

Sequential Diffusion Language Models

Yangzhou Liu^{1,2*} Yue Cao^{1,2*} Hao Li^{2*} Gen Luo^{2*} Shihao Wang⁸
Zhe Chen^{1,2} Weiyun Wang^{4,2} Xiaobo Liang⁶ Biqing Qi² Lijun Wu²
Changyao Tian^{5,2} Yanting Zhang^{7†} Yuqiang Li² Tong Lu¹ Yu Qiao²
Jifeng Dai^{3,2} Wenhai Wang^{5,2}

¹Nanjing University, China ²Shanghai AI Laboratory, China

³Tsinghua University, China ⁴Fudan University, China

⁵The Chinese University of Hong Kong, China ⁶Soochow University, China

⁷Donghua University, China ⁸Hong Kong Polytechnic University, China

*Equal contribution †Corresponding author

Abstract

Diffusion language models (DLMs) are theoretically efficient but are limited by fixed-length decoding and poor compatibility with KV caches. Block diffusion mitigates these issues, yet still enforces a fixed block size and costly training. We introduce Next Sequence Prediction (NSP), which unifies next token and next block prediction, enabling the model to adaptively determine generation length at each step. NSP reduces to standard next token prediction when the length is one. Building on NSP, we propose the Sequential Diffusion Language Model (SDLM), which retrofits pretrained autoregressive language models (ALMs) at minimal cost. Specifically, SDLM performs diffusion inference within fixed-size mask blocks, but dynamically decodes contiguous subsequences based on model confidence, thereby preserving KV-cache compatibility and improving robustness to the varying uncertainty and semantics across the sequence. SDLM matches or surpasses strong autoregressive baselines with only 3.5M training samples, while achieving $2.0 \times \sim 3.4 \times$ higher throughput than Qwen2.5 across three model scales (3B, 7B, 32B) with confidence-based decoding. Furthermore, when sampling with self-speculative decoding, SDLM attains up to $5.7 \times$ end-to-end speedup measured as wall-clock tokens per second (TPS). Notably, SDLM-32B achieves performance comparable to Qwen2.5-32B with up to $2.5 \times$ speedup, demonstrating the scalability of our approach. Project page and codes: <https://github.com/OpenGVLab/SDLM>

1 Introduction

In recent years, diffusion models have made rapid progress in computer vision, showing strong empirical results in areas such as image generation (Ho et al., 2020; Rombach et al., 2022) and robot control (Chi et al., 2025; Kapelyukh et al., 2023). This successful paradigm has recently emerged as a po-

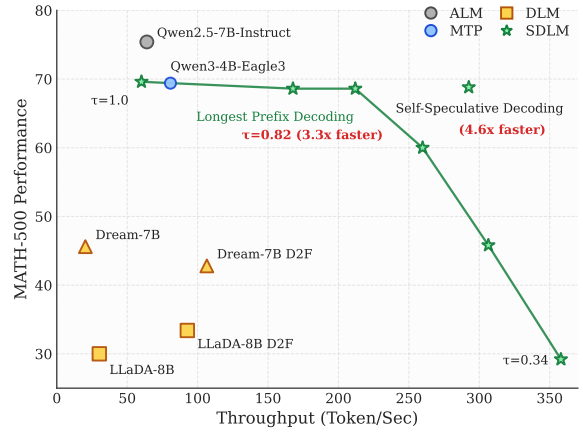


Figure 1: **MATH-500 accuracy vs. throughput across 7B/8B-scale model families.**

tential solution for language modeling, *i.e.*, diffusion language models (DLMs). Compared to autoregressive language models (ALMs), DLMs generate tokens in parallel through a denoising process, offering higher theoretical parallelism. However, DLMs are also criticized for their fixed decoding length and limited compatibility with KV cache (Radford et al., 2019).

To address these limitations, a natural approach is to combine the benefits of DLMs and ALMs, similar to existing efforts like Block Diffusion (Arriola et al., 2025). Specifically, Block Diffusion reformulates the next token prediction of ALMs as next block prediction, where tokens in each block are decoded in a diffusion manner. In this case, Block Diffusion preserves the autoregressive property for flexible prediction while leveraging diffusion-style parallelism for efficiency.

Despite its effectiveness, Block Diffusion still has two practical limitations. First, the block size is fixed, meaning the model must predict a constant number of tokens at each step regardless of the varying difficulty and semantics across the sequence. As shown in Figure 2(c), a rigid block size

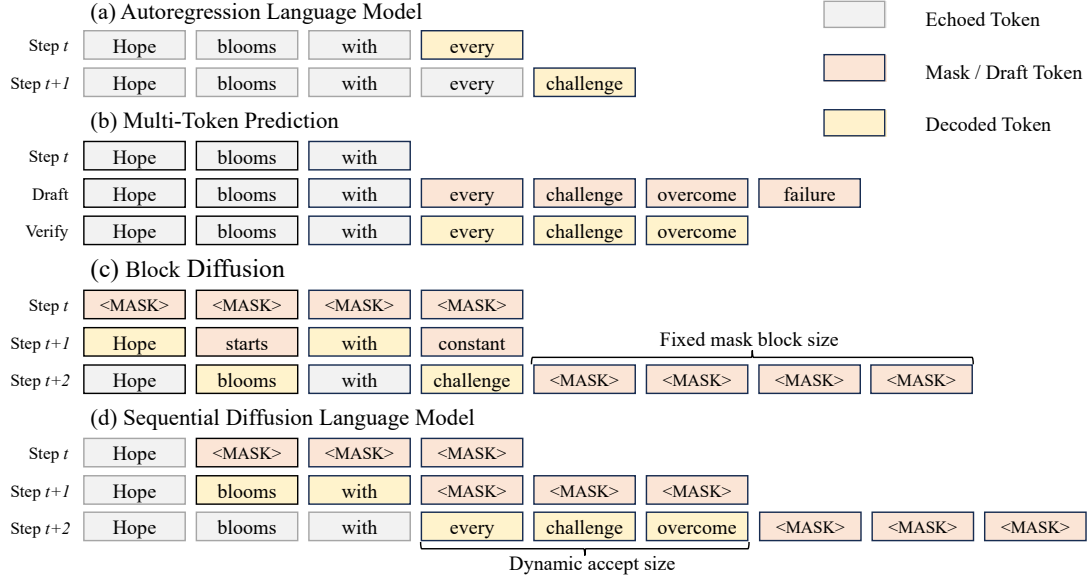


Figure 2: **Comparison of decoding paradigms.** (a) ALMs: one token per step. (b) MTPs: draft multiple tokens then verify. (c) DLMs (e.g. Block Diffusion): denoise a fixed-size block before moving to the next. (d) SDLM (Ours): dynamically accept a variable-length prefix per block.

easily leads to inaccurate predictions for tokens that depend heavily on preceding context. Second, its random masking objective samples mask ratios over a continuous range, yielding a sparser training signal that requires far more epochs to converge compared with full masking; it also complicates *[EOS]* token handling when the sequence length is not a multiple of the block size. Together, these factors increase training costs and create obstacles for scaling to larger models.

In this paper, we introduce *Next Sequence Prediction* (NSP), a general form of next token and next block prediction. Specifically, NSP autoregressively predicts variable-length future sequences (Figure 2(d)), where each block of tokens is decoded in a single diffusion step. The predicted length adapts dynamically to local semantic difficulty, ranging from a single token to a full block. When the block size is 1, NSP becomes standard next token prediction, enabling low-cost adaptation from pretrained ALMs.

Based on the principle of NSP, we propose *Sequential Diffusion Language Models* (SDLMs) with training and inference strategies tailored to variable-length prediction. As shown in Figure 3, SDLMs are developed based on pretrained ALMs, employing a parallel block training approach to extend next token prediction to next sequence prediction. In parallel block training, we use a custom attention mask that makes the prefix and the current

block visible to each prediction window, enabling parallel training over multiple future blocks. During inference, SDLM predicts a fixed-length block at each step and then dynamically decodes a continuous subsequence via a confidence scheme based on threshold or verification. This dynamic decoding strategy yields a favorable speed–performance trade-off, which we illustrate on MATH-500 in Figure 1. Unlike the concurrent work of Samragh et al. (2025), which uses gated LoRA with next token and multi token losses, we use NTP cross-entropy for full supervised fine-tuning and pair bidirectional attention with confidence decoding, without extra heads.

To validate our approach, we construct LLMs at different scales and conduct experiments on 13 diverse benchmarks across general, math, knowledge and coding tasks. Experiments show that our SDLMs achieve comparable performance to existing ALMs at higher inference speed. With longest-prefix decoding, SDLMs deliver $2.0\times\sim 3.4\times$ speedup across three model scales (3B, 7B, 32B) on code and math benchmarks. With self-speculative decoding, the speedup further increases to $2.1\times\sim 5.7\times$ (measured as end-to-end wall-clock TPS on a matched HuggingFace `.generate()` backend), where SDLM-3B reaches $5.7\times$ on math and $4.3\times$ on code. Compared to existing representative DLMs, our SDLMs show advantages in the evaluated settings for performance, efficiency,

and training costs. For example, SDLM-3B outperforms DLMs like Dream-7B (Ye et al., 2025) and LLaDA-8B (Nie et al., 2026) across multiple benchmarks, while requiring less training compute and yielding higher inference speed. The scalability of SDLMs is further evaluated on larger model scales, *i.e.*, Qwen-2.5-32B, requiring only 3.5M training samples. In summary, our contributions are threefold:

- We introduce *Next Sequence Prediction* (NSP) as a general form of next token prediction and next block prediction. NSP not only combines the advantage of autoregressive models and diffusion models, but also overcomes the limitations in existing block diffusion models, *i.e.*, the fixed block size and scalability.
- Based on NSP, we deploy *Sequential Diffusion Language Models* (SDLMs) through a parallel block training method. SDLMs employ a customized attention mask where each block is visible to its prefix and itself, enabling parallel training and dynamic variable-length sequence generation via threshold- or verification-based selection.
- Extensive experiments demonstrate the effectiveness and efficiency of SDLMs against existing ALMs and DLMs across three model scales (3B, 7B, 32B). In particular, with only 3.5M training samples, SDLM-32B achieves comparable performance and up to $2.5\times$ speedup over Qwen2.5-32B.

