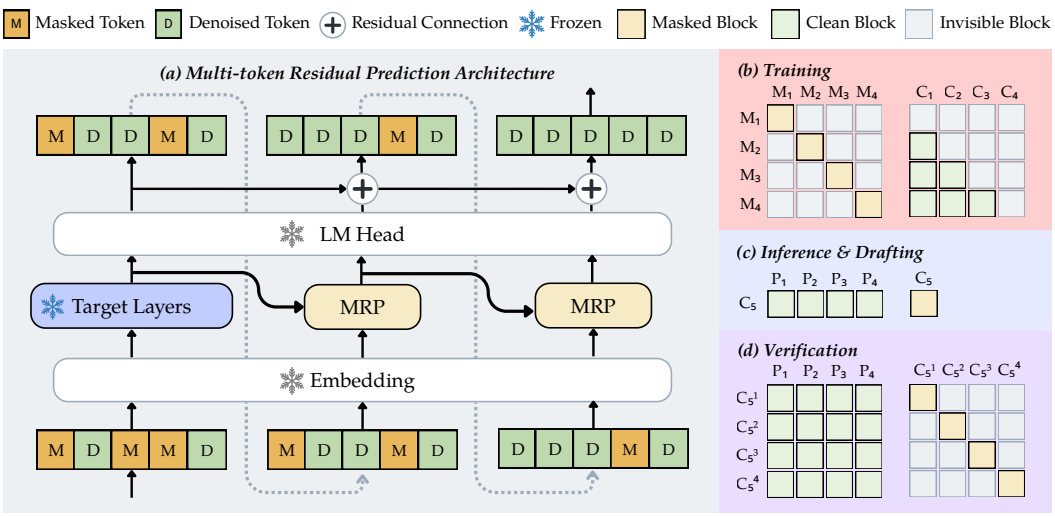

Figure 1: The figure above illustrates the training and inference pipeline of our multi-token residual prediction (MRP) method. *(a) Architecture.* A frozen DLM backbone embeds the input sequence, target layers produce hidden states for the masked positions, and a chain of lightweight MRP heads sequentially predict the residual of the next logit state and decode the next probable token [D] at the adjacent masked [M] position through the shared LM head. *(b) Training.* The sequence is duplicated and one half is randomly masked; loss is computed only on the masked half, where each masked block M_i attends to itself and to all preceding clean blocks $C_{1:i-1}$. *(c) Inference & drafting.* A single new block is decoded in parallel given the committed clean prefix $P_{1:i-1}$, supporting both direct and speculative generation under an identical attention pattern. *(d) Verification.* The $K{+}1$ draft candidates $C_i^{0:K}$ are verified jointly against the backbone, where candidate C_i^k embeds the first k draft tokens and attends only to the clean prefix $P_{1:i-1}$ and to itself, so all candidates are scored in one forward pass.

We answer this question affirmatively with a key insight: the logit distributions produced by a diffusion backbone at adjacent denoising steps are remarkably similar. When only a few tokens are unmasked between two consecutive steps, the backbone's output barely changes, yet computing it requires a full forward pass. The residual between the two outputs is significantly smaller in magnitude than the outputs themselves, making it a far easier prediction target: the zero function is already a reasonable approximation, and a small module only needs to learn corrections to an already-good prediction, following the same principle underlying residual learning [He et al., 2016].

Building on this observation, we introduce **Multi-token Residual Prediction (MRP)**, a lightweight module that predicts the inter-step logit residual directly from the backbone's hidden states. At inference time, a single backbone forward pass produces both the current logits and the hidden representation from which MRP infers the residual correction. The corrected logits approximate the output of a second backbone forward pass at a fraction of the computational cost, enabling additional tokens to be decoded without running the backbone again.

We deploy MRP across the two operating regimes of DLM decoding. In the *high-quality, low-throughput* static denoising regime, where output quality must be preserved, MRP serves as a lightweight drafter for **speculative decoding**: the MRP module proposes additional tokens beyond what the backbone's confidence yields, and these proposals are verified by the backbone in its subsequent forward pass. Accepted tokens incur no quality loss, yielding lossless acceleration.

As an additional knob, MRP also supports **direct decoding**, in which the corrected log-density is used without verification, providing a tunable quality–speed tradeoff that is strictly more favorable than threshold-based decoding at matched throughput. In the *low-quality, high-throughput* dynamic denoising regime, where a low threshold unmasks many tokens per step at the cost of accuracy, MRP instead drives a **remasking decoding** scheme: the corrected log-density revokes over-eager reveals, recovering most of the accuracy lost to aggressive unmasking at a moderate efficiency cost.

Our contributions are as follows:

- We identify and theoretically characterize the inter-step log-density residual in masked diffusion language models (Section 3). This characterization motivates and justifies the design of a lightweight residual predictor that operates on backbone hidden states.

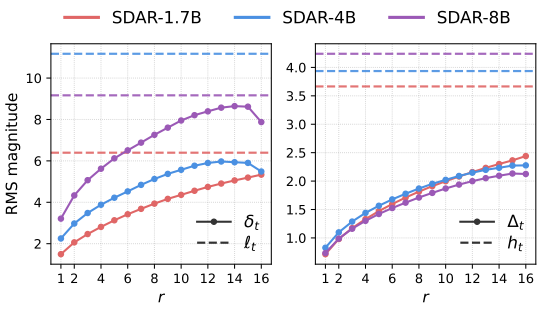

Figure 2: Magnitudes of the per-step residual versus the full backbone state, measured on GSM8K and averaged per entry (RMS). The horizontal axis r is the number of tokens unmasked between two adjacent denoising states $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t-1}$; line color indicates the SDAR backbone scale (1.7B / 4B / 8B). **Left:** logits. Solid curves with markers show the residual $\boldsymbol{\delta}_t = \boldsymbol{\ell}_{t-1} - \boldsymbol{\ell}_t$; dashed lines show the average magnitude of the backbone logits $\boldsymbol{\ell}_t$ for reference. **Right:** hidden states. Solid curves show the residual $\boldsymbol{\Delta}_t = \boldsymbol{h}_{t-1} - \boldsymbol{h}_t$; dashed lines show the average magnitude of the backbone hidden states $\boldsymbol{h}_t$. Across all three scales and both representations, the residual is consistently smaller than the underlying state, with the gap widening sharply for small r (the practical decoding regime).

- We introduce MRP, a residual prediction module trained with a tailored objective, and show how it applies across both regimes of DLM decoding: speculative decoding in the static regime and remasking in the dynamic regime, analyzing the operating point of each (Section 3).
- We integrate our method into the open-sourced inference engine SGLang [Zheng et al., 2024] and demonstrate that MRP achieves up to $1.4\times$ lossless speedup in the static regime and recovers up to 22.6 accuracy points in the dynamic regime, on SDAR models across three model scales and four reasoning and code benchmarks (Section 4).

2 Preliminary

We start by describing the decoding procedure of Diffusion Language Models (DLMs) and establish the notation and terminology used throughout the rest of the paper.

Denoising Loop. A DLM generates a sequence of L tokens by iteratively reversing a masking process [Sahoo et al., 2024, Nie et al., 2025]. Generation begins from a fully masked sequence $\boldsymbol{x}_T$ in which every position is set to a special `[MASK]` token. At each denoising step $t = T, T-1, \ldots, 1$, a bidirectional Transformer f (the *backbone*) takes the partially masked sequence $\boldsymbol{x}_t$ as input and produces logits $\boldsymbol{\ell}_t = f(\boldsymbol{x}_t) \in \mathbb{R}^{L\times V}$, where V is the vocabulary size.

These logits are unnormalized log-densities; a softmax over the vocabulary normalizes them into the per-position predictive distribution $\boldsymbol{\pi}_t^i = \mathrm{softmax}(\boldsymbol{\ell}_t^i)$, where $\boldsymbol{\pi}_t^i = \boldsymbol{\pi}_t \boldsymbol{e}_i$ denotes the distribution for the i-th position. We follow the literature and denote the top-1 probability at each position as its *confidence* [Cheng et al., 2025, Nie et al., 2025]. After each forward pass, the model reveals the masked positions it is most confident about, replacing their `[MASK]` tokens with the predicted tokens to form $\boldsymbol{x}_{t-1}$. This repeats until all positions are revealed.

Block Diffusion. In the block diffusion variant [Arriola et al., 2025], generation is partitioned into blocks of tokens produced left to right. Previously generated blocks serve as fully unmasked context while the current block is denoised through the loop described above. Our method applies identically within each block; all results in this paper use the block diffusion setting.

3 Multi-Token Residual Prediction

3.1 Motivation

Our motivation starts with a natural question following Section 2: *how close are the logits $\boldsymbol{\ell}_t$ and $\boldsymbol{\ell}_{t-1}$ in practice, and can we characterize the structure of their difference?* We address this question empirically and then provide theoretical grounding.

Empirical characterization of the residual. We define the per-step logit residual as

$$\boldsymbol{\delta}_t = \boldsymbol{\ell}_{t-1} - \boldsymbol{\ell}_t \in \mathbb{R}^{L\times V}, \tag{1}$$

where $\boldsymbol{\ell}_t$ and $\boldsymbol{\ell}_{t-1}$ are the backbone logits before and after unmasking a set of positions. Analogously, we define the per-step hidden-state residual $\boldsymbol{\Delta}_t = \boldsymbol{h}_{t-1} - \boldsymbol{h}_t$ between the backbone hidden states at adjacent denoising steps. To probe the residual at different effective step sizes, we run greedy block-diffusion decoding on the GSM8K evaluation set with SDAR-1.7B/4B/8B, and compare backbone states that are r revealed-token steps apart, with revealed token count r ranging from 1 to 16 (a full block). Figure 2 reports the resulting residuals; see Appendix B for the full measurement protocol.

The key observation here is the *magnitude gap*: the RMS magnitude of $\boldsymbol{\delta}_t$ is consistently and significantly smaller than that of the full logits $\boldsymbol{\ell}_t$. When $r = 1$, the residual is around 20% of the magnitude of the logits; even at $r = 16$ (unmasking an entire block at once), the residual remains below the full logits in magnitude across all model scales. This gap widens for smaller r, which is precisely the operating regime of practical denoising. This suggests that predicting $\boldsymbol{\delta}_t$ requires capturing a signal with substantially lower dynamic range than predicting $\boldsymbol{\ell}_{t-1}$ from scratch.

