

000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053

SEQUENTIAL DIFFUSION LANGUAGE MODELS

Anonymous authors

Paper under double-blind review

ABSTRACT

Diffusion language models (DLMs) have strong theoretical efficiency but are limited by fixed-length decoding and incompatibility with key-value (KV) caches. Block diffusion mitigates these issues, yet still enforces a fixed block size and requires expensive training. We introduce Next Sequence Prediction (NSP), which unifies next-token and next-block prediction, enabling the model to adaptively determine the generation length at each step. When the length is fixed to 1, NSP reduces to standard next-token prediction. Building on NSP, we propose Sequential Diffusion Language Model (SDLM), which can retrofit pre-trained autoregressive language models (ALMs) at minimal cost. Specifically, SDLM performs diffusion inference within fixed-size mask blocks, but dynamically decodes consecutive subsequences based on model confidence, thereby preserving KV-cache compatibility and improving robustness to varying uncertainty and semantics across the sequence. Experiments show that SDLM matches or surpasses strong autoregressive baselines using only 3.5M training samples, while achieving 2.1 \times higher throughput than Qwen-2.5. Notably, the SDLM-32B model delivers even more pronounced efficiency gains, demonstrating the strong scalability potential of our modeling paradigm. Code and models will be released.

1 INTRODUCTION

In recent years, diffusion models have made significant progress in computer vision, dominating various fields such as image generation (Ho et al., 2020; Rombach et al., 2022) and robot control (Chi et al., 2023; Kapelyukh et al., 2023). This successful paradigm has recently emerged as a potential 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, showing superior theoretical efficiency. However, DLMs are also criticized for its fixed decoding length and inability to use KV cache (Radford et al., 2019).

To address these limitations, it is a natural thought to combine the benefit of DLM and ALM, similar to existing efforts like Block Diffusion (Arriola et al., 2025). Specifically, Block Diffusion reformulate the next token prediction of ALM as the next block prediction, where tokens in each block are decoded in a diffusion manner. In this case, Block Diffusion not only preserve the autoregressive property for flexible and robust prediction, [while also exploiting diffusion-style parallel prediction for better efficiency](#).

Despite the effectiveness, Block Diffusion models still remain two practical limitations. Firstly, the block size is fixed in block diffusion models, which means that the model should predict a constant number of tokens in each step. However, the distribution of certainty and semantics varies across the entire sequence, typically requiring adjusting the suitable block size in predicting different subsequences. As shown in Figure 1(b), a fixed block size easily fails in token prediction that requires previous context information. Secondly, both the DLM and the block diffusion model require training from scratch and cannot be easily developed from a pre-trained ALM. This not only leads to significant training costs but also creates obstacles for developing larger models.

In this paper, we introduce *Next Sequence Prediction* (NSP), a general form of next token prediction and next block prediction. Specifically, NSP defines an autoregressive probability distribution for sequences of discrete random variables. As shown in Figure 1(c), NSP predicts future sequences of variable length, where a sequence can be either one token or a block of tokens. At each step, NSP decodes the tokens in the sequence in a diffusion manner. Therefore, NSP can dynamically adjust

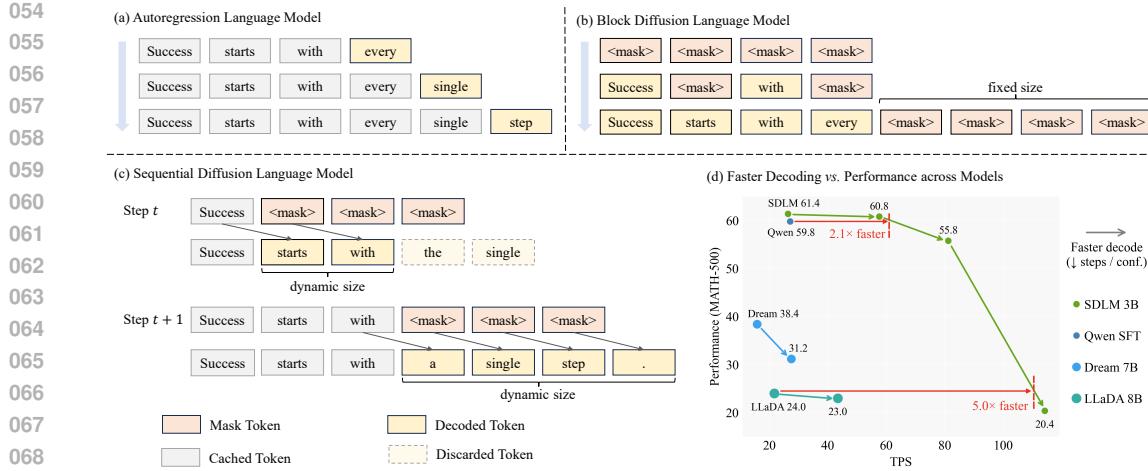


Figure 1: **Comparison of decoding paradigms.** (a) ALMs