2 Related Work

2.1 Autoregressive LMs and Multi-Token Prediction

Autoregressive language models (ALMs) (Radford et al., 2018; OpenAI, 2022, 2024; Touvron et al., 2023; Grattafiori et al., 2024; Bai et al., 2023; Yang et al., 2024, 2025; DeepMind, 2025; xAI, 2025; Anthropic, 2023; Liu et al., 2024) generate tokens one at a time via next token prediction (NTP). Despite strong task performance, this sequential decoding requires one forward pass per token, fundamentally limiting throughput even with a KV cache.

Multi-token prediction (MTP) (Cai et al., 2024; Gloeckle et al., 2024; Liu et al., 2024) mitigates this by predicting multiple future tokens via additional output heads, combined with speculative decoding (Xia et al., 2023; Stern et al., 2018) to

reduce forward steps. Representative systems include Medusa (Cai et al., 2024), DeepSeek-V3 (Liu et al., 2024), and Eagle-3 (Li et al., 2025).

Unlike MTP, which attaches isolated prediction heads to an autoregressive backbone, our method performs blockwise sequence prediction with full intra-block interaction.

2.2 Masked Diffusion Language Models

Recent diffusion models have shown increasing potential in language tasks. Masked discrete diffusion models (MDMs) (Zheng et al., 2023; Gong et al., 2025; Ou et al., 2025; Nie et al., 2025) have achieved perplexity comparable to ALMs. LLaDA (Nie et al., 2026) further scales MDMs to 8B parameters, matching state-of-the-art ALMs. Dream (Ye et al., 2025) adopts shifted prediction and autoregressive initialization, effectively reducing training costs while also delivering strong performance. Block Diffusion (Arriola et al., 2025) introduces block-level generation for variable-length decoding with KV cache reuse, while Gemini Diffusion (Google DeepMind, 2025) and Seed Diffusion (Song et al., 2025) further improve speed while narrowing the gap with ALMs.

Recent work has also focused on improving DLM inference efficiency. dKV-Cache (Ma et al., 2026), Fast-dLLM (Wu et al., 2026), and dLLM-Cache (Liu et al., 2025) accelerate diffusion decoding via approximate or cache-aware parallel decoding. Adaptive Parallel Decoding (Israel et al., 2026) dynamically adjusts the number of tokens decoded in parallel with the help of a small auxiliary autoregressive model, while D2F (Wang et al., 2026) reformulates diffusion decoding as an AR-diffusion hybrid with block-wise autoregressive generation and pipelined parallel decoding.

3 Method

3.1 Preliminaries and Notation

In autoregressive large language models (OpenAI, 2024; Grattafiori et al., 2024; Yang et al., 2025; DeepMind, 2025), text generation follows the next token prediction paradigm, where the model minimizes the cross-entropy loss:

$$\mathcal{L}_{\text{ALM}}(x; \theta) = -\mathbb{E}_x \left[\sum_{i=1}^L \log P_{\theta}(x^i | x^{<i}) \right], \quad (1)$$

where the model $P_{\theta}(\cdot | x^{<i})$ maximizes the conditional probability of the current token by leveraging

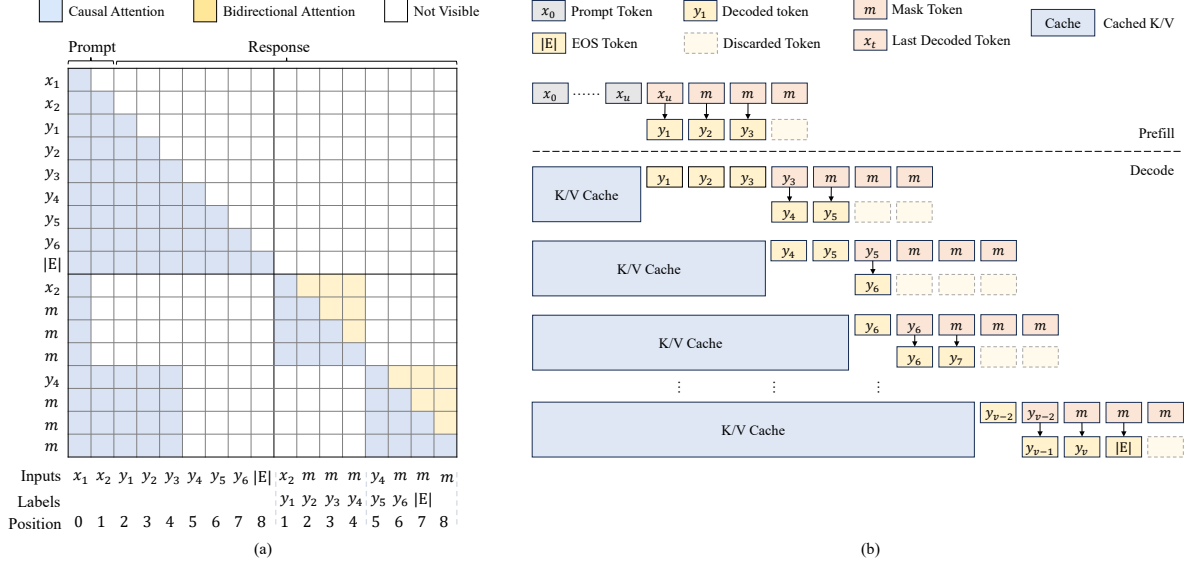


Figure 3: **Structured attention mask for parallel block training and sampling.** (a) Reordered input produces a causal prefix (top-left), cross-block visible prefix (bottom-left), and intra-block bidirectional region (bottom-right). (b) Confidence-based next sequence prediction: one anchor + $D-1$ mask tokens predict a block; the longest high-confidence prefix is emitted with KV reuse.

the preceding context $x^{<i} = x^1, \dots, x^{i-1}$. While this sequential formulation ensures strong local coherence, it imposes an inherent inference bottleneck: generating a sequence of length L requires L separate forward passes, which fundamentally limits inference throughput.

In contrast, diffusion language models (Ou et al., 2025; Nie et al., 2026; Ye et al., 2025) generate outputs by progressively denoising from a fully noisy state in parallel. A specialized variant, Block Diffusion (Arriola et al., 2025), constrains the diffusion to proceed sequentially in fixed-size blocks of length D , where each block is denoised over multiple time steps with a random masking schedule (full objective in Appendix A).

3.2 From Token to Sequence Prediction

In DLMS, the full sequence is predicted in parallel based on confidence scores, which can lead to premature and inaccurate predictions for later tokens. However, predictions at lower position indices generally benefit from more reliable contextual information and introduce less bias (Wang et al., 2024), suggesting that a left-to-right ordering of predictions is preferable. Moreover, the distribution of certainty and semantics varies across the sequence, calling for an adaptive decoding length rather than a fixed block size. To this end, we introduce *Next Sequence Prediction* (NSP), which dynamically adjusts the decoding length based on local semantic

difficulty, allowing the model to “sprint” through predictable patterns (e.g., common phrases), while “decelerating” at high-entropy junctions to maintain generation quality.

To instantiate the NSP paradigm, we propose Sequential Diffusion Language Models (SDLMS). SDLM is built upon two key design choices that simplify the diffusion process and enhance architectural compatibility:

Single-Step Full-Noise Denoising. From a theoretical perspective, SDLM can be viewed as a special case of masked diffusion that fixes $\alpha_t = 0$ in the Block Diffusion objective (Equation 6 in Appendix A), meaning all tokens in the prediction window are always fully masked. This *single-step, full-noise* simplification collapses the continuous-time expectation to a single point, reducing the weighted objective to a standard cross-entropy loss. Specifically, in the training phase, SDLM applies a masking probability of 1 to all tokens within a prediction window of length D , supervising the model to minimize the cross-entropy of all masked tokens:

$$\mathcal{L}(X; \theta) = -\mathbb{E}_{X, X_T} \left[\frac{1}{D} \sum_i \log P_\theta(X^i | x^{<(i-1)}, X_T^i) \right] \quad (2)$$

where $X^i = x^{i:(i+D)}$, $X_T^i = [x^{i-1}, [m], \dots, [m]]$, and i denotes a random index within the target se-

quence, accommodating the non-fixed starting positions caused by dynamic-length inference. Compared to random- t sampling, this full-noise objective of Equation 2 yields denser training signals aligned with the NTP objective of pretrained ALMs, avoiding the exponential mask-pattern space that random masking must cover, and ensures $[EOS]$ is consistently masked and predicted, eliminating the complex stopping criteria needed when random masking leaves $[EOS]$ unmasked. To better unify NTP and NSP, we continue to employ the standard one-position shift prediction objective of ALMs between input ids and labels.

Asymmetric Design with Causal Prefix. SDLM adopts an *Asymmetric Design*: strict causal attention for the historical context (ensuring full KV-cache compatibility) and a *bidirectional local window* within the target noise block X_T^i for intra-block dependency modeling. This causal prefix naturally handles variable-length prefixes, avoiding the alignment issues of Block Diffusion’s grid causal mask where the prefix length must be a multiple of the block size. During inference, SDLM employs *Longest Prefix Decoding* (LPD) to realize the dynamic length of NSP:

$$\hat{X}^i = \text{Decode} \left(Z^i, \gamma_\tau(Z^i) \right), \quad (3)$$

where Z^i are the D logits in the current block and $\gamma_\tau(Z^i) \in [1, D]$ is an adaptive accepted length. The model commits only a high-confidence prefix and discards the rest; details in Section 3.4.