Theoretical justification. The empirical observations above can be grounded in the Markov chain structure of the denoising process. The sequence of partially masked states $\boldsymbol{x}_T, \boldsymbol{x}_{T-1}, \ldots, \boldsymbol{x}_0$ forms a Markov chain: each state $\boldsymbol{x}_{t-1}$ is obtained from $\boldsymbol{x}_t$ by unmasking a subset of positions $\mathcal{R}_t$, so the transition depends only on the current state. The backbone output $\boldsymbol{\ell}_t = f(\boldsymbol{x}_t)$ is a deterministic function of the chain's state. We exploit this structure to bound the residual $\boldsymbol{\delta}_t = \boldsymbol{\ell}_{t-1} - \boldsymbol{\ell}_t$.

The key observation is that the transition $\boldsymbol{x}_t \to \boldsymbol{x}_{t-1}$ is a *small perturbation*: it modifies only $|\mathcal{R}_t|$ out of L positions, leaving the remaining $L - |\mathcal{R}_t|$ positions unchanged. For any masked position $i \notin \mathcal{R}_t$, the change in its predictive distribution is driven entirely by the indirect influence of the newly revealed tokens through the backbone's attention mechanism. Recall from Section 2 the predictive distribution $\boldsymbol{\pi}_t^i = \mathrm{softmax}(\boldsymbol{\ell}_t^i)$. We bound the change across one transition as follows.

Proposition 1 (One-step contraction)**.** *If the backbone f is κ-Lipschitz with respect to its input embeddings, then for any masked position $i \notin \mathcal{R}_t$:*

$$D_{\mathrm{TV}}(\boldsymbol{\pi}_{t-1}^i,\, \boldsymbol{\pi}_t^i) \le \kappa \cdot \frac{|\mathcal{R}_t|}{L} \cdot \max_{j \in \mathcal{R}_t} \|\boldsymbol{e}_{v_j} - \boldsymbol{e}_{\texttt{[MASK]}}\|_2, \tag{2}$$

where $\boldsymbol{e}_v$ denotes the embedding of token v, $\boldsymbol{e}_{\texttt{[MASK]}}$ denotes the embedding of mask token and D_{TV} is the total variation distance between the two distributions.

The bound depends on three intuitive quantities: the model's sensitivity κ, the fraction of positions perturbed $|\mathcal{R}_t|/L$, and the embedding distance between the revealed tokens and the mask token. Since $|\mathcal{R}_t| \ll L$ at each step[1], the perturbation ratio is small and the TV distance is bounded.

Applying this bound recursively along the chain yields a cumulative characterization:

Corollary 2 (Cumulative residual decay)**.** *As denoising progresses, the number of remaining masked positions $m_t = |\{i : x_t^i = \texttt{[MASK]}\}|$ decreases monotonically. At later steps, the perturbation ratio $|\mathcal{R}_t|/L$ remains small while the predictions $\boldsymbol{\pi}_t^i$ concentrate (the backbone becomes more confident on fewer remaining positions). Consequently, $\|\boldsymbol{\delta}_t\|$ decreases along the chain: later denoising steps produce smaller residuals than earlier ones.*

We provide the full proofs in Appendix G. These results justify the design of MRP: the residual $\boldsymbol{\delta}_t$ is a low-complexity prediction target, learnable by a module far smaller than the backbone.

3.2 Architecture

The MRP module follows an architectural pattern similar to EAGLE [Li et al., 2024] and Multi-Token Prediction (MTP) [Liu et al., 2024]: a lightweight head that receives hidden states from a frozen backbone and produces additional token predictions. The key difference is that MRP outputs a *residual correction* rather than the entire distribution. The final logits are obtained by summing the backbone logits and the MRP output, $\boldsymbol{\ell}_t + \boldsymbol{\delta}_t$, so the module only needs to learn the deviation from the backbone's prediction rather than a complete distribution over the vocabulary. The MRP module takes the backbone's hidden states $\boldsymbol{h}$ and the token embeddings of the revealed sequence as input, fuses them via a learned projection, passes the result through a small number of transformer layers, and reuses the backbone's language model head to produce the residual logits $\boldsymbol{\delta}_t$. The corrected logits are $\boldsymbol{\ell}_t + \boldsymbol{\delta}_t$. The full architecture is illustrated in Figure 1.

[1] In some recent models such as SDAR, L could be small for a single block (e.g., 16). However, this doesn't violate our bound or assumption.

Algorithm 1 MRP Training

Require: Backbone f (frozen), MRP module g (trainable)

for each training sequence $\boldsymbol{x}_0$ **do**
  Mask response tokens independently with probability $\mathrm{Uniform}(0,1)$ to obtain $\boldsymbol{x}_t$
  $\boldsymbol{h}, \boldsymbol{\ell}_t \leftarrow f(\boldsymbol{x}_t)$ ▷ backbone forward: hidden states + logits
  Reveal r ground-truth tokens per block to obtain $\boldsymbol{x}_{t-1}$
  $\boldsymbol{\delta}_t \leftarrow g(\boldsymbol{x}_t, \boldsymbol{h})$ ▷ MRP residual prediction
  $\boldsymbol{\ell}_{t-1} \leftarrow f(\boldsymbol{x}_{t-1})$ ▷ Obtain teacher target
  $\mathcal{L} \leftarrow \mathrm{KL}\Big(\mathrm{softmax}\big(\boldsymbol{\ell}_{t-1}\big) \,\Big\|\, \mathrm{softmax}\big(\boldsymbol{\ell}_t + \boldsymbol{\delta}_t\big) \Big)\Big|_{\text{masked}}$
  Update g with $\nabla\mathcal{L}$
end for

3.3 Training

The MRP module is trained to predict the log-density residual between two consecutive denoising states. Given a clean sequence $\boldsymbol{x}_0$, we first corrupt it by independently masking response tokens with probability $\mathrm{Uniform}(0,1)$ to obtain the noisy input $\boldsymbol{x}_t$. We then reveal r ground-truth tokens per block to form the partially revealed input $\boldsymbol{x}_{t-1}$, simulating one denoising step. The frozen backbone f is run twice: once on $\boldsymbol{x}_t$ to produce hidden states $\boldsymbol{h}$ and log-densities $\log p(\boldsymbol{x}_0|\boldsymbol{x}_t)$, and once on $\boldsymbol{x}_{t-1}$ to produce the teacher target $\log p(\boldsymbol{x}_0|\boldsymbol{x}_{t-1})$ without gradients. The MRP module g takes $\boldsymbol{x}_{t-1}$ and the backbone hidden states $\boldsymbol{h}$ as input and predicts the residual $\boldsymbol{\delta}_t$. Training encourages $\boldsymbol{\delta}_t$ to match the teacher output $\log p(\boldsymbol{x}_0|\boldsymbol{x}_{t-1}) - \log p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ via a KL divergence objective restricted to still-masked positions:

$$\mathcal{L} = \mathrm{KL}\Big(\mathrm{softmax}\big(\boldsymbol{\ell}_{t-1}\big) \,\Big\|\, \mathrm{softmax}\big(\boldsymbol{\ell}_t + \boldsymbol{\delta}_t\big) \Big)\Big|_{\text{masked}}, \tag{3}$$

where $\boldsymbol{\ell}_t = f(\boldsymbol{x}_t)$ and $\boldsymbol{\ell}_{t-1} = f(\boldsymbol{x}_{t-1})$ are the backbone logits on the noisy and revealed inputs respectively. Since the logits are unnormalized log-densities, and the KL divergence is computed after softmax, the normalizing constants cancel and the objective is equivalent to matching the true conditional distributions. The only trainable parameters are those of the MRP module g. Algorithm 1 summarizes the full training procedure, from masking to the gradient update.

3.4 Inference

At inference time, the MRP module augments each backbone forward pass with additional residual prediction, allowing more tokens to be resolved per backbone forward. After the backbone processes $\boldsymbol{x}_t$ and unmasks high-confidence positions to form $\boldsymbol{x}_{t-1}$, the MRP module predicts the residual $\boldsymbol{\delta}_t$ from the backbone's hidden states and the updated sequence, and the corrected logits $\boldsymbol{\ell}_t + \boldsymbol{\delta}_t$ approximate a second backbone forward at a fraction of the cost.

How this correction is used depends on the decoding regime. In the *static* regime, where each step unmasks a fixed (typically small) number of tokens and quality is the priority, MRP *adds* reveals beyond the threshold through two modes: *direct* decoding, which uses the corrected logits without verification for a tunable quality–throughput tradeoff, and *speculative* decoding, which verifies them against the backbone for lossless acceleration. In the *dynamic* regime, where a confidence threshold unmasks many tokens per step and accuracy degrades, MRP instead *revokes* over-eager reveals through *remasking* decoding. We describe each in turn.

Direct decoding (Algorithm 2). The simplest use of MRP in the static regime is to unmask additional positions directly from the corrected logits, without any verification. After the backbone unmasks high-confidence positions, the MRP module runs K additional decoding steps. At each step, the module predicts both a hidden state residual $\boldsymbol{\Delta}$ and a logit residual $\boldsymbol{\delta}$, which are accumulated into the running hidden states and logits: $\boldsymbol{h} \leftarrow \boldsymbol{h} + \boldsymbol{\Delta}$, $\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} + \boldsymbol{\delta}$. r tokens with highest confidence are unmasked immediately. This introduces a small approximation error. Larger K unmasks more tokens per backbone forward at the cost of compounding approximation error; we study this tradeoff in Section 4. Direct decoding thus sits on a strictly more favorable quality–throughput frontier than threshold-based decoding at matched throughput, at the price of a controlled quality loss.

Algorithm 2 MRP Inference (Direct Decoding Mode) in Static Denoising

Require: backbone f, MRP module g, MRP steps K, number of reveal tokens r, fully masked sequence $\boldsymbol{x}_T$

$t \leftarrow T$
while $\boldsymbol{x}_t$ contains `[MASK]` tokens **do**
  $\boldsymbol{h}, \boldsymbol{\ell} \leftarrow f(\boldsymbol{x}_t)$ ▷ backbone forward to produce hidden states and logits
  Reveal r tokens with highest confidence in $\boldsymbol{x}_t$
  for $k = 1, \ldots, K$ **do**
    $\boldsymbol{\Delta}, \boldsymbol{\delta} \leftarrow g(\boldsymbol{x}_t, \boldsymbol{h})$ ▷ MRP step to produce hidden residual and logits residual
    $\boldsymbol{h} \leftarrow \boldsymbol{h} + \boldsymbol{\Delta}; \quad \boldsymbol{\ell} \leftarrow \boldsymbol{\ell} + \boldsymbol{\delta}$
    Reveal r tokens with highest confidence in $\boldsymbol{x}_t$
  end for
  $t \leftarrow t - 1$
end while
return $\boldsymbol{x}_t$

Speculative decoding (Algorithm 3). When output quality must be preserved, the same MRP-decoded tokens are used as a draft that the backbone verifies, as in speculative decoding for autoregressive models [Liu et al., 2024]. Drafting is identical to direct decoding, but instead of committing the drafts, we adopt the verification framework of Self Speculative Decoding (SSD) [Gao et al., 2025]: after all K MRP steps complete, the backbone runs a single verification forward pass on the updated sequence $\boldsymbol{x}_{t-1}$, producing $\boldsymbol{\ell}' = f(\boldsymbol{x}_{t-1})$. Drafted positions where the backbone's prediction matches the draft ($\arg\max \ell'^i = \arg\max \ell^i$) are accepted; disagreeing positions are remasked.