3.3 Training

As noted in Section 3.2, setting $D = 1$ reduces SDLM to the autoregressive paradigm, allowing us to reuse pretrained ALM weights and cut training costs. To efficiently train the model to predict multiple tokens simultaneously, we design a specialized data packing and attention masking strategy.

Appending Noise Blocks for Parallel Training. Instead of training on a single prediction window per sequence, we construct the training sequence by appending multiple independent noise blocks to the end of the original sequence. Given a standard instruction-tuning sequence $S = [X; Y]$ (where X is the prompt and Y is the response), we sample multiple starting positions $\{p_1, p_2, \dots, p_K\}$ within the response Y . For each position p_k , we construct a noise block Y_T^k of length D (consisting entirely of mask tokens). The final training sequence is

formed by concatenation:

$$S_{\text{train}} = \text{concat}(S, Y_T^1, Y_T^2, \dots, Y_T^K). \quad (4)$$

Crucially, we manipulate the *position IDs* of each noise block Y_T^k to align with its corresponding starting position p_k in the original sequence S . This allows multiple prediction windows to be trained in parallel within a single forward pass without interfering with each other.

Structured Attention Mechanism. To support this, we design an asymmetric attention mask (Figure 3(a)): causal for the prefix, bidirectional within each noise block, and masked across blocks. The formal definition is in Appendix B. This sparse pattern is compatible with Magi Attention (Zewei and Yunpeng, 2025) and data packing; despite not using Flash Attention (Dao et al., 2022; Dao, 2024), training cost remains comparable to standard SFT (Appendix B.2).

3.4 Inference

At each decoding step, the model appends one copied anchor token and $D-1$ mask tokens to the current prefix, predicts a block of D candidates in parallel, and commits only the accepted prefix (Equation 3). Because the prefix remains causal, accepted tokens are directly merged into the KV cache. Pseudocode is in Appendix B.3.

Confidence-Based Longest Prefix Decoding.

For the standard LPD variant, we determine the accepted length with a confidence threshold:

$$\gamma_\tau(Z^i) = \max \left(\mathcal{A}_\tau(Z^i) \cup \{1\} \right), \quad (5)$$

$$\mathcal{A}_\tau(Z^i) = \left\{ j \in [0, D] \mid \prod_{k=1}^j p(z_i^k) \geq \tau \right\}.$$

where $Z^i = [z_i^1, \dots, z_i^D]$ denotes the D logits in the current decoding block, $z_i^k \in \mathbb{R}^{|V|}$ is the logit vector at position k , $p(z_i^k)$ is its scalar confidence score, and τ is a predefined threshold. We instantiate $p(z_i^k) = \text{softmax}(z_i^k)_v$, i.e., the probability assigned to the greedy token v , which is simple, efficient, and already available from the decoding step with no extra computation (alternative confidence functions such as normalized entropy yield similar results in our experiments). The model then keeps the longest high-confidence prefix and discards the remaining suffix.

¹https://huggingface.co/AngelSlim/Qwen3-4B_eagle3

Table 1: **Performance and inference efficiency of instruct models.** *Speedup*: tokens per forward pass; *TPS*: end-to-end wall-clock tokens per second (subscript = speedup over baseline). All metrics are averaged within each benchmark group. For ALMs, *HF* (`transformers.generate`) and *vLLM* denote inference backends. For DLMs and SDLM, *Plain/D2F* (Wang et al., 2026) and *LPD/SSD* denote decoding methods, respectively. All reported speedups are relative to the matched *HF* baseline; vLLM numbers are shown for reference. *Scores are from prior D2F work; full setup and data sources are in the Appendix D and E.

Model	Backend/ Decode	Code						Math			
		HumanEval	MBPP	HumanEval+	MBPP+	Speedup	TPS	GSM8K	MATH	Speedup	TPS
Autoregressive Language Models (ALMs)											
Qwen2.5-3B	HF	73.8	65.0	64.0	37.8	1.0	49.7	86.7	66.2	1.0	49.7
	vLLM	76.2	68.5	68.9	39.1	1.0	279.2	86.9	65.6	1.0	280.5
Qwen2.5-7B	HF	82.9	75.5	73.2	47.6	1.0	63.7	91.4	75.4	1.0	63.9
	vLLM	84.2	75.5	75.0	40.7	1.0	165.2	92.4	76.2	1.0	165.9
Qwen2.5-32B	HF	89.0	84.0	81.7	59.3	1.0	30.1	95.2	83.8	1.0	30.4
	vLLM	87.8	85.6	81.1	58.2	1.0	41.6	95.8	84.0	1.0	41.7
Qwen3-4B	HF	78.7	77.8	69.5	49.5	1.0	42.7	92.4	80.6	1.0	42.6
Multi-Token Prediction (MTP)											
Eagle3-Qwen3-4B ¹	–	78.0	75.5	71.3	47.1	1.9	82.0 _{1.92×}	91.9	69.4	2.0	84.2 _{1.98×}
Diffusion Language Models (DLMs)											
LLaDA-8B	Plain	36.0*	39.0*	40.2	25.9	0.6	17.8	77.4*	23.7*	0.8	19.4
	D2F	40.2*	38.0*	30.5	27.8	3.0	84.7	77.3*	29.1*	3.5	105.1
Dream-7B	Plain	54.3*	56.2*	54.3	46.8	0.1	2.6	75.0*	35.8*	0.7	20.3
	D2F	54.3*	55.2*	44.5	41.8	2.9	111.2	77.6*	35.4*	2.8	86.2
Sequential Diffusion Language Models (SDLM, ours)											
SDLM-3B-D8	LPD	65.2	62.6	56.7	38.6	2.6	127.0 _{2.55×}	83.3	56.8	3.4	169.5 _{3.41×}
	SSD	64.0	65.4	59.8	40.2	4.3	215.3 _{4.33×}	84.2	59.8	5.7	281.1 _{5.65×}
SDLM-7B-D8	LPD	76.8	72.8	70.7	48.4	2.5	159.2 _{2.50×}	81.1	68.6	2.8	180.8 _{2.83×}
	SSD	78.0	74.3	73.2	47.4	3.5	220.6 _{3.46×}	83.5	68.8	4.1	260.0 _{4.07×}
SDLM-32B-D4	LPD	79.9	80.9	73.8	57.7	2.0	60.6 _{2.01×}	92.3	73.0	2.4	73.6 _{2.42×}
	SSD	81.1	80.5	74.4	58.5	2.1	63.3 _{2.10×}	92.6	74.2	2.5	75.4 _{2.48×}

Self-Speculative Decoding. We further design a cache-enabled *Self-Speculative Decoding* (SSD) procedure. By carefully designing the attention mask, each single forward pass simultaneously verifies the previous draft and generates the next one in a pipelined fashion: the accepted prefix from draft $\hat{Y}^{i-1}$ is verified while a new draft $\hat{Y}^i = [\hat{y}_i^1, \dots, \hat{y}_i^D]$ is produced. SSD accepts the longest prefix where draft and verification agree: $\gamma_{\text{ssd}}(Z^i) = \max(\{j \in [0, D] \mid \hat{y}_i^k = \tilde{y}_i^k, \forall k \leq j\} \cup \{1\})$. Since verification and drafting are fused into one forward pass, SSD incurs no additional forward-pass overhead compared to LPD, while replacing the heuristic confidence threshold with explicit self-consistency checks. Moreover, because the bidirectional window generalizes beyond the training block size, SSD can expand the inference block to $D_g > D$ (e.g., $D_g=2D$) for further acceleration at no extra training cost. The details are provided in Appendix B.3.

4 Experiments

4.1 Setting

All models are fine-tuned on the same 3.5M-sample instruction corpus and evaluated with OpenCompass. We compare SDLM against same-scale autoregressive, diffusion, and MTP-style baselines; full training and evaluation details are in Appendix C and Appendix D.

4.2 Main Results

Long-Form Code and Math Benchmarks. Table 1 reports accuracy and efficiency on code (HumanEval/+, MBPP/+) and math (GSM8K, MATH) benchmarks. With LPD, SDLM-7B ($D=8$) achieves 159.2 TPS on code and 180.8 TPS on math (2.50 $\times$ and 2.83 $\times$ the Qwen2.5-7B HF baseline) while retaining strong accuracy (e.g., 76.8 on HumanEval, 81.1 on GSM8K). We note that the accuracy gap to the AR baseline is somewhat larger on code than on math, which we attribute in part to the low proportion of coding data in our SFT mixture (Appendix C). Scaling to 32B ($D=4$), SDLM retains over 95% of the Qwen2.5-32B scores at

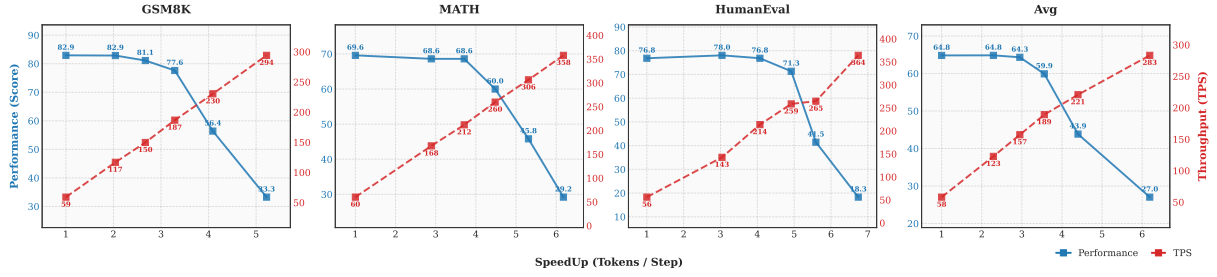


Figure 4: **Quality-speed trade-off under varying τ .** x-axis: speedup; left y-axis (blue): score; right y-axis (red): TPS. Avg = macro-average over 8 benchmarks. Per-benchmark curves in Figure 9.

2.0–2.4 $\times$ speedup. Compared with existing DLMs, SDLM-7B achieves higher accuracy than LLaDA-8B and Dream-7B (e.g., 76.8 vs. 36.0/54.3 on HumanEval). Although D2F decoding greatly improves DLM throughput over plain decoding (e.g., Dream-7B: 111.2 vs. 2.6 TPS), SDLM-7B LPD still surpasses D2F-accelerated DLMs in both accuracy and speed.

Further Acceleration with SSD. As described in Section 3.4, SSD expands the inference-time generation block size to $D_g=2D$ at no extra training cost. SSD consistently boosts throughput with small accuracy changes: SDLM-7B SSD reaches 220.6 TPS on code (3.46 $\times$) and 260.0 TPS on math (4.07 $\times$); using a single-GPU HF backend, SDLM-3B SSD achieves 281.1 TPS on math, even surpassing the vLLM-served Qwen2.5-3B (279.2 TPS).

Comparison with Speculative and MTP-style Baselines. Compared with Eagle3-Qwen3-4B (Li et al., 2025), a representative MTP-style method ($\sim 1.9\times$ speedup, 82–84 TPS), SDLM-3B SSD achieves a higher relative speedup on code (2.6 $\times$) and math (3.3 $\times$). Since Eagle3 uses a different backbone and inference backend, absolute TPS is not directly comparable; we therefore focus on relative speedup here. This advantage comes from SDLM’s bidirectional local window, which captures intra-block dependencies beyond independent MTP heads.

Other Benchmarks. Table 2 shows that SDLM largely preserves general abilities across MMLU, commonsense reasoning, GPQA, and IFEval, closely tracking SFT baselines while outperforming LLaDA-8B and Dream-7B at the 7B scale on MMLU and IFEval. Detailed per-benchmark results are provided in Appendix E.

Table 2: **Performance of instruct models across general benchmarks.**

Model	MMLU	Winogrande	Hellaswag	ARC-C	ARC-E	GPQA	IFEval
<i>Autoregressive Language Models (ALMs)</i>							
Qwen2.5-3B-SFT	67.6	60.8	75.3	83.1	91.4	27.8	61.5
Qwen2.5-7B-SFT	73.8	65.0	85.2	90.2	96.5	33.8	69.1
Qwen2.5-32B-SFT	83.7	78.0	92.4	94.2	99.1	33.8	76.7
<i>Diffusion Language Models (DLMs)</i>							
LLaDA-8B	65.5	–	74.6	88.5	–	31.8	59.9
Dream-7B	67.0	–	–	–	–	33.0	62.5
<i>Sequential Diffusion Language Models (SDLM, ours)</i>							
SDLM-3B	64.1	55.6	72.3	79.7	90.8	29.8	58.6
SDLM-7B	72.1	62.4	81.1	89.2	95.9	32.3	65.1
SDLM-32B	82.8	79.2	92.0	94.9	98.9	36.9	78.0

Training Dynamics: Performance & SpeedUp Rise Together

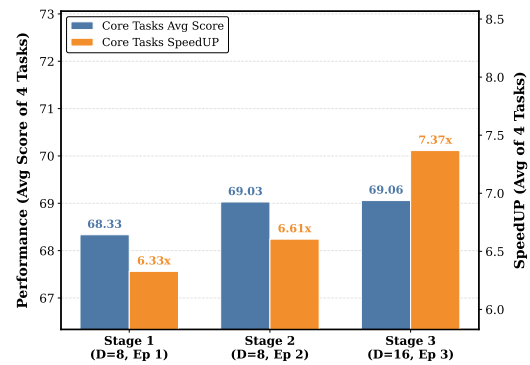
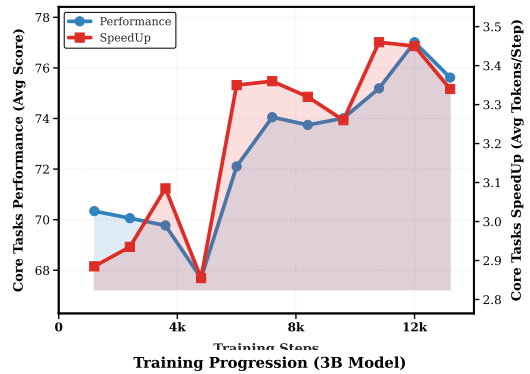


Figure 5: **Training dynamics.** Top: 7B model over 13k steps. Bottom: 3B model across stages. Score and speedup improve throughout training.

Table 3: **Comparison with BD3LM.** 3B models trained for one epoch on the same data. Cell: score / (TPS). BD3LM uses random masking + bidirectional prefix; SDLM uses full masking + causal prefix with LPD ($\tau=0.96$).

Model	D	Mask	Prefix	GSM8K	MATH	HumanEval+	MBPP+	IFEval	Avg	Speedup	TPS
SFT (HF)	–	–	–	86.0 (50.9)	60.4 (50.9)	61.0 (51.0)	42.6 (50.9)	61.5 (51.4)	62.3 (51.0)	1.0	51.0
SFT (vLLM)	–	–	–	86.2 (280.6)	61.0 (280.9)	61.6 (278.8)	42.6 (269.6)	63.2 (280.2)	62.9 (278.0)	1.0	278.0 _{5.45} ×
BD3LM	8	Random	Bidirectional	75.7 (49.5)	52.8 (49.6)	39.0 (49.9)	33.6 (49.0)	44.7 (50.0)	49.2 (49.6)	1.0	49.60 ₉₇ ×
SDLM	4	Full	Causal	84.9 (109.6)	60.0 (111.6)	59.1 (90.3)	40.7 (92.7)	56.4 (81.9)	60.2 (97.2)	2.0	97.2 _{1.91} ×
SDLM	8	Full	Causal	83.6 (130.4)	58.2 (130.2)	58.5 (103.6)	39.7 (117.0)	57.3 (100.7)	59.5 (116.4)	2.5	116.4 _{2.28} ×
SDLM	16	Full	Causal	83.2 (131.9)	57.0 (135.3)	59.1 (132.1)	41.0 (111.2)	58.8 (85.4)	59.8 (119.2)	2.6	119.2 _{2.34} ×

User:	What fraction of 2 feet is 3 inches? Express your answer as a common fraction.
SDLM:	To find the fraction of 2 feet that is 3 inches, we need to convert the feet to inches and then divide the by the total number. \n 1 foot = 12 inches \n 2 feet = 2 * 12 = 24 inches \n Now, we can the 3 inches by the fraction of 24 inches: \n $\frac{3}{24} = \frac{1}{8}$ \n So, the answer is 1/8. \n $\boxed{\frac{1}{8}}$ \n 38 step with 113 tokens
User:	Write a Fibonacci function in Python.
SDLM:	def fibonacci(n): \n if n == 0: \n return 0 \n elif n == 1: \n return 1 \n else: \n return fibonacci(n-1) + fibonacci(n-2) \n 12 step with 44 tokens

Figure 6: **Sampling visualization.** Each block represents one forward pass; darker blue = later step, showing SDLM’s left-to-right block decoding.

4.3 Trade-off Between Speedup and Performance

Existing DLMs (Nie et al., 2026; Ye et al., 2025) support parallel token generation but face a quality-speed trade-off: accepting more tokens per step often degrades quality, while fixed-length noise sequences limit flexibility and practical efficiency gains over ALMs. SDLM mitigates this trade-off: even when accepting only one token per step, SDLM-7B ($D=8$) reaches 58.0 TPS, merely $\sim 7\%$ below the AR baseline (63.8 TPS) despite an $8\times$ longer query window. Moreover, TPS scales nearly linearly with speedup, suggesting that the additional KV-cache recomputation from longer accepted prefixes incurs limited overhead.

Figure 4 shows the trade-off under varying confidence threshold τ . As τ decreases, SDLM generates more tokens per step, achieving up to $3\times$ speedup with limited score degradation; noticeable drops only appear at aggressive thresholds. Math benchmarks such as GSM8K and MATH exhibit a particularly favorable trade-off region, while code and instruction-following tasks are slightly more sensitive but still maintain strong performance at moderate speedup. Per-benchmark trade-off curves

are provided in Appendix E.

4.4 Ablation Studies

Comparison with Block Diffusion. Table 3 compares SDLM with BD3LM (Arriola et al., 2025) and the SFT baseline under the same 3B, one-epoch setting. BD3LM uses its quality-oriented configuration, with the number of intra-block forward steps equal to the block size, yielding about one accepted token per step. Even so, SDLM outperforms BD3LM in this setting (59.5 vs. 49.2 Avg), benefiting from full masking and the causal-prefix prior used by LPD. Compared with SFT (62.3 Avg), SDLM ($D=4$) stays within 2.1 points while reaching $1.9\times$ speedup; $D=8$ and $D=16$ trade small accuracy drops for higher throughput ($2.3\times$).

Longer Training Steps and Stages. We evaluate intermediate checkpoints of the 7B model and the progressive training stages of the 3B model. As shown in Figure 5, both score and speedup improve with training: the 7B model improves steadily over 13k steps, and the 3B model gains additional speedup when moving from $D=8$ to $D=16$. This confirms that longer training produces more confident block-level predictions, which di-

rectly translates to higher acceleration.

Case Study. Figure 6 shows SDLM’s flexible decoding: it emits longer spans in structured regions and shorter ones under uncertainty.