Since accepted tokens are exactly those the backbone itself would have produced, acceptance incurs no quality loss, making this mode entirely *lossless*. This position-wise verification is natural for diffusion models, where all positions are predicted simultaneously in a single forward pass. The verification pass is not wasted: its hidden states and logits seed the next iteration, so when the acceptance rate is high, the verification forward effectively serves as the backbone forward for the next step, amortizing its cost across consecutive denoising steps.

Algorithm 3 MRP Inference (Speculative Decoding Mode) in Static Denoising

Require: backbone f, MRP module g, MRP steps K, number of reveal tokens r, fully masked sequence $\boldsymbol{x}_T$

$t \leftarrow T$
while $\boldsymbol{x}_t$ contains `[MASK]` tokens **do**
  $\boldsymbol{h}, \boldsymbol{\ell} \leftarrow f(\boldsymbol{x}_t)$ ▷ backbone forward
  Reveal r tokens with highest confidence in $\boldsymbol{x}_t$
  for $k = 1, \ldots, K$ **do** ▷ MRP drafting
    $\boldsymbol{\Delta}, \boldsymbol{\delta} \leftarrow g(\boldsymbol{x}_t, \boldsymbol{h})$
    $\boldsymbol{h} \leftarrow \boldsymbol{h} + \boldsymbol{\Delta}; \quad \boldsymbol{\ell} \leftarrow \boldsymbol{\ell} + \boldsymbol{\delta}$
    Draft token with highest confidence into $\boldsymbol{x}_{t-1}$
  end for
  $\boldsymbol{h}', \boldsymbol{\ell}' \leftarrow f(\boldsymbol{x}_{t-1})$ ▷ backbone verify
  Reject drafted positions where $\arg\max \ell'^i \neq \arg\max \ell^i$; remask rejected positions in $\boldsymbol{x}_{t-1}$
  $t \leftarrow t - 1$
end while
return $\boldsymbol{x}_t$

Remasking decoding (Algorithm 4). The two modes above operate in the static regime and use MRP to *add* reveals beyond what the backbone's threshold yields. In the dynamic, high-throughput regime, a low confidence threshold causes the backbone to unmask many positions simultaneously, and accuracy degrades sharply as a result. Here MRP instead *remasks* over-eager reveals. At a low threshold τ, the backbone reveals the set $\mathcal{R}$ of positions whose confidence exceeds τ. A single MRP pass then predicts the residual $\boldsymbol{\delta}$ conditioned on these fresh reveals, giving the estimated logits for the subsequent step $\boldsymbol{\ell} + \boldsymbol{\delta}$. We then use the estimated next-step logits to remask revealed tokens in the

Algorithm 4 MRP Inference (Remasking Mode) in Dynamic Denoising

Require: backbone f, MRP module g, threshold τ, fully masked sequence $\boldsymbol{x}_T$

$t \leftarrow T$
while $\boldsymbol{x}_t$ contains `[MASK]` tokens **do**
  $\boldsymbol{h}, \boldsymbol{\ell} \leftarrow f(\boldsymbol{x}_t)$ ▷ backbone forward
  Reveal all the tokens with confidence exceeding a threshold τ, forming a set $\mathcal{R}$
  $\boldsymbol{\Delta}, \boldsymbol{\delta} \leftarrow g(\boldsymbol{x}_t, \boldsymbol{h})$
  $\boldsymbol{h} \leftarrow \boldsymbol{h} + \boldsymbol{\Delta}; \quad \boldsymbol{\ell} \leftarrow \boldsymbol{\ell} + \boldsymbol{\delta}$ ▷ MRP forward
  Remask positions $i \in \mathcal{R}$ where confidence is below τ ▷ remask tokens below the same τ
  $t \leftarrow t - 1$
end while
return $\boldsymbol{x}_t$

current step. The intuition is that a single-step confidence estimate is noisy: a position clearing τ once may do so spuriously, whereas a position that stays above τ across two consecutive steps has more stably crossed the threshold, and only such positions are kept committed. This double-check echoes the re-masking principle of WINO [Hong et al., 2025], but replaces its second backbone pass with a single lightweight MRP forward. Using the same τ for both the reveal and the remasking, it trades one extra MRP forward for higher accuracy at aggressive thresholds, and introduces no tuning beyond what dynamic decoding already requires (Section 4.4).

Complementary Operating regimes. The decoding strategies map onto the two operating regimes of DLM inference rather than competing within one. In the static regime, where few tokens are unmasked per step and quality is paramount, MRP adds reveals: direct decoding trades a controlled quality loss for throughput, while speculative decoding preserves backbone quality through verification. In the dynamic regime, where a low threshold unmasks aggressively for speed, MRP revokes reveals through remasking, recovering most of the accuracy that aggressive unmasking would otherwise sacrifice. All strategies share the same trained MRP module; the choice among them is made at serving time and requires no retraining of the module.

4 Experiments

In this section, we present our main empirical results. We first validate the residual learning hypothesis by showing that predicting the inter-step residual is substantially easier than predicting the full next-step distribution (Section 4.2). We then evaluate MRP under both budget-based static denoising (Section 4.3) and threshold-based dynamic denoising (Section 4.4), and study how the MRP module scales with its depth in Appendix E.

4.1 Setup

Models. We attach our MRP module to the SDAR-Chat family of block-diffusion language models [Cheng et al., 2025] at three scales (1.7B, 4B, 8B), all trained with block size 16. The MRP module is a D=3-layer transformer that follows the architecture of Section 3.2; the backbone, language-modeling head, and token embeddings remain *frozen* during training.

Dataset. We adopt the training dataset SFTDatasetV3 [Wang et al., 2024], a 12.4M-conversation supervised fine-tuning corpus assembled from three public instruction-tuning sources: OpenHermes-2.5 [Teknium, 2024], GenQA [Chen et al., 2024a], and Infinity-Instruct [BAAI, 2024], and reformatted into a uniform two- to eight-turn user/assistant message schema. The mixture spans diverse domains (reasoning, mathematics, code, dialogue) at a scale large enough that the MRP module never revisits a sample, which is well-suited to the distillation-style residual objective we train against.

Training. We train with the residual KD loss of Algorithm 1 at temperature T_{KD}=1.0 in the forward direction KL, restricted to still-masked positions, and unroll K=2 MRP iterations per backward with uniform per-step weighting. We use AdamW with peak learning rate $1\mathrm{e}{-3}$ on a cosine schedule for a single epoch. The full hyperparameter list is given in Appendix C (Table 4).

Evaluation. We follow the evaluation protocol of SDAR [Cheng et al., 2025] and focus on the two generation task categories: *(i) Mathematics* – GSM8K (0-shot, CoT) [Cobbe et al., 2021] and

Type	1.7B				4B				8B			
	K=1	K=2	K=3	K=4	K=1	K=2	K=3	K=4	K=1	K=2	K=3	K=4
Direct Modeling	76.5	73.1	62.4	48.0	84.8	19.6	5.9	1.9	90.5	72.0	34.2	6.7
Residual Modeling	76.7	73.9	66.7	60.9	88.6	87.6	70.9	57.2	90.1	89.2	84.8	79.8
Δ (Res. − Dir.)	+0.2	+0.8	+4.3	+12.9	+3.8	+68.0	+65.0	+55.3	-0.4	+17.2	+50.6	+73.1

Table 1: GSM8K (0-shot, CoT) accuracy (%) of MRP trained with the *residual* objective versus a *direct* distillation objective, evaluated at $K=\{1,2,3,4\}$ MRP steps per backbone forward. Both variants share identical architecture and hyperparameters and are trained on the same data (`sftdatasetv3` [Wang et al., 2024]); they are evaluated under static low-confidence decoding. The Δ row reports the improvement of residual over direct training, with green indicating residual is better and red indicating residual is worse.

Model	Setting	GSM8K		MATH500		HumanEval		MBPP	
		Acc.	TPS	Acc.	TPS	Acc.	TPS	Acc.	TPS
1.7B	Base (r=1)	77.6	339.0	55.0	358.0	53.7	314.0	52.9	282.0
	Base (r=2)	70.8	1.79×	45.8	1.79×	29.9	1.70×	32.3	1.92×
	Direct (K=1)	76.7	1.50×	54.2	1.45×	48.2	1.45×	45.5	1.45×
	Direct (K=2)	73.9	1.71×	50.6	1.65×	41.5	1.53×	42.4	1.72×
	Spec (K=3)	78.3	1.25×	58.4	1.23×	55.5	1.23×	53.3	1.17×
4B	Base (r=1)	90.3	188.0	71.4	204.0	67.1	170.0	66.5	161.0
	Base (r=2)	86.4	1.84×	64.0	1.81×	45.7	1.74×	37.7	1.81×
	Direct (K=1)	88.6	1.56×	64.4	1.54×	62.8	1.55×	55.6	1.43×
	Direct (K=2)	87.6	1.84×	60.4	1.80×	53.0	1.79×	51.0	1.82×
	Spec (K=3)	90.0	1.36×	68.0	1.26×	67.7	1.35×	66.5	1.27×
8B	Base (r=1)	90.9	126.0	72.2	132.0	73.8	114.0	67.7	102.0
	Base (r=2)	87.7	1.80×	68.4	1.81×	42.1	1.74×	48.2	1.60×
	Direct (K=1)	90.1	1.59×	71.4	1.61×	67.1	1.53×	63.8	1.51×
	Direct (K=2)	89.2	1.89×	70.8	1.91×	64.0	1.78×	59.9	1.75×
	Spec (K=3)	90.4	1.40×	74.8	1.39×	72.6	1.34×	67.3	1.34×

Table 2: MRP under static denoising across model scales. *Base (r)* denotes the backbone-only baseline that unmasks r tokens per denoising step. *Direct* mode runs MRP with $K=\{1,2\}$ steps; *speculative* mode runs MRP with K=3 steps.