5 Conclusion

We present *Next Sequence Prediction* (NSP), a unified framework that subsumes next token and next block prediction and build on it to develop *Sequential Diffusion Language Models* (SDLMs). SDLMs retrofits pretrained ALMs with minimal cost, largely preserving SFT-level accuracy while achieving up to $3.4\times$ (LPD) and $5.7\times$ (SSD) end-to-end wall-clock TPS gains across model scales (3B, 7B, 32B). We hope this work encourages further exploration of bridging autoregressive and diffusion paradigms.

Limitations

Our training recipe is fully open-source and mainly uses instruction-tuning, code, and math data. This data scale and composition may limit the final model quality, especially compared with larger or proprietary training corpora. On the systems side, SDLM training relies on Magi Attention (Zewei and Yunpeng, 2025) for sparse masks, and its inference attention kernels are not yet as optimized as vLLM (Kwon et al., 2023), leaving room for further speedup. Because SDLM improves inference efficiency, it may also lower the cost of generating undesirable or harmful text when applied to unsafe base models; deployment should therefore follow the safety practices of the underlying model. Our experiments use publicly available instruction-tuning corpora and benchmarks, and we rely on their documented collection and filtering procedures without performing an additional PII or offensive-content audit. Released artifacts are intended for research use and should respect the licenses and terms of the underlying models, datasets, and software.

Acknowledgments

This work is funded by the National Natural Science Foundation of China (Grant No. 62372223 and U24A20330), and supported by the Fundamental Research Funds for the Central Universities under Grant 2232026D60, and supported by the Youth PhD Student Research Project under the National Natural Science Foundation (No. 623B2050).

References

- Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, and 3 others. 2025. *Smollm2: When smol goes big – data-centric training of a small language model*. *Preprint*, arXiv:2502.02737.
- Anthropic. 2023. *Introducing Claude*.
- Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pages 50726–50753.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.
- Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. 2025. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

- OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. <https://github.com/open-compass/opencompass>.
- Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Google DeepMind. 2025. **Gemini 2.5**.
- Yuyang Ding, Xinyu Shi, Xiaobo Liang, Juntao Li, Qiaoming Zhu, and Min Zhang. 2024. Unleashing reasoning capability of llms via scalable question synthesis from scratch. *arXiv preprint arXiv:2410.18693*.
- Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*.
- Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, and 1 others. 2025. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, volume 2025, pages 5046–5073.
- Google DeepMind. 2025. <https://blog.google/technology/google-deepmind/gemini-diffusion/>. <https://blog.google/technology/google-deepmind/gemini-diffusion/>. Accessed: 2024-07-24.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.
- Siming Huang, Tianhao Cheng, Jason Klein Liu, Weidi Xu, Jiaran Hao, Liuyihan Song, Yang Xu, Jian Yang, Jiaheng Liu, Chenchen Zhang, and 1 others. 2025. Opencoder: The open cookbook for top-tier code large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33167–33193.
- Daniel Israel, Guy Van den Broeck, and Aditya Grover. 2026. Accelerating diffusion llms via adaptive parallel decoding. *Advances in neural information processing systems*, 38:52870–52888.
- Ivan Kapelyukh, Vitalis Vosylius, and Edward Johns. 2023. Dall-e-bot: Introducing web-scale diffusion models to robotics. *IEEE Robotics and Automation Letters*, 8(7):3956–3963.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.
- Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2023. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv preprint arXiv:2310.09263*.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Annual Conference on Neural Information Processing Systems*.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.
- Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572.
- Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. 2025. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*.
- Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2026. dkv-cache: The cache for diffusion

- language models. *Advances in Neural Information Processing Systems*, 38:149009–149033.
- Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025. Scaling up masked diffusion models on text. In *International Conference on Learning Representations*, volume 2025, pages 82974–82997.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2026. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646.
- OpenAI. 2022. [Introducing ChatGPT](#).
- OpenAI. 2024. [ao GPT-4o](#).
- Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2025. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *International Conference on Learning Representations*, volume 2025, pages 64972–65009.
- Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. WinoGrande: An adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.
- Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. 2025. Your llm knows the future: Uncovering its multi-token prediction potential. *arXiv preprint arXiv:2507.11851*.
- Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, and 1 others. 2025. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*.
- Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- David Wadden, Kejian Shi, Jacob Morrison, Alan Li, Aakanksha Naik, Shruti Singh, Nitzan Barzilay, Kyle Lo, Tom Hope, Luca Soldaini, and 1 others. 2025. Sciriff: A resource to enhance language model instruction-following over scientific literature. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6083–6120.
- Weiyun Wang, Zhe Chen, Wenhai Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Jinguo Zhu, Xizhou Zhu, Lewei Lu, Yu Qiao, and 1 others. 2024. Enhancing the reasoning ability of multimodal large language models via mixed preference optimization. *arXiv preprint arXiv:2411.10442*.
- Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, Kai Yu, and Zhijie Deng. 2026. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. In *International Conference on Learning Representations*, volume 2026, pages 2282–2299.
- Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2026. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *International Conference on Learning Representations*, volume 2026, pages 57027–57051.
- xAI. 2025. [Grok 3 beta — the age of reasoning agents](#).
- Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.
- Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800.

Tao Zewei and Huang Yunpeng. 2025. Magiattention: A distributed attention towards linear scalability for ultra-long context, heterogeneous mask training. <https://github.com/SandAI-org/MagiAttention/>.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. 2023. A reparameterized discrete diffusion model for text generation. *arXiv preprint arXiv:2302.05737*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Block Diffusion Objective

Block Diffusion (Arriola et al., 2025) constrains the diffusion operation to proceed sequentially in blocks of length D . At each time step t , the model receives a noisy block $X_t^i = x_t^{iD:(i+1)D}$ and predicts all masked tokens within the block:

$$\mathcal{L}_{\text{BD}}(X; \theta) = - \sum_{i=1}^{L/D} \mathbb{E}_{t \sim [0,1]} \mathbb{E}_q \frac{\alpha'_t}{1 - \alpha_t} \cdot \log P_\theta(X^i | X^{<i}, X_t^i), \quad (6)$$

where X denotes the ground truth, q is the forward masking process, $\alpha_t \in [0, 1]$ is the probability of keeping a token at time t , and α'_t is the instantaneous rate of change of α_t in continuous time.

B Implementation Details

B.1 Training

To efficiently implement the appended sequence structure and the asymmetric attention mask described in the main text, we adopt a streamlined approach based on position ID manipulation.

Formally, for a noise block Y_T^k corresponding to prefix length p_k , the attention mask A for query index u and key index v is:

$$A_{uv} = \begin{cases} 1, & v \leq u < p_k, \\ 1, & u \in Y_T^k, v < p_k, \\ 1, & u \in Y_T^k, v \in Y_T^k, \\ 0, & \text{otherwise.} \end{cases}$$

The three active cases correspond to the global causal prefix, visible prefix projection, and local bidirectional block, respectively.

The detailed procedures for data packing and mask construction are presented in Algorithm 1 and Algorithm 2, respectively.

B.2 Training Efficiency

We measure training efficiency on $8 \times \text{H100 80GB}$ GPUs with a 3B model, a maximum sequence length of 16k, and stream packing. Standard AR SFT uses a regular causal mask and thus can directly leverage FlashAttention. In contrast, SDLM concatenates additional target blocks to each instance and employs a custom sparse attention mask, which prevents using FlashAttention. We instead use Magi Attention (Zewei and Yunpeng, 2025), which supports such sparse masks and works seamlessly with packing; the extra cost compared to SFT is dominated by the **backward** pass.

Algorithm 1 Data Packing for Parallel Training

```
1: Input: Base sequence  $S$  of length  $L_S$ , block size  $D$ , number of blocks  $K$ , sampled target positions  $\{p_1, p_2, \dots, p_K\}$ 
2: Output: Concatenated  $x_{\text{in}}$  and  $\text{PE}_{\text{in}}$ 
3:  $x_{\text{in}} \leftarrow S$  // initialize with base sequence
4:  $\text{PE}_{\text{in}} \leftarrow [0, 1, \dots, L_S - 1]$ 
5: for  $k \leftarrow 1$  to  $K$  do
6:    $Y_T^k \leftarrow [\text{m}]^{\times D}$  // create noise block with  $D$  mask tokens
7:    $x_{\text{in}} \leftarrow x_{\text{in}} \| Y_T^k$ 
8:    $P^k \leftarrow [p_k, p_k + 1, \dots, p_k + D - 1]$  // rollback position IDs to align with prefix boundary
9:    $\text{PE}_{\text{in}} \leftarrow \text{PE}_{\text{in}} \| P^k$ 
10: end for
11: return  $x_{\text{in}}, \text{PE}_{\text{in}}$ 
```

Algorithm 2 Vectorized Mask Construction

```
1: Input: Position IDs  $\text{PE}_{\text{in}}$  of length  $L_{\text{total}}$ , block size  $D$ 
2: Output: Boolean attention mask  $A$  of shape  $L_{\text{total}} \times L_{\text{total}}$ 
3:  $L_S \leftarrow \min(\{i \mid \text{PE}_{\text{in}}[i] < \text{PE}_{\text{in}}[i - 1]\})$  // detect base sequence boundary
4:  $u \leftarrow [0, 1, \dots, L_{\text{total}} - 1]^T$  // query indices, shape:  $[L_{\text{total}}, 1]$ 
5:  $v \leftarrow [0, 1, \dots, L_{\text{total}} - 1]$  // key indices, shape:  $[1, L_{\text{total}}]$ 
6:  $u_{\text{blk}} \leftarrow (u - L_S) // D$  // compute block indices for noise blocks
7:  $v_{\text{blk}} \leftarrow (v - L_S) // D$ 
8:  $p_{\text{len}} \leftarrow \text{PE}_{\text{in}}[L_S + u_{\text{blk}} \times D]$  // retrieve prefix length for each query token
9:  $\text{is\_base}_u \leftarrow u < L_S$  // construct the three functional regions
10:  $\text{is\_base}_v \leftarrow v < L_S$ 
11:  $A_{\text{causal}} \leftarrow \text{is\_base}_u \wedge \text{is\_base}_v \wedge (u \geq v)$  // global causal prefix
12:  $A_{\text{prefix}} \leftarrow \neg \text{is\_base}_u \wedge \text{is\_base}_v \wedge (v < p_{\text{len}})$  // visible prefix projection
13:  $A_{\text{mutual}} \leftarrow \neg \text{is\_base}_u \wedge \neg \text{is\_base}_v \wedge (u_{\text{blk}} == v_{\text{blk}})$  // local bidirectional synergy
14:  $A \leftarrow A_{\text{causal}} \vee A_{\text{prefix}} \vee A_{\text{mutual}}$  // combine regions
15: return  $A$ 
```

The detailed training efficiency comparison is summarized in Table 4.

For SDLM-3B ($D=8$), the mask contains more contiguous block structure and is therefore more regular, leading to improved efficiency compared to $D=4$. Although target block concatenation increases sequence length and may require more optimization steps under a “samples seen” accounting, the overall training overhead relative to standard SFT remains modest and practical.

B.3 Inference

Self-Speculative Decoding. A first forward pass produces a draft block $\hat{Y}^i = [\hat{y}_i^1, \dots, \hat{y}_i^D]$. A second pipelined pass simultaneously updates the KV cache, verifies the draft, and generates the next speculative continuation. Let Y^i denote the verified predictions; SSD accepts the longest matching

prefix:

$$\gamma_{\text{ssd}}(Z^i) = \max(\mathcal{A}_{\text{ssd}}(Z^i) \cup \{1\}), \quad (7)$$
$$\mathcal{A}_{\text{ssd}}(Z^i) = \left\{j \in [0, D] \mid \hat{y}_i^k = \tilde{y}_i^k, \forall k \leq j\right\}.$$

This forms a chained draft–verify pipeline with cache reuse.

We provide pseudocode for SDLM sampling, including confidence-based decoding (Algorithm 3) and self-speculative decoding (Algorithm 4). The corresponding input IDs, position IDs, and attention masks are shown in Figure 7. For confidence-based decoding, the relevant mask is the bold upper-left block in Figure 7.

During inference, we use two attention patterns. Within each decoding window, tokens attend to one another using bidirectional attention. For the subsequence accepted from the previous decoding step, we apply causal attention to recompute the KV cache in a manner consistent with training.

Table 4: Training efficiency comparison on $8 \times \text{H100}$ 80GB GPUs (3B model, 16k max sequence length). Time is measured per step and for a total of 5k steps.

Method	Backend	Time per Step (s)			5k Steps (hr)
		Fwd	Bwd	Total	
Qwen2.5-3B SFT	Flash Attention	~ 0.7	~ 1.4	~ 2.1	~ 3.0
SDLM-3B ($D=4$)	Magi Attention	~ 0.9	~ 2.6	~ 3.6	~ 5.0
SDLM-3B ($D=8$)	Magi Attention	~ 0.6	~ 1.9	~ 2.8	~ 3.8

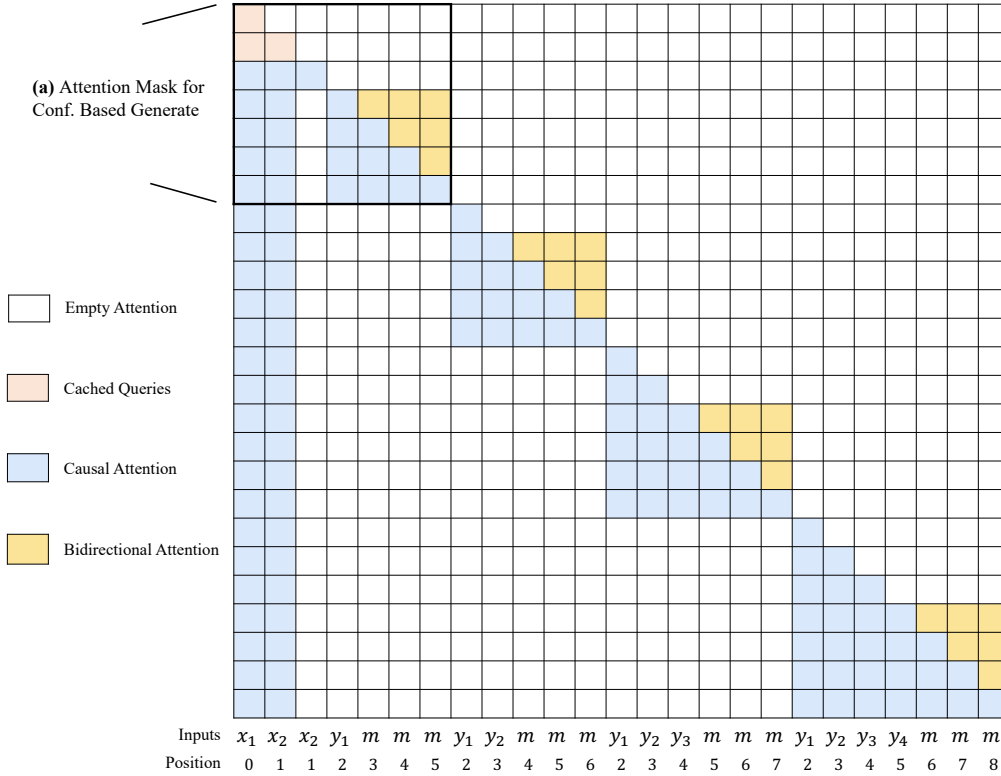


Figure 7: **Structured attention mask for sampling.** (a) The bold upper-left block shows the mask used during confidence-based generation; the full mask shows the mask used in self-speculative decoding.

In particular, the last token of the previous window serves two roles and is therefore duplicated: it participates in causal attention for KV recomputation, during which it is masked from the subsequent decoding window, and it also acts as the first token (anchor) of the next decoding window. The attention mask is illustrated in Figure 7(a).

C Training Details

Table 5 summarizes the training hyperparameters. The training corpus comprises Tulu-3-SFT-Mixture (Lambert et al., 2024), TableGPT (Li et al., 2023), SciRIFF (Wadden et al., 2025), SmolTalk (Allal et al., 2025), OPC-SFT-Stage2 (Huang et al., 2025), and ScaleQuest-Math (Ding et al., 2024), with a combined total

of 3.5 million samples (~ 2.3 billion tokens). The detailed composition is shown in Table 6.

As shown in Table 6, math-related data (ScaleQuest, $\sim 1.0\text{M}$ samples) constitutes a larger portion of the training set than code-related data (OPC-SFT-S2, $\sim 436\text{K}$ samples). This imbalance in data composition likely explains the performance gap observed in our main results: SDLM achieves relatively stronger performance on math benchmarks (GSM8K, MATH), while showing a larger gap to the autoregressive baseline on code benchmarks (HumanEval, MBPP).

D Evaluation Setup

Hardware and Inference Backends. All models are evaluated on $8 \times \text{NVIDIA H100 80GB}$

Algorithm 3 Confidence-Based Decoding

```
1: Input: Model  $\mathcal{M}$ , prompt  $x$ , block size  $D$ 
2: Output: Generated sequence  $y$ 
3: Initialize KV-cache.
4:  $y \leftarrow x$ ;  $L \leftarrow |x|$ 
5: while not EOS do
6:    $x_{\text{in}} \leftarrow [y, y_{-1}, [\text{m}]^{\times(D-1)}]$ 
7:    $\mathbf{p} \leftarrow \text{range}(0, L) \parallel [L-1]$ 
8:    $\text{PE}_{\text{in}} \leftarrow \mathbf{p} \parallel \text{range}(L-1, L+D-2)$ 
9:    $(\text{logits}, \text{KV}_{\text{new}}) \leftarrow \mathcal{M}(x_{\text{in}}, \text{KV}_{\text{cache}}, \text{PE}_{\text{in}})$ 
10:   $\text{KV}_{\text{cache}} \leftarrow \text{truncate } \text{KV}_{\text{new}} \text{ to } 0:L$ 
11:   $\hat{d}_{1:D} \leftarrow \text{last } D\text{-token predictions}$ 
12:   $k \leftarrow \max\{j \in [1, D] \mid \text{confidence}(\hat{d}_{1:j}) \geq \tau\}$ 
13:   $y \leftarrow y \parallel \hat{d}_{1:k}$ ;  $L \leftarrow L + k$ 
14:  if  $\hat{d}_{1:k}$  contains EOS then
15:    break
16:  end if
17: end while
18: return  $y$ 
```

Table 5: Training hyperparameters.

Parameter	3B/7B	32B
Block size D	8	4
Max sequence length	5,632	
Epochs	1	
Batch size (global)	256	464
Training steps	13,699	7,558
Learning rate	5×10^{-6} (constant)	
ZeRO stage	1	3

GPUs with batch size 1. Benchmark accuracy is measured using OpenCompass (Contributors, 2023). For Qwen2.5 models, we use transformers (v4.37.2) as the HF backend and vLLM (v0.14.0) as the optimized engine. For Qwen3-4B and Eagle3-Qwen3-4B, we use transformers (v4.57.1); Eagle3 draft weights are from https://huggingface.co/AngelSlim/Qwen3-4B_eagle3.