MATH500 (0-shot, CoT) [Hendrycks et al., 2021, Lightman et al., 2024]; and *(ii) Code Generation* – HumanEval [Chen et al., 2021] and MBPP [Austin et al., 2021b], both evaluated zero-shot. Following SDAR, we use greedy decoding for all models (temperature 1.0, Top-K=1, Top-P=1.0) with a maximum generation length of 4096 tokens. The SDAR-Chat backbones we build on are the block-size-16 variants, so block length and number of denoising steps per block are both set to 16.

System. We implement MRP in the SGLang inference engine [Zheng et al., 2024]. Unless otherwise stated, all the experimental numbers are obtained on a single NVIDIA H100 GPU.

Denoising Mode. We evaluate MRP under both denoising modes described in SDAR [Cheng et al., 2025]. In *static denoising*, each step unmasks a fixed number r of positions per block, selecting the highest-confidence predictions. In *dynamic denoising*, all positions with confidence above a threshold τ are unmasked at once, with a minimum of one position revealed per step to guarantee progress.

4.2 Residual Modeling

We start by comparing two modeling variants trained with identical protocols: (i) *Residual Modeling*, which predicts $\boldsymbol{\delta}_t = \log p(\boldsymbol{x}_0|\boldsymbol{x}_{t-1}) - \log p(\boldsymbol{x}_0|\boldsymbol{x}_t)$ on top of the frozen backbone logits, and (ii) *Direct Modeling*, which is distilled to predict $\log p(\boldsymbol{x}_0|\boldsymbol{x}_{t-1})$ directly without the residual reparameterization. Table 1 reports GSM8K accuracy under direct decoding (τ=1.0) at $K \in \{1,2,3,4\}$ MRP steps across all three SDAR scales. The two objectives are close at K=1 (gaps of +0.2, +3.8, and −0.4 on 1.7B, 4B, and 8B), but the gap opens sharply as K increases: at K=2 residual training already leads by +68.0 on 4B and +17.2 on 8B, and by K=4 the direct model has collapsed at every scale (Δ of +12.9, +55.3, and +73.1). The reason is that direct distillation must reproduce the full log-density at every iteration, so per-step errors compound across steps, whereas residual training

only models a low-magnitude correction and extrapolates more reliably to deeper MRP steps. This substantiates our motivation for residual modeling.

4.3 MRP Inference under Static Denoising

Table 2 reports the static denoising results. For each task we compare the unmodified backbone (*Base*, $r{=}1$ token unmasked per step) against our *Direct* mode at $K{\in}\{1, 2\}$ MRP steps and our *Spec* mode at $K{=}3$, and report task accuracy together with end-to-end throughput.

Speculative mode preserves quality at meaningful speedup. In every setting, the speculative mode has no quality drop as expected. It delivers 1.17–$1.40\times$ speed up in throughput across all configurations, with the largest absolute speedups on the larger backbone where the verification cost is best amortized (i.e., $1.40\times$ on 8B/GSM8K). Appendix F further reports acceptance rates, per-step cost and memory breakdowns, and exact-match rates against the backbone.

Direct mode buys further throughput at a controlled quality cost. At $K{=}1$, the direct mode has close quality compared to backbone (e.g., 88.6 vs. 90.3 GSM8K at 4B; 90.1 vs. 90.9 at 8B) while running 1.43–$1.61\times$ faster. Pushing to $K{=}2$ adds another step of speedup (1.53–$1.91\times$) with a modest accuracy hit concentrated on code benchmarks (e.g., HumanEval). The clean monotonic tradeoff between K and accuracy lets practitioners pick the operating point: *Speculative Mode* when quality is non-negotiable, *Direct Mode* ($K{=}1$ or 2) when latency dominates.

4.4 MRP Inference under Dynamic Denoising

Table 3 evaluates the remasking mode (Algorithm 4) under threshold-based dynamic denoising. We sweep the unmasking threshold $\tau \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ on SDAR-1.7B/4B/8B. For each τ we compare plain threshold-based *Dynamic* decoding against *MRP Remask*, reporting task accuracy on GSM8K, MATH500, HumanEval, and MBPP together with the accuracy change over Dynamic at the matched τ. As described in Section 3.4, the reveal and the remasking share the same τ, so it introduces no tuning beyond what dynamic decoding already requires.

The intuition is that a single forward pass gives a noisy confidence estimate: a position may clear τ once by chance, but a position that stays above τ across two consecutive estimates has more stably crossed the threshold. MRP Remask keeps only these doubly-confirmed positions and revokes the rest, filtering out the premature reveals that a single noisy estimate would commit. We deliberately study low thresholds: these make the backbone unmask many positions per step, precisely the regime where simultaneous reveals are most likely premature and MRP remasking has the most to correct.

In addition, in our experiments we adopt the *max-accept* trick from WINO [Hong et al., 2025], which caps the number of positions unmasked per step at $m = \min(\max(\lfloor 0.7\, n \rfloor, 5),\ L)$, where n is the number of still-masked positions in the block and L denotes the block size; when more than m positions clear τ, we keep the m highest-confidence ones.

Remasking recovers the accuracy lost to aggressive unmasking. At low thresholds the backbone commits many positions in a single step, before their neighbors are revealed, and Dynamic accuracy degrades sharply as a result. MRP Remask reverses most of this loss: at $\tau = 0.5$ it improves accuracy by at least 11 points at every task and scale, and by up to $+22.6$ (e.g., $32.3 \to 54.9$ on 8B HumanEval, $63.4 \to 81.1$ on 4B GSM8K, $26.0 \to 37.4$ on 1.7B MATH500). The improvement shrinks as τ rises: by $\tau = 0.9$ the backbone already unmasks conservatively, leaving little for MRP remasking to revoke, and the gain narrows to near zero. MRP remasking thus acts exactly where it is needed—when the backbone is over-eager—and stays out of the way otherwise.

Reasoning improves uniformly; the rare regressions are small and confined to code. On the reasoning benchmarks (GSM8K, MATH500), every operating point improves, without exception across thresholds and model scales, indicating that the residual reliably identifies premature reveals for natural-language reasoning. On code generation the gains are equally large at aggressive thresholds, but a few minor regressions surface at conservative thresholds (e.g., -1.8 and -1.2 on 4B HumanEval at $\tau = 0.8$ and 0.9, and -0.8 on 4B MBPP at $\tau = 0.9$). Code tokens undergo sharper, less predictable distributional shifts between denoising steps, which makes the residual harder to estimate; where Dynamic is already near its ceiling, an imperfect estimate can occasionally revoke a correct reveal. The effect is small and confined to high τ, where remasking is least needed.

Model	τ	GSM8K		MATH500		HumanEval		MBPP	
		Dyn.	Remask	Dyn.	Remask	Dyn.	Remask	Dyn.	Remask
1.7B	0.5	41.6	59.1 (+17.5)	26.0	37.4 (+11.4)	17.7	28.7 (+11.0)	26.9	41.3 (+14.4)
	0.6	56.3	67.0 (+10.7)	33.4	40.4 (+7.0)	31.7	43.3 (+11.6)	42.4	49.0 (+6.6)
	0.7	65.4	71.8 (+6.4)	39.4	48.6 (+9.2)	40.9	45.1 (+4.2)	49.8	51.0 (+1.2)
	0.8	70.6	75.4 (+4.8)	47.4	52.0 (+4.6)	45.7	48.8 (+3.1)	51.8	51.8 (0.0)
	0.9	76.2	77.3 (+1.1)	51.2	57.0 (+5.8)	49.4	52.4 (+3.0)	53.7	54.1 (+0.4)
4B	0.5	63.4	81.1 (+17.7)	44.2	58.4 (+14.2)	32.3	53.1 (+20.8)	38.5	50.6 (+12.1)
	0.6	76.4	85.5 (+9.1)	53.6	61.4 (+7.8)	49.4	57.9 (+8.5)	49.8	57.2 (+7.4)
	0.7	84.6	88.5 (+3.9)	60.4	65.6 (+5.2)	60.4	62.2 (+1.8)	61.1	63.4 (+2.3)
	0.8	87.9	90.1 (+2.2)	66.8	70.6 (+3.8)	64.6	62.8 (−1.8)	63.8	64.2 (+0.4)
	0.9	88.5	90.1 (+1.6)	69.0	70.6 (+1.6)	67.1	65.9 (−1.2)	65.4	64.6 (−0.8)
8B	0.5	67.9	82.3 (+14.4)	45.2	58.0 (+12.8)	32.3	54.9 (+22.6)	34.6	49.4 (+14.8)
	0.6	79.6	86.8 (+7.2)	54.8	63.8 (+9.0)	48.8	63.4 (+14.6)	48.3	59.9 (+11.6)
	0.7	85.9	89.0 (+3.1)	60.8	69.0 (+8.2)	64.6	72.6 (+8.0)	54.9	60.3 (+5.4)
	0.8	89.3	91.0 (+1.7)	68.0	70.2 (+2.2)	74.4	75.0 (+0.6)	62.3	66.5 (+4.2)
	0.9	90.8	91.4 (+0.6)	70.0	72.0 (+2.0)	75.0	75.0 (0.0)	66.9	68.5 (+1.6)

Table 3: MRP remasking improves the accuracy of low-threshold dynamic decoding. For each SDAR backbone (1.7B/4B/8B) and unmasking threshold τ, we report task accuracy (%, ↑) for plain threshold-based *Dynamic* decoding (*Dyn.*; our baseline) and for *MRP Remask* (*Remask*; our approach), which reuses the MRP-corrected confidence to remask over-eager reveals with the *same* τ for both the reveal and the remask (no extra tuning). Next to each Remask value we report its accuracy change over Dynamic at matched τ (green: gain, red: loss).

The benefit is consistent across model scale. All three backbones follow the same profile—large gains at low τ that taper to near zero at high τ—so remasking does not wash out as the backbone strengthens: even the 8B model gains $+14.4$ on GSM8K and $+22.6$ on HumanEval at $\tau = 0.5$. In effect, remasking lets a practitioner run at an aggressive, low-latency threshold while retaining most of the accuracy of a conservative one, at the cost of a single additional MRP forward per step; we report the corresponding decoding throughput in Table 5 (Appendix D).

5 Conclusion and Future Work

We introduced Multi-token Residual Prediction (MRP), a lightweight module that accelerates diffusion language model inference by predicting the logit residual between adjacent denoising steps. MRP supports three inference modes: direct decoding for a tunable quality–speed tradeoff, speculative decoding for lossless acceleration, and remasking decoding for improved accuracy under aggressive low-threshold decoding. Experiments on SDAR models at three scales demonstrate up to $1.40\times$ lossless speedup in SGLang, with consistent gains across reasoning and code generation benchmarks. All training data, code, and model weights are publicly available.

Several directions remain open. First, the threshold schedule could be co-designed with MRP: remasking already recovers most of the accuracy lost at aggressive thresholds (Section 4.4), so a jointly tuned schedule could push throughput further at matched accuracy. Second, MRP uses a fixed number of residual steps K; an adaptive schedule that runs more MRP steps when the backbone unmasks few tokens, and skips MRP when it already makes rapid progress, would better match compute to difficulty. Third, MRP degrades more on code than on reasoning, as code tolerates single-token errors poorly (Appendix F.5); domain-aware training or architectural changes could narrow this gap between the two domains.

Two further directions concern the theory and scope of this work. Our residual analysis rests on a Lipschitz assumption about the backbone, and a sharper theoretical characterization of how the residual evolves across denoising steps is left for future work. Our experiments also cover only the SDAR block-diffusion family; extending MRP to other DLM families, such as LLaDA [Nie et al., 2025] and Dream [Ye et al., 2025], is a natural next step.

Acknowledgments and Disclosure of Funding

This work was supported in part through STCSM 25PJA108 and NYU IT High Performance Computing resources, services, and staff expertise. We gratefully acknowledge Nous Research and Modal for providing the GPU compute used in this work.

References

Sudhanshu Agrawal, Risheek Garrepalli, Raghavv Goel, Mingu Lee, Christopher Lott, and Fatih Porikli. Spiffy: Multiplying diffusion LLM acceleration via lossless speculative decoding. *arXiv preprint arXiv:2509.18085*, 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. In *ICLR*, 2025.

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

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

BAAI. Infinity Instruct: Scaling instruction selection and synthesis to enhance language models. https://huggingface.co/datasets/BAAI/Infinity-Instruct, 2024.

Wenrui Bao, Zhiben Chen, Dan Xu, and Yuzhang Shang. Learning to parallel: Accelerating diffusion large language models via learnable parallel decoding. In *ICLR*, 2026.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Jiuhai Chen, Rifaa Qadri, Yuxin Wen, Neel Jain, John Kirchenbauer, Tianyi Zhou, and Tom Goldstein. GenQA: Generating millions of instructions from a handful of prompts. *arXiv preprint arXiv:2406.10323*, 2024a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. δ-DiT: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024b.

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

Jacob K. Christopher, Brian R. Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative diffusion decoding: Accelerating language generation through diffusion. In *NAACL*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yifeng Gao, Ziang Ji, Yuxuan Wang, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Self speculative decoding for diffusion large language models. *arXiv preprint arXiv:2510.04147*, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

Feng Hong, Geng Yu, Yushi Ye, Haicheng Huang, Huangjie Zheng, Ya Zhang, Yanfeng Wang, and Jiangchao Yao. Wide-in, narrow-out: Revokable decoding for efficient and effective DLLMs. *arXiv preprint arXiv:2507.18578*, 2025.

Yuezhou Hu, Harman Singh, Monishwaran Maheswaran, Haocheng Xi, Coleman Hooper, Jintao Zhang, Aditya Tomar, Michael W. Mahoney, Sewon Min, Mehrdad Farajtabar, et al. Residual context diffusion language models. *arXiv preprint arXiv:2601.22954*, 2026.

Daniel Israel, Guy Van den Broeck, and Aditya Grover. Accelerating diffusion LLMs via adaptive parallel decoding. In *NeurIPS*, 2025.

Guanghao Li, Zhihui Fu, Min Fang, Qibin Zhao, Ming Tang, Chun Yuan, and Jun Wang. DiffuSpec: Unlocking diffusion language models for speculative decoding. *arXiv preprint arXiv:2510.02358*, 2025a.

Pengxiang Li, Yefan Zhou, Dilxat Muhtar, Lu Yin, Shilin Yan, Li Shen, Soroush Vosoughi, and Shiwei Liu. Diffusion language models know the answer before decoding. In *NeurIPS*, 2025b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *ICML*, 2024.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *NeurIPS*, 2025c.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. 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. *arXiv preprint arXiv:2412.19437*, 2024.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *ICML*, 2024.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *CVPR*, 2024.

Amr Mohamed, Yang Zhang, Michalis Vazirgiannis, and Guokan Shang. Fast-decoding diffusion language models via progress-aware confidence schedules. *arXiv preprint arXiv:2512.02892*, 2025.

Pravin Nair. Softmax is $1/2$-Lipschitz: A tight bound across all ℓ_p norms. *arXiv preprint arXiv:2510.23012*, 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.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *NeurIPS*, 2024.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. FORA: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024.

Yeongbin Seo, Dongha Lee, Jaehyung Kim, and Jinyoung Yeo. Fast and fluent diffusion language models via convolutional decoding and rejective fine-tuning. In *NeurIPS*, 2025.

Teknium. OpenHermes 2.5: An open dataset of synthetic data for generalist LLM assistants. https://huggingface.co/datasets/teknium/OpenHermes-2.5, 2024.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The Mamba in the Llama: Distilling and accelerating hybrid models. In *NeurIPS*, 2024.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. Diffusion LLMs can do faster-than-AR inference via discrete diffusion forcing. In *ICLR*, 2026.

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. *arXiv preprint arXiv:2509.26328*, 2025.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *ICLR*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H. Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, et al. SGLang: Efficient execution of structured language model programs. In *NeurIPS*, 2024.

A Related Work

A.1 Diffusion Language Models

Discrete-sequence diffusion was formalized by D3PM [Austin et al., 2021a], whose absorbing-state ([MASK]) variant seeded masked diffusion language modeling. SEDD [Lou et al., 2024] and MDLM [Sahoo et al., 2024] subsequently sharpened the training objective, with MDLM's Rao-Blackwellized loss reducing masked diffusion training to a weighted mixture of masked language modeling losses. This recipe enabled the first wave of large-scale DLMs.

LLaDA [Nie et al., 2025] scaled masked diffusion to 8B parameters and matched strong AR LLMs on broad reasoning benchmarks. BD3-LM [Arriola et al., 2025] introduced a block-causal design that generates tokens block-by-block with discrete diffusion within each block, addressing fixed-length generation and the absence of KV caching. This paradigm has since become dominant, with Dream 7B [Ye et al., 2025], D2F [Wang et al., 2026], and most recently SDAR [Cheng et al., 2025] further validating the recipe at scale. SDAR is particularly relevant to our work: it lightly adapts a pre-trained AR backbone into a block-diffusion sampler, and serves as the backbone for all experiments in this paper. Our contribution is orthogonal to these efforts: we accelerate *inference* of any such backbone rather than proposing a new training paradigm.

A.2 Fast Decoding for Diffusion Language Models

As discussed in Section 1, naively unmasking many tokens per step degrades quality because the unmasking decisions at different positions are made simultaneously from a single forward pass. Existing approaches to accelerating DLM inference can be organized into two families: *threshold-based* methods, which use the backbone's own per-token confidence as the unmasking criterion, and *speculative* methods, which introduce auxiliary mechanisms such as drafting, learned predictors, or structural priors to decode additional tokens beyond what confidence alone would permit.

Threshold-based methods. Fast-dLLM [Wu et al., 2026] established the standard recipe: at each denoising step, unmask every position whose top-1 confidence exceeds a fixed threshold τ, and enable a KV cache for block-diffusion models. Subsequent work refines this recipe along several axes. WINO [Hong et al., 2025] allows the bidirectional context to re-mask previously committed tokens in later steps, relaxing the irrevocable commitment of Fast-dLLM.

Prophet [Li et al., 2025b] takes the opposite approach, committing all remaining tokens at once when a global confidence margin indicates that further refinement would be wasteful. SchED [Mohamed et al., 2025] generalizes the static threshold to a progress-aware schedule $\tau(t)$ that aggregates full-span logit margins. Despite these advances, all threshold-based methods rely on backbone confidence as the sole unmasking signal, and therefore cannot decode tokens whose confidence has not yet crossed the threshold, even when the residual suggests they are already near their final value.

Speculative methods. A complementary line of work introduces explicit drafting mechanisms. SpecDiff [Christopher et al., 2025] uses a discrete diffusion model as a drafter inside a speculative decoding loop with an autoregressive verifier, while DiffuSpec [Li et al., 2025a] reverses this arrangement, using a DLM to draft for AR verifiers via causal-consistency path search. Within the DLM family itself, Spiffy [Agrawal et al., 2025] drafts block states from the model's own distribution and verifies them through a directed draft graph, and SSD [Gao et al., 2025] eliminates the auxiliary drafter entirely by using the backbone as both drafter and hierarchical-tree verifier in one pass.

Learn2PD [Bao et al., 2026] takes a different approach, training a lightweight filter that predicts whether each draft token already matches its final committed value, avoiding unnecessary recomputation. APD [Israel et al., 2025] and Conv/R2FT [Seo et al., 2025] explore orthogonal directions: APD modulates the joint decoding distribution, while Conv/R2FT exploits positional locality through a convolutional decoding window over the masked positions.

Feature caching in continuous diffusion. A related line of work in image diffusion also exploits the similarity between adjacent denoising steps. DeepCache [Ma et al., 2024] caches high-level U-Net features and recomputes only the shallow U-Net blocks at intermediate steps. FORA [Selvaraju et al., 2024] reuses the attention and MLP outputs of a Diffusion Transformer across consecutive steps,

refreshing the cache with a full forward every few steps. Δ-DiT [Chen et al., 2024b] caches, within a single step, the offset between the output and the input of a contiguous segment of DiT blocks, and adds this offset in place of running the segment at the next step; it applies the cache to a front or rear segment depending on the sampling stage.

These methods are training-free and approximate the backbone's *internal features* at a step that is still executed, so they lower the cost of each step but not the number of steps. MRP differs in both target and effect. Its residual is a difference *across* adjacent denoising steps in the output logits, whereas the Δ-DiT offset is a difference *within* one step between intermediate features. MRP is also trained to predict this residual, and each MRP step commits additional discrete tokens without a backbone forward, reducing the number of backbone calls needed to decode a block.

Position of our work. MRP sits at the intersection of these two families. Like threshold-based methods, it uses the backbone's confidence for verification, preserving plug-in compatibility with any block-diffusion backbone. Like speculative methods, it generates additional token candidates beyond what the threshold alone would yield, through a chain of lightweight residual heads whose proposals the backbone then accepts or rejects. The key distinction from prior speculative approaches such as SSD [Gao et al., 2025] and Learn2PD [Bao et al., 2026] is MRP's *residual* parameterization: rather than predicting the full next-step distribution or learning a binary accept/reject filter, MRP predicts only the small correction between adjacent denoising steps, which we show in Section 3 is decisive for keeping the module lightweight while matching backbone quality at high parallelism.