Benchmarks. We evaluate across a diverse set of benchmarks:

- *General Tasks:* MMLU (Hendrycks et al., 2020) (5-shot), Winogrande (Sakaguchi et al., 2021) (0-shot), Hellaswag (Zellers et al., 2019) (10-shot), ARC-C/E (Clark et al., 2018) (0-shot), IFEval (Zhou et al., 2023) (0-shot).
- *Math & Science:* GSM8K (Cobbe et al., 2021)

Table 6: Training data composition.

Dataset	# Samples	Domain
ScaleQuest	1,000K	Math
OPC-SFT-S2	436K	Code
SmolTalk	1,100K	General
Tulu-3-SFT	939K	General
SciRIFF	79K	Science
Table-GPT	13K	Table
Total	3,506K	–

(0-shot), MATH-500 (Hendrycks et al., 2021) (0-shot), GPQA (Rein et al., 2024) (0-shot).

- *Coding:* HumanEval (Chen et al., 2021) (0-shot), HumanEval+ (Liu et al., 2023) (0-shot), MBPP (Austin et al., 2021) (3-shot), MBPP+ (Liu et al., 2023) (3-shot).

Speedup. Speedup is defined as the average number of tokens generated per forward pass:

$$\text{Speedup} = \frac{\text{number of generated tokens}}{\text{number of forward steps}}. \quad (8)$$

By definition, autoregressive decoding has a speedup of 1. For masked diffusion models (e.g., Dream, LLaDA), the generation length is fixed and post-EOS tokens are truncated, so speedup can fall below 1.

Algorithm 4 Self-Speculative Decoding

```
1: Input: Draft/Verify model  $\mathcal{M}$ , prompt  $x$ , block size  $D$ 
2: Output: Generated sequence  $y$ 
3: Initialize KV-cache.
4:  $y \leftarrow x$ ;  $L \leftarrow |x|$ 
5:  $\text{pad\_len} \leftarrow D(D-1) + \frac{D(D+1)}{2}$ 
6: Prefill: obtain initial draft block  $y_{1:D}^{\text{draft}}$ 
7:  $x_{\text{in}} \leftarrow [y, y_{-1}, [m]^{\times(D-1)}]$ 
8:  $\mathbf{p} \leftarrow \text{range}(0, L) \parallel [L-1]$ 
9:  $\text{PE}_{\text{in}} \leftarrow \mathbf{p} \parallel \text{range}(L-1, L+D-2)$ 
10:  $(\text{logits}, \text{KV}) \leftarrow \mathcal{M}(x_{\text{in}}, \text{PE}_{\text{in}}, \text{KV})$ 
11:  $\text{KV} \leftarrow \text{truncate to positions } 0:L$  // keep only causal cache
12:  $y^{\text{draft}} \leftarrow \text{last-}D \text{ predictions from logits}$ 
13: Decode: draft & verify loop
14: while not EOS do
15:    $(\text{pad}(y^{\text{draft}}), \text{pad}(\mathbf{PE})) \leftarrow \text{PadInputs}(y^{\text{draft}}, L)$ 
16:    $x_{\text{in}} \leftarrow [y, y_{-1}, \text{pad}(y^{\text{draft}})]$ 
17:    $\mathbf{p} \leftarrow \text{range}(0, L) \parallel [L-1]$ 
18:    $\text{PE}_{\text{in}} \leftarrow \mathbf{p} \parallel \text{pad}(\mathbf{PE})$ 
19:    $(\text{logits}, \text{KV}_{\text{new}}) \leftarrow \mathcal{M}(x_{\text{in}}, \text{PE}_{\text{in}}, \text{KV})$ 
20:    $\text{KV} \leftarrow \text{truncate } \text{KV}_{\text{new}} \text{ to } 0:L$ 
21:    $y^{\text{verify}} \leftarrow \text{last-pad\_len predictions from logits}$ 
22:    $(y^{(0)}, y^{(1)}) \leftarrow \text{reshape}(y^{\text{draft}}, y^{\text{verify}})$ 
23:    $(y^{\text{new-draft}}, k) \leftarrow \text{AcceptCount}(y^{(0)}, y^{(1)})$ 
24:    $y \leftarrow y \parallel y_{1:k}^{\text{draft}}; \quad L \leftarrow L + k$ 
25:   if  $y_{1:k}^{\text{draft}}$  contains EOS then
26:     break
27:   end if
28:    $y^{\text{draft}} \leftarrow y^{\text{new-draft}}$ 
29: end while
30: return  $y$ 
```

TPS. Tokens Per Second (TPS) measures wall-clock throughput for a single request:

$$\text{TPS} = \frac{\text{number of generated tokens}}{\text{wall-clock inference time}}. \quad (9)$$

Both metrics are averaged over all test samples within each benchmark group.

LLaDA and Dream Evaluation. We evaluate LLaDA-8B (Nie et al., 2026) and Dream-7B (Ye et al., 2025) with both their original (plain) decoding and D2F (Wang et al., 2026) on the same $8 \times \text{H100}$ platform. In Table 1, D2F scores marked with * are from Wang et al. (2026); all other results (including Table 8) are independently evaluated by us using OpenCompass. Per-benchmark generation hyperparameters are listed in Table 7.

E Detailed Results

Per-Model Breakdown. Table 8 reports per-benchmark scores (top of each cell) and throughput in tokens per second (green, bottom) for every model and inference backend. Models are grouped by parameter size (3B, 4B, 7B, 32B) and by model family (DLM). Within each group, thin rules separate three categories: (i) official pretrained checkpoints with HF and vLLM backends, (ii) our SFT baselines trained on the same data with the standard autoregressive loss, and (iii) SDLM variants decoded with self-speculative decoding.

Inference Efficiency on Long Sequences. As shown in Figure 8, grouping 7B evaluation samples by generation length reveals a strong positive correlation between length and efficiency: short responses (0–64 tokens) achieve $3.88 \times$ speedup,

Table 7: **DLM generation hyperparameters per benchmark.** Each cell shows *gen length* / *forward steps* for plain decoding and *max out len* / *block size* for D2F. Plain decoding runs exactly *gen length* forward steps regardless of output content; D2F forward steps are adaptive.

Model	Decoding	GSM8K	MATH	GPQA	HumanEval(+)	MBPP(+)	IFEval
LLaDA-8B	Plain	512/512	512/512	64/64	512/512	256/256	512/512
	D2F	512/64	512/32	512/32	512/32	512/32	512/32
Dream-7B	Plain	512/512	512/512	128/128	768/768	1024/1024	1280/1280
	D2F	256/32	256/64	256/32	256/32	256/48	256/32

Table 8: **Detailed Evaluation Results.** Each cell shows the benchmark score on top and throughput (TPS) in green parentheses below. “HF” = Hugging Face transformers; “vLLM” = vLLM engine. “SFT” models are fine-tuned on the same data as SDLM with autoregressive loss. SDLM results use self-speculative decoding (SSD) with $D_g=2D$. Speedup and TPS are averaged over all 8 benchmarks.

Size	Model	GSM8K	MATH	GPQA	HumanEval	HumanEval+	MBPP	MBPP+	IFEval	Speedup (×)	Avg TPS
3B	Qwen2.5 3B (HF)	86.7 (49.7)	66.2 (49.8)	35.9 (49.1)	73.8 (49.0)	64.0 (50.4)	65.0 (49.3)	37.8 (50.1)	57.7 (50.1)	1.0	49.7
	Qwen2.5 3B (vLLM)	86.9 (280.4)	65.6 (280.6)	35.9 (278.7)	76.2 (279.3)	68.9 (280.4)	68.5 (278.8)	39.1 (278.3)	58.6 (279.9)	1.0	279.6
	SFT 3B (HF)	86.0 (50.9)	60.4 (50.9)	27.8 (49.9)	73.2 (50.4)	61.0 (51.0)	68.5 (50.2)	42.6 (50.9)	61.5 (51.4)	1.0	50.7
	SFT 3B (vLLM)	86.2 (280.6)	61.0 (280.9)	26.3 (258.0)	72.6 (274.0)	61.6 (278.8)	69.7 (266.2)	42.6 (269.6)	63.2 (280.2)	1.0	273.5
	SDLM-3B-D8	84.2 (290.3)	59.8 (271.9)	29.8 (181.2)	64.0 (174.8)	59.8 (242.8)	65.4 (186.3)	40.2 (257.2)	58.6 (176.4)	6.0	222.6
4B	Qwen3-4B (HF)	92.4 (42.4)	80.6 (42.8)	42.9 (43.0)	78.7 (42.5)	69.5 (42.9)	77.8 (42.6)	49.5 (42.7)	78.9 (43.0)	1.0	42.7
	Eagle3-Qwen3-4B	91.9 (87.7)	69.4 (80.7)	38.4 (74.4)	78.0 (82.8)	71.3 (83.7)	75.5 (81.0)	47.1 (80.5)	78.4 (62.3)	3.1	79.1
7B	Qwen2.5 7B (HF)	91.4 (63.8)	75.4 (63.9)	36.4 (63.6)	82.9 (63.1)	73.2 (64.0)	75.5 (63.7)	47.6 (64.0)	71.2 (64.5)	1.0	63.8
	Qwen2.5 7B (vLLM)	92.4 (165.8)	76.2 (166.1)	36.4 (165.3)	84.2 (165.1)	75.0 (165.8)	75.5 (164.9)	40.7 (165.2)	72.3 (165.9)	1.0	165.5
	SFT 7B (HF)	89.1 (62.6)	69.4 (62.5)	33.8 (60.7)	76.2 (61.0)	72.6 (62.8)	73.9 (60.4)	52.4 (61.7)	69.1 (63.1)	1.0	61.8
	SFT 7B (vLLM)	89.3 (165.6)	71.0 (166.2)	34.3 (156.6)	77.4 (163.1)	72.0 (164.7)	73.9 (158.6)	51.9 (159.8)	69.7 (165.9)	1.0	162.6
	LLaDA-8B (Plain)	74.8 (8.7)	30.0 (30.2)	30.8 (14.2)	43.3 (30.2)	40.2 (30.7)	42.0 (4.8)	25.9 (5.4)	60.3 (21.8)	0.7	18.2
	LLaDA-8B (D2F)	79.5 (117.5)	33.4 (92.7)	32.8 (49.0)	40.2 (99.4)	30.5 (94.5)	48.2 (65.0)	27.8 (70.0)	46.4 (48.3)	2.8	79.5
	Dream-7B (Plain)	46.2 (11.1)	45.6 (20.3)	30.8 (5.8)	57.9 (4.2)	54.3 (7.2)	67.3 (0.9)	46.8 (1.0)	56.8 (5.0)	0.3	6.9
	Dream-7B (D2F)	81.8 (66.0)	42.8 (106.5)	29.8 (72.2)	56.1 (61.1)	44.5 (68.8)	46.3 (163.7)	41.8 (151.2)	26.8 (56.2)	2.6	93.2
	SDLM-7B-D8	83.5 (227.5)	68.8 (292.5)	32.3 (190.2)	78.0 (282.1)	73.2 (259.5)	74.3 (178.9)	47.4 (161.9)	65.1 (133.8)	5.9	215.8
32B	Qwen2.5 32B (HF)	95.2 (30.4)	83.8 (30.4)	47.0 (30.2)	89.0 (30.3)	81.7 (30.4)	84.0 (29.9)	59.3 (29.9)	79.7 (30.5)	1.0	30.2
	Qwen2.5 32B (vLLM)	95.8 (41.7)	84.0 (41.8)	46.5 (41.7)	87.8 (41.5)	81.1 (41.8)	85.6 (41.5)	58.2 (41.6)	79.5 (41.7)	1.0	41.7
	SFT 32B (HF)	93.0 (28.8)	75.4 (28.4)	33.8 (27.5)	82.9 (28.2)	76.8 (28.6)	82.1 (27.2)	59.3 (27.3)	76.7 (28.6)	1.0	28.1
	SFT 32B (vLLM)	93.6 (39.1)	73.2 (39.2)	33.8 (34.2)	84.2 (37.9)	76.8 (38.6)	82.1 (37.6)	59.3 (37.7)	76.3 (39.2)	1.0	37.9
	SDLM-32B-D4	92.6 (76.4)	74.2 (74.4)	36.9 (57.3)	81.1 (69.9)	74.4 (73.2)	80.5 (54.7)	58.5 (55.5)	78.0 (48.4)	3.6	63.7