B Residual Magnitude Measurement (Figure 2)

This appendix specifies the protocol used to produce Figure 2, which compares the magnitude of the per-step residual against the magnitude of the underlying backbone state.

Models and data. We measure on three frozen SDAR-Chat backbones [Cheng et al., 2025] at the 1.7B, 4B, and 8B scales (block size B=16). Prompts are sampled from the GSM8K test set [Cobbe et al., 2021] with a fixed seed; questions are wrapped in the SDAR chat template with the prefix *"Let's think step by step."* and prompts longer than 1024 tokens are skipped. We collect statistics from the first four decoded blocks of each prompt and stop after at least 200 blocks per model.

Decoding. Each block of 16 positions is decoded with greedy, low-confidence-static unmasking, revealing one token per step (the position with the highest argmax-softmax confidence among masked positions). Decoding stops early on EOS tokens. This procedure traverses 17 block states $s = 0, 1, \ldots, 16$, where s counts the number of revealed positions in the current block.

What we record. At every state s we save both the backbone final-layer hidden states $\boldsymbol{h}^{(s)} \in \mathbb{R}^{B\times d}$ and the post-projection logits $\boldsymbol{\ell}^{(s)} \in \mathbb{R}^{B\times V}$, restricted to the current block's positions. For brevity we let $X^{(s)} \in \{\boldsymbol{\ell}^{(s)}, \boldsymbol{h}^{(s)}\}$ denote either tensor below.

Residual and reference quantities. For each $r \in \{1, \ldots, 16\}$ and each valid offset t, we compute the residual

$$\boldsymbol{\delta}_{t,r} \;=\; X^{(t+r)} - X^{(t)}, \qquad t = 0, 1, \ldots, 16 - r,$$

which is the change in X produced by revealing r additional tokens. The backbone reference is taken over the partially decoded states $X^{(s)}$ for $s = 1, \ldots, 16$.

Magnitude metric. Throughout the figure we report the root-mean-square magnitude per entry, $\|Y\|_{\mathrm{RMS}} = \sqrt{\frac{1}{|Y|}\sum_i Y_i^2}$, computed in float64 to avoid bf16 underflow. For each model, space (logits or hidden states), and value of r, we average $\|\boldsymbol{\delta}_{t,r}\|_{\mathrm{RMS}}$ over all collected (block, t) pairs; the dashed reference lines show the average of $\|X^{(s)}\|_{\mathrm{RMS}}$ over all collected (block, s) pairs.

C Implementation Details

Table 4 lists the full set of training hyperparameters used for all three model scales.

Hyperparameter	Value
Architecture	
Backbone (frozen)	SDAR-{1.7B, 4B, 8B}-Chat (block size 16)
MRP transformer layers D	3
Frozen modules	backbone, LM head, token embeddings
MRP init. std. σ_{init}	0.2
Objective	
Loss	forward KD on residual logits
KD temperature T_{KD}	1.0
Reveal mode	ground-truth, k=1 token per block
MRP unroll steps K	2
Per-step loss weighting	uniform
Optimization	
Optimizer	AdamW
Peak learning rate	1e$-$3
LR schedule	cosine with min-LR
Epochs	1
Sequence length	4096
Block length	16
Global / micro batch size	16 / 1
Gradient checkpointing	enabled
Data	
Dataset	SFTDatasetV3 [Wang et al., 2024]
Supervised tokens	assistant response tokens only

Table 4: Training hyperparameters used for all three model scales.

Model	τ	GSM8K		MATH500		HumanEval		MBPP	
		Dyn.	Remask	Dyn.	Remask	Dyn.	Remask	Dyn.	Remask
1.7B	0.5	975	792	1126	956	588	587	533	363
	0.6	881	672	1065	871	554	528	420	378
	0.7	820	651	985	818	462	555	506	389
	0.8	719	565	929	761	528	556	420	337
	0.9	627	491	784	639	480	512	375	297
4B	0.5	720	558	753	603	448	354	441	309
	0.6	641	513	728	593	449	437	269	297
	0.7	596	489	660	532	433	409	253	214
	0.8	523	422	599	490	400	400	235	206
	0.9	466	366	519	437	318	363	211	179
8B	0.5	507	416	532	469	354	324	253	199
	0.6	454	378	524	432	361	342	229	180
	0.7	419	340	479	394	370	334	210	166
	0.8	378	306	439	349	309	298	187	154
	0.9	313	257	380	312	326	277	167	134

Table 5: Decoding throughput (tokens/s) of MRP Remask vs. Dynamic decoding. Measured in SGLang on a single NVIDIA H100, for each SDAR backbone (1.7B/4B/8B) and unmasking threshold τ. *Dyn.* is plain threshold-based dynamic decoding and *Remask* is our MRP remasking mode. This is the throughput counterpart to the accuracy results in Table 3.

D Throughput of MRP Remasking

Table 5 reports decoding throughput (tokens per second, measured in SGLang on a single NVIDIA H100) for *Dynamic* decoding and *MRP Remask*, over the same backbones and thresholds as the accuracy results in Table 3. Remasking adds one MRP forward per step and commits fewer tokens per step (the revoked reveals), so it generally runs somewhat slower than Dynamic at matched τ: on the reasoning benchmarks it retains roughly 0.76–$0.88\times$ of the Dynamic throughput, while on the code benchmarks the ratio is more variable, ranging from modestly slower to occasionally faster. This throughput reduction is the efficiency cost of the accuracy gains in Table 3.

E MRP Depth Sweep on SDAR-1.7B

Holding the backbone fixed at SDAR-1.7B, we vary the number of MRP transformer layers $D \in \{1, 2, 3, 4, 8\}$. A deeper MRP module has more capacity to predict the residual but adds non-trivial cost per inference step. Figure 3 reports accuracy and throughput across the four benchmarks for both *direct* ($K \in \{1, 2\}$) and *speculative* ($K \in \{2, 3\}$) decoding modes.

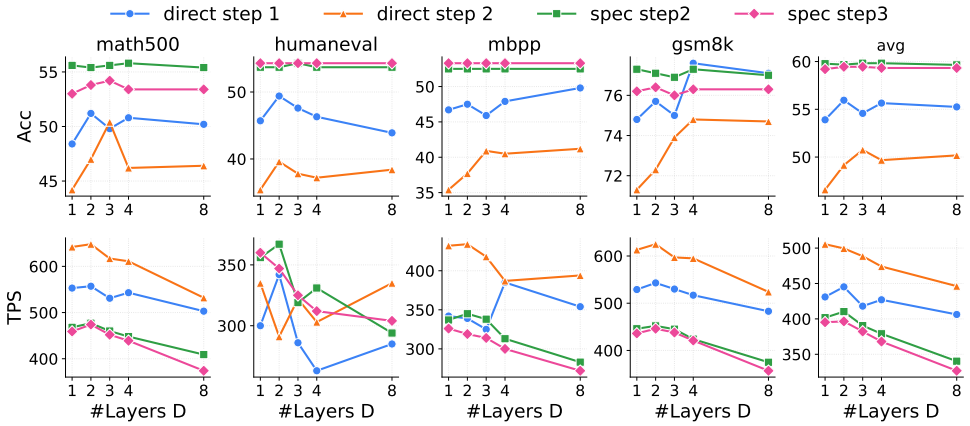

Figure 3: Effect of MRP depth on SDAR-1.7B. We sweep the number of MRP transformer layers $D \in \{1, 2, 3, 4, 8\}$ on the x-axis and report, per benchmark column (and an averaged column at the right), accuracy on the top row and decoding throughput on the bottom row. Each subplot draws four curves corresponding to the four decoding configurations: *direct* with $K \in \{1, 2\}$ MRP steps and *speculative* with $K \in \{2, 3\}$ MRP steps. The underlying numerical values are reported in Table 6.

A small module is enough. Adding MRP layers improves direct-decoding accuracy with diminishing returns: the gains are clearest for $K=2$ (the four-benchmark average rises by about four points from $D=1$ to $D=3$) and on GSM8K (where direct accuracy keeps climbing to a peak at $D=4$), but for $K=1$ they are small and non-monotonic, and speculative accuracy is essentially flat across depth. Throughput, by contrast, falls steadily as each added layer increases the per-step latency of the MRP forward pass, so the best speed–quality tradeoff sits at small depth—a few layers recover essentially all of the achievable accuracy, and the 8-layer variant only trades throughput for no meaningful accuracy gain over shallower MRP modules.

This is consistent with our theoretical characterization: the inter-step residual has limited complexity, and a small module suffices to capture the bulk of the signal. HumanEval is an exception: its short, single-function generations decode comparatively few tokens per problem, yielding the least stable throughput estimates across the five MRP depths we evaluate.

The optimal depth depends on the inference mode. In direct decoding, where MRP predictions are used without verification, deeper modules improve quality more noticeably: the accuracy gap between 1-layer and 3-layer MRP is larger than in speculative mode. In speculative decoding, verification corrects MRP errors, so a shallower, faster module can achieve comparable final quality at higher throughput. This suggests a guideline: use deeper MRP for direct decoding when quality is prioritized, and shallower MRP for speculative decoding when lossless acceleration is the goal.

Table 6 reports the raw accuracy and decoding throughput numbers underlying Figure 3. The MRP module depth is varied over $D \in \{1, 2, 3, 4, 8\}$ on top of a frozen SDAR-1.7B-Chat backbone, and we evaluate the four decoding configurations shown in the figure: *direct* with $K \in \{1, 2\}$ MRP steps and *speculative* with $K \in \{2, 3\}$ MRP steps. All other settings follow Table 4; throughput is reported in tokens per second on a single H100, and accuracy is the standard task metric (%).

F Additional Analyses

This section collects additional analyses of the MRP module: which input signals it needs and from which backbone layer (Appendices F.1 and F.2), where the speculative speedup comes from and what it costs in memory (Appendix F.3), what *lossless* means under finite-precision arithmetic

Depth D	Setting	GSM8K		MATH500		HumanEval		MBPP	
		Acc.	TPS	Acc.	TPS	Acc.	TPS	Acc.	TPS
1	Direct (K=1)	74.8	529	48.4	553	45.7	300	46.7	342
	Direct (K=2)	71.3	613	44.2	642	35.4	335	35.4	432
	Speculative (K=2)	77.3	446	55.6	468	53.7	356	52.5	337
	Speculative (K=3)	76.2	436	53.0	459	54.3	360	53.3	326
2	Direct (K=1)	75.7	543	51.2	557	49.4	342	47.5	339
	Direct (K=2)	72.3	625	47.0	648	39.6	291	37.7	434
	Speculative (K=2)	77.1	452	55.4	477	53.7	367	52.5	345
	Speculative (K=3)	76.4	446	53.8	474	54.3	347	53.3	319
3	Direct (K=1)	75.0	530	49.8	531	47.6	286	45.9	325
	Direct (K=2)	73.9	597	50.4	617	37.8	322	40.9	418
	Speculative (K=2)	76.9	445	55.6	460	54.3	319	52.5	338
	Speculative (K=3)	76.0	438	54.2	452	54.3	325	53.3	314
4	Direct (K=1)	77.6	517	50.8	543	46.3	263	47.9	385
	Direct (K=2)	74.8	595	46.2	611	37.2	303	40.5	387
	Speculative (K=2)	77.3	424	55.8	448	53.7	331	52.5	313
	Speculative (K=3)	76.3	421	53.4	439	54.3	312	53.3	300
8	Direct (K=1)	77.1	483	50.2	503	43.9	285	49.8	354
	Direct (K=2)	74.7	524	46.4	532	38.4	335	41.2	394
	Speculative (K=2)	77.0	375	55.4	409	53.7	294	52.5	283
	Speculative (K=3)	76.3	357	53.4	374	54.3	304	53.3	272

Table 6: Raw numerical results for the SDAR-1.7B MRP depth sweep plotted in Figure 3. Accuracy (Acc., %) and decoding throughput (TPS, tokens per second) are reported per benchmark for MRP depths $D \in \{1, 2, 3, 4, 8\}$ and the four decoding configurations. *Direct* uses MRP predictions without verification; *Speculative* uses the backbone to verify and correct MRP drafts.

(Appendix F.4), and why code generation is more sensitive to MRP errors than mathematical reasoning (Appendix F.5). Unless otherwise stated, all measurements use a single NVIDIA H100 GPU.

F.1 Input Components of the MRP Module

The MRP module reads two signals: the token embeddings $\boldsymbol{e}$ of the revealed sequence and the backbone hidden states $\boldsymbol{h}$ (Section 3.2). To isolate their contributions, we train SDAR-1.7B variants that receive only one of them, using the UltraChat-200K subset for all runs to keep the sweep affordable. Table 7 shows that neither signal suffices alone: fusing both raises the four-benchmark average from 46.8 ($\boldsymbol{e}$ only) and 45.0 ($\boldsymbol{h}$ only) to 50.2, and the gain is concentrated on code (42.1 vs. 32.9 and 28.7 on HumanEval, a gap of more than nine points).

Input	GSM8K	MATH500	HumanEval	MBPP	Avg.
$\boldsymbol{e}$ only	71.04	**50.0**	32.93	33.07	46.76
$\boldsymbol{h}$ only	68.92	49.0	28.66	33.46	45.01
$\boldsymbol{e} + \boldsymbol{h}$ (default)	**72.25**	48.0	**42.07**	**38.52**	**50.21**

Table 7: Input-component ablation of the MRP module on SDAR-1.7B. $\boldsymbol{e}$: token embeddings of the revealed sequence; $\boldsymbol{h}$: last-layer backbone hidden states. All variants are trained on the UltraChat-200K subset rather than the full SFTDatasetV3, so absolute numbers are not comparable to Table 2. Best value per column in bold.

F.2 Choice of Backbone Layer

We also vary which backbone layer supplies $\boldsymbol{h}$, and, following the multi-layer feature fusion of EAGLE-3 [Li et al., 2025c], fuse low-, mid-, and high-level layers (2, 14, 25). In Table 8, the intermediate layers and the fusion reach slightly higher GSM8K accuracy, but all of them lose on the four-benchmark average to the last layer, mostly through HumanEval. The last layer also suits the iterative loop: from the second MRP step onward, the module refines its own last-layer representation together with the accumulated residual, which an intermediate-layer input would not provide.

Backbone layer of h	GSM8K	MATH500	HumanEval	MBPP	Avg.
layer 28 (last, default)	72.25	**48.0**	**42.07**	38.52	**50.21**
layer 2	73.84	47.8	35.37	**41.63**	49.66
layer 14	74.00	47.0	35.37	36.96	48.33
layer 25	74.07	**48.0**	39.02	37.74	49.71
layers $\{2, 14, 25\}$ fused	**74.98**	46.0	37.80	38.13	49.23

Table 8: Backbone-layer ablation of the MRP input on SDAR-1.7B (28 layers). Each variant feeds $\boldsymbol{h}$ from the listed layer(s) together with $\boldsymbol{e}$, and is trained on the same UltraChat-200K subset as Table 7; the default row is the $\boldsymbol{e} + \boldsymbol{h}$ row there. Best value per column in bold.

F.3 Cost Breakdown and Memory Overhead

Table 9 explains the speedups in Table 2. In direct decoding with $K{=}1$, the MRP forward takes only 12–16% of decoding time, while each MRP step commits an extra token without a backbone call. In speculative decoding with $K{=}3$, 45–55% of drafted tokens are accepted, so each verification commits 2.15–2.42 tokens instead of one. The verification pass dominates the time budget (52–58%), but it also seeds the next iteration (Section 3.4), so its cost is shared by consecutive denoising steps.

MRP adds a small module next to the frozen backbone, so its memory cost should be modest. Table 10 reports peak GPU memory during decoding with HuggingFace `generate`, because SGLang preallocates the KV cache and would hide the difference. Across all three scales, MRP adds 7.3–9.5% over the backbone alone, and this overhead grows only slowly as the context length increases from 4k to 64k tokens, so MRP remains practical for long-context generation.

Model	Direct, $K{=}1$: time (%)		Spec., $K{=}3$: acceptance		Spec., $K{=}3$: time (%)		
	Backbone	MRP	Rate α	Length	Backbone	MRP ($3\times$)	Verify
1.7B	83.8	16.2	0.550	2.42	16.9	31.6	51.5
4B	87.8	12.2	0.450	2.15	17.2	25.1	57.7
8B	87.9	12.1	0.488	2.26	17.7	25.0	57.3

Table 9: Decoding cost breakdown on GSM8K. *Time* columns give the share of decoding time spent in each forward pass. In speculative mode, *Backbone* covers prefill and KV-cache updates, *MRP* covers all three drafting steps, and *Verify* is the backbone verification pass. *Rate* is the fraction of drafted tokens accepted; *Length* is the mean number of tokens committed per verification.

Model	4k	8k	16k	32k	64k
1.7B	4.27	4.72	5.63	7.45	11.07
1.7B w/ MRP	4.62 (+8.2%)	5.12 (+8.4%)	6.12 (+8.7%)	8.12 (+9.1%)	12.13 (+9.5%)
4B	8.89	9.46	10.62	12.93	17.56
4B w/ MRP	9.54 (+7.3%)	10.16 (+7.4%)	11.41 (+7.4%)	13.91 (+7.6%)	18.91 (+7.7%)
8B	15.87	16.45	17.61	19.92	24.55
8B w/ MRP	17.07 (+7.5%)	17.69 (+7.6%)	18.94 (+7.6%)	21.44 (+7.7%)	26.45 (+7.7%)

Table 10: Peak GPU memory (GB) during decoding at increasing context lengths, in bf16 with HuggingFace `generate`. Relative overhead of MRP over the backbone alone is shown in parentheses.

F.4 Exact Match of Speculative Decoding

We call speculative decoding *lossless* because every committed token equals the backbone's own arg-max prediction at that position (Section 3.4). Table 11 tests this by comparing whole generations against backbone-only decoding. In fp32, the two produce identical sequences on every GSM8K and HumanEval problem for all K, confirming that verification reproduces the backbone trajectory. Verification thus changes how many backbone passes run, but never which tokens are produced.

In bf16, 57.5–79.3% of sequences and 75.1–92.6% of tokens match. The mismatches are purely numerical: the two procedures execute different sequences of bf16 operations, so their logits differ

slightly, and a near-tied arg-max can flip. Once one token differs, the generations diverge. Speculative accuracy still matches the backbone in Table 2, since flips occur only between near-tied tokens.

Benchmark	Precision	K=1	K=2	K=3
GSM8K (200 problems)	bf16	57.5 (75.1)	57.5 (76.6)	59.5 (78.9)
	fp32	100.0	100.0	100.0
HumanEval (164 problems)	bf16	76.8 (91.1)	79.3 (92.6)	67.7 (87.7)
	fp32	100.0	100.0	100.0

Table 11: Sequence-level exact-match rate (%) between MRP speculative decoding and backbone-only decoding, for K drafting steps. Token-level match rates (%) are given in parentheses; in fp32 both are 100.0.

F.5 Sensitivity of Code to Mid-Sequence Errors

Section 4.3 shows that direct decoding loses more accuracy on code than on mathematics. To test whether code is simply less tolerant of a single wrong token, we take SDAR-1.7B decoding trajectories that solve the problem, replace one token at 25% or 75% of the response with a token that already appears earlier in the prefix, and let the backbone continue with bf16 sampling. As a control, we resume the same trajectories from 25% without injection; since bf16 sampling breaks logit ties at random, the continuation can still go wrong, so the control solves 84.3% of HumanEval and 92.9% of GSM8K problems rather than all of them. Relative to this control, Table 12 shows that one injected token drops HumanEval accuracy by 31.4 and 54.3 points, against 12.9 and 5.8 points on GSM8K.

Code is therefore far less forgiving: a misplaced identifier or delimiter usually breaks the program, and the damage grows later in the response, whereas a reasoning chain often recovers from a local error. Since direct decoding commits MRP predictions without verification, any residual error becomes such an injected token, which explains the larger code degradation. This suggests speculative or remasking decoding for code, where verification or revocation catches the error.

Benchmark	Control @25%	Inject @25%	Inject @75%
HumanEval	84.3	52.9 (−31.4)	30.0 (−54.3)
GSM8K	92.9	80.0 (−12.9)	87.1 (−5.8)

Table 12: Accuracy (%) after a single injected token on SDAR-1.7B, measured on generations the backbone originally solves. *Control* resumes the correct trajectory from 25% of the response with bf16 sampling and no injection, and falls below 100% because ties are broken at random; *Inject* replaces one token at 25% or 75% of the response with a token from the preceding prefix. Differences are relative to the control.