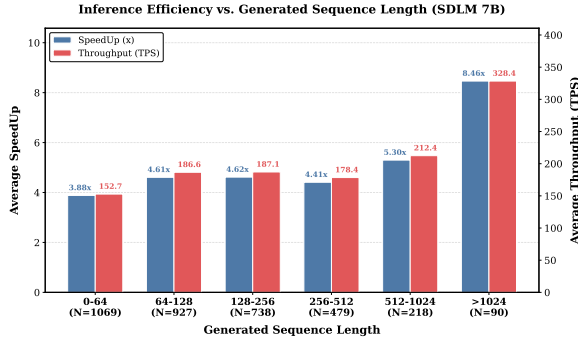


Figure 8: **Efficiency vs. sequence length (SDLM-7B).** Speedup and TPS grow with longer outputs.

while sequences exceeding 1024 tokens reach $8.46\times$ speedup and 328.4 TPS.

Per-Benchmark Trade-off Analysis. Figure 9 presents the complete quality–speed trade-off curves for all 8 individual benchmarks under logit-based confidence decoding. Each curve is obtained by sweeping the confidence threshold τ from strict ($\tau=0.96$) to aggressive ($\tau=0.34$); the left-most point on each curve corresponds to the greedy baseline that accepts exactly one token per step ($\tau=1$). Across all benchmarks, we observe a consistent pattern: relaxing τ steadily increases both speedup and TPS, while the performance score degrades gracefully. Notably, math and code benchmarks (GSM8K, MATH, HumanEval, MBPP) exhibit a favorable trade-off region where a $2\text{--}3\times$ speedup is achievable with only marginal score loss, whereas instruction-following (IFEval) and knowledge-intensive tasks (GPQA) are more sensitive to aggressive thresholds.

F Additional Ablation Studies

We conduct ablation studies on the model design of SDLM, examining the effects of shift prediction and bidirectional attention. All variants are retrained under the same settings to evaluate their impact on performance and efficiency.

No Shift Prediction. To verify the effectiveness of shift prediction, we adopt a variant similar to LLaDA that directly predicts the original tokens at the masked positions. As shown in Figure 10, under the same training cost, this variant leads to a noticeable decline in model performance, with HumanEval+ scores dropping by approximately 14 points. Log analysis further reveals that this variant produces more repeated outputs. This indicates that the shift prediction strategy exploits the strong

ability of ALMs to predict the first token, providing a stable starting point for diffusion decoding.

Causal Attention. As shown in Figure 10, we replace the *bidirectional* attention inside each block with a *causal* (unidirectional) mask. With a block size $D = 4$, the two variants obtain almost identical scores on some benchmarks and exhibit comparable training difficulty. However, the average number of tokens generated per step drops from 1.88 to 1.82, indicating that bidirectional attention enlarges the local receptive field during decoding and improves parallel generation efficiency.

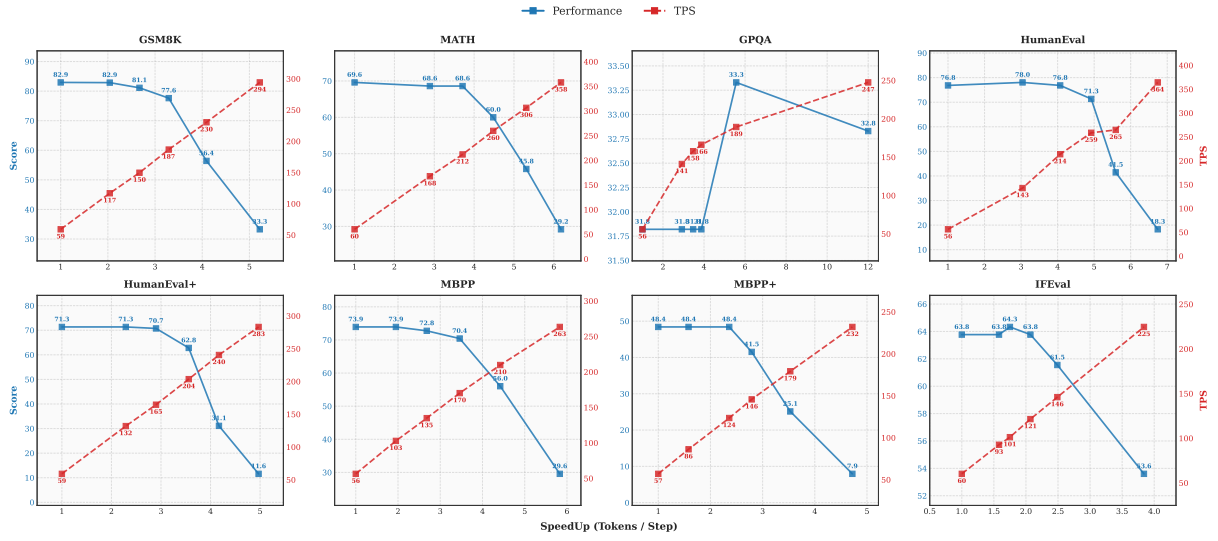


Figure 9: **Per-benchmark quality-speed trade-off under varying confidence threshold τ .** Each panel shows one of the 8 evaluation benchmarks. x -axis: speedup (tokens per forward pass); left y -axis (blue): score; right y -axis (red): TPS.

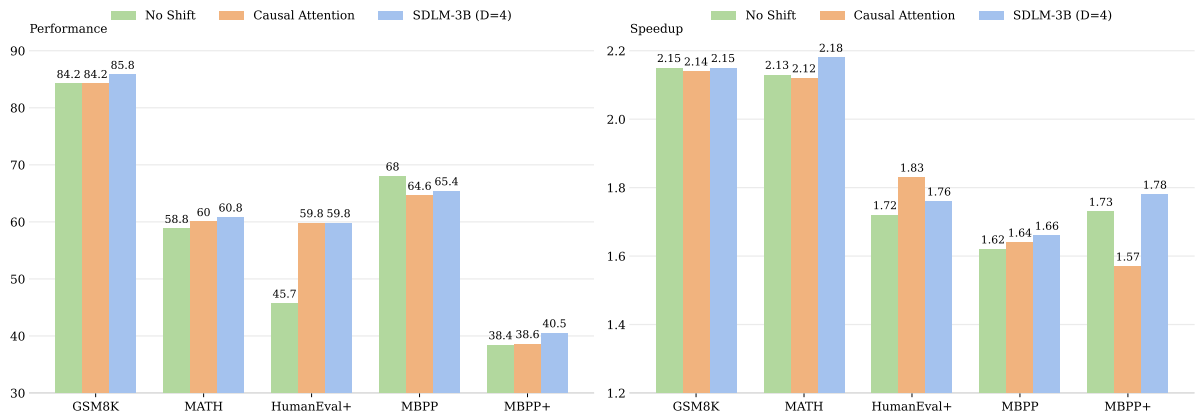


Figure 10: **Ablation on attention mask type and prediction shift strategy.** (1) No shift: predicting x_t instead of x_{t+1} ; (2) Causal-only: using a causal mask instead of bidirectional attention within each block. The left panel shows benchmark performance, while the right panel shows the acceleration ratio.