G Proofs and Derivations

We provide complete proofs for the results stated in Section 3. Throughout, we use the notation established in Section 2: $\boldsymbol{x}_t$ is the partially masked sequence at step t, f is the backbone, $\boldsymbol{\ell}_t = f(\boldsymbol{x}_t) \in \mathbb{R}^{L\times V}$ are the logits, $\boldsymbol{\pi}_t^i = \mathrm{softmax}(\boldsymbol{\ell}_t^i) \in \Delta^V$ is the predictive distribution at position i, $\mathcal{R}_t$ is the set of positions unmasked at step t, and $\boldsymbol{e}_v \in \mathbb{R}^{d_e}$ denotes the embedding of token v.

G.1 Proof of Proposition 1

Proof. The transition $\boldsymbol{x}_t \to \boldsymbol{x}_{t-1}$ replaces the `[MASK]` token at each position $j \in \mathcal{R}_t$ with the predicted token v_j, leaving all other positions unchanged. In embedding space, the perturbation $\boldsymbol{E}_{t-1} - \boldsymbol{E}_t$ is nonzero at exactly $|\mathcal{R}_t|$ rows, with Frobenius norm:

$$\|\boldsymbol{E}_{t-1} - \boldsymbol{E}_t\|_F = \sqrt{\sum_{j\in\mathcal{R}_t} \|\boldsymbol{e}_{v_j} - \boldsymbol{e}_{\texttt{[MASK]}}\|_2^2} \le \sqrt{|\mathcal{R}_t|}\cdot \max_{j\in\mathcal{R}_t} \|\boldsymbol{e}_{v_j} - \boldsymbol{e}_{\texttt{[MASK]}}\|_2. \tag{4}$$

Since f is κ-Lipschitz with respect to the input embedding matrix in Frobenius norm, and the Lipschitz condition applies to the full output tensor $\boldsymbol{\ell} \in \mathbb{R}^{L\times V}$ while we bound a single row, the

per-position logit change satisfies:

$$\|\boldsymbol{\ell}_{t-1}^i - \boldsymbol{\ell}_t^i\|_2 \leq \frac{\kappa\sqrt{|\mathcal{R}_t|}}{L} \cdot \max_{j \in \mathcal{R}_t} \|\boldsymbol{e}_{v_j} - \boldsymbol{e}_{\texttt{[MASK]}}\|_2. \tag{5}$$

To convert this into a total variation bound, we invoke the fact that softmax is $\frac{1}{2}$-Lipschitz in all ℓ_p norms [Nair, 2025]. In particular, for the $\ell_2 \to \ell_1$ case:

$$D_{\mathrm{TV}}(\boldsymbol{\pi}_{t-1}^i, \boldsymbol{\pi}_t^i) = \frac{1}{2}\|\boldsymbol{\pi}_{t-1}^i - \boldsymbol{\pi}_t^i\|_1 \leq \frac{1}{2}\|\boldsymbol{\ell}_{t-1}^i - \boldsymbol{\ell}_t^i\|_2. \tag{6}$$

Substituting (5) and using $\sqrt{|\mathcal{R}_t|} \leq |\mathcal{R}_t|$ for $|\mathcal{R}_t| \geq 1$, absorbing the factor of $\frac{1}{2}$ into κ:

$$D_{\mathrm{TV}}(\boldsymbol{\pi}_{t-1}^i, \boldsymbol{\pi}_t^i) \leq \kappa \cdot \frac{|\mathcal{R}_t|}{L} \cdot \max_{j \in \mathcal{R}_t} \|\boldsymbol{e}_{v_j} - \boldsymbol{e}_{\texttt{[MASK]}}\|_2. \tag{7}$$

□

G.2 Proof of Corollary 2

Proof. The bound in Proposition 1 tightens as denoising progresses through two mechanisms. First, the number of tokens revealed per step $|\mathcal{R}_t|$ is bounded by the number of remaining masked positions m_t, which decreases monotonically along the chain. In threshold-based decoding, later steps reveal fewer tokens because the remaining positions are those where the backbone had low confidence; in fixed-schedule decoding, $|\mathcal{R}_t|$ is typically non-increasing by design. Second, as more context is revealed, predictions at remaining positions become more concentrated — when $\max_v \pi_t^{i,v} \geq 1 - \eta$ for small η, no perturbation can shift the distribution by more than $O(\eta)$ in total variation, regardless of how large the backbone's Lipschitz constant κ is.

Combining these, the average per-position TV distance at step t satisfies:

$$\frac{1}{m_{t-1}} \sum_{i \notin \mathcal{R}_t} D_{\mathrm{TV}}(\boldsymbol{\pi}_{t-1}^i, \boldsymbol{\pi}_t^i) \leq \kappa D_{\max} \cdot \frac{m_t}{L}, \tag{8}$$

where $D_{\max} = \max_v \|\boldsymbol{e}_v - \boldsymbol{e}_{\texttt{[MASK]}}\|_2$. Since m_t/L decreases monotonically from 1 to 0, the residual magnitude bound decays along the chain toward zero. □

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction (Section 1) state the three claims that the body delivers: (i) a theoretical and empirical characterization of the inter-step logit residual in masked DLMs (Section 3, Figure 2, Proposition 1); (ii) the MRP module and its three inference modes (Sections 3.2–3.4); and (iii) up to $1.40\times$ lossless speedup in the static regime (Table 2) and up to 22.6 accuracy points recovered by remasking in the dynamic regime (Table 3), on SDAR-1.7B/4B/8B across GSM8K, MATH500, HumanEval, and MBPP. The scope (block-diffusion DLMs, SDAR backbones, single-H100 evaluation) is stated explicitly.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.

- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the experiments, appendices, and future work rather than in a dedicated section. Section 4.3 shows that the quality cost of direct decoding is task-dependent: reasoning accuracy tracks the backbone closely, while code generation degrades more, which Appendix F.5 attributes to code being less tolerant of single-token errors. Section 4.4 reports that remasking gains shrink toward zero at conservative thresholds and occasionally revoke correct reveals on code, and Appendix D reports its throughput cost. Appendix E notes diminishing returns past $D{=}4$ MRP layers, and Proposition 1 states the Lipschitz assumption on which the residual-decay argument rests. Evaluation is restricted to the SDAR block-diffusion family at three scales; we do not claim transfer to other DLM backbones, which Section 5 lists as future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Proposition 1 and Corollary 2 are stated in Section 3 with the Lipschitz assumption made explicit; complete proofs are provided in Appendix G. All theorems and equations are numbered and cross-referenced.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.

- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The MRP architecture (Section 3.2), training objective and algorithm (Section 3.3, Algorithm 1), and inference algorithms (Algorithms 2–4) are described in full. The evaluation setup (Section 4.1) fixes the public SDAR-Chat backbones, the SFTDatasetV3 training corpus, the block size, decoding temperature, top-k/top-p, and maximum generation length. Appendix C (Table 4) lists the full set of training hyperparameters; Appendix B documents the measurement protocol behind Figure 2. Our training and inference code and the trained MRP checkpoints are publicly released (see Question 5).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Training and inference code, including the SGLang integration, is released on GitHub, and the trained MRP checkpoints are released on Hugging Face; both are linked on the first page. All underlying assets (SDAR-Chat backbones, SFTDatasetV3 corpus, evaluation benchmarks) are public, and full details are given in Section 3, Algorithms 1–4, and Appendix C.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 specifies the models, dataset, training recipe (KD residual loss, T_{KD}=1.0, K=2 unroll, AdamW with peak learning rate 1e−3, cosine schedule, 1 epoch), evaluation benchmarks and protocol (greedy, temperature 1.0, top-k=1, top-p=1.0, max length 4096, block size 16), and the inference system. The full hyperparameter list is provided in Appendix C (Table 4).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Following the SDAR evaluation protocol we adopt (Section 4.1), we decode with top-k=1 (temperature 1.0, top-p=1.0). This is nearly deterministic, but in bf16 two tokens occasionally share the top logit, and the HuggingFace decoding path breaks such

ties at random (Appendix F.5), so repeated runs can differ slightly; we do not quantify this run-to-run variance. We do not report error bars over multiple training seeds because each MRP module is trained for a single epoch over a 12.4M-sample corpus, which is computationally prohibitive to repeat at three scales. Throughput numbers are reported as the mean over the full benchmark set.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All inference and decoding-throughput experiments are run on a single NVIDIA H100 GPU (Section 4.1). MRP modules at all three scales (1.7B, 4B, 8B) are trained on H100 GPUs using the optimization recipe in Appendix C (Table 4): AdamW, 1 epoch over SFTDatasetV3, sequence length 4096, global batch size 16, gradient checkpointing enabled. The backbone, LM head, and token embeddings are kept frozen, so only the D=3-layer MRP module is updated.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work studies inference-time acceleration of existing, publicly released diffusion language models on standard public benchmarks (GSM8K, MATH500, HumanEval, MBPP) and a public instruction-tuning corpus (SFTDatasetV3). It does not involve human

subjects, personal data, or new data collection, and we have reviewed the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: MRP is an inference-time acceleration mechanism for diffusion language models; it does not change the underlying model capabilities, training data, or output distribution (in speculative mode every committed token is the backbone's own prediction; see Appendix F.4). Any societal impact derives from the underlying DLM rather than from MRP itself, so the work has no direct societal-impact path of its own beyond the generic efficiency improvement of serving systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper releases no new pre-trained model, data scrape, or standalone generative asset. The released MRP checkpoints are small residual modules that run only on top of an already-public backbone (SDAR-Chat) and inherit whatever safeguards apply to that backbone; they do not unlock new capabilities or remove existing restrictions.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All third-party assets are cited where introduced and used in accordance with their upstream licenses: SDAR-Chat backbones [Cheng et al., 2025] (Apache-2.0), SFTDatasetV3 [Wang et al., 2024] (Apache-2.0; itself wraps OpenHermes-2.5 [Teknium, 2024], GenQA [Chen et al., 2024a], and Infinity-Instruct [BAAI, 2024]), SGLang [Zheng et al., 2024], and the evaluation benchmarks GSM8K [Cobbe et al., 2021] (MIT), MATH500 [Hendrycks et al., 2021, Lightman et al., 2024] (MIT), HumanEval [Chen et al., 2021] (MIT), and MBPP [Austin et al., 2021b] (CC-BY-4.0).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: No new datasets are introduced. The new assets are our code, released under the MIT license with a README documenting installation, training, and evaluation, and the trained MRP checkpoints for all three SDAR scales; both are linked in Question 5.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB review does not apply.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are the *subject* of this research (we accelerate inference of pre-trained diffusion LMs) but are not used as a non-standard component of the core methodology. LLM-based assistants were used only for writing, editing, and formatting the manuscript, which does not require declaration under the policy.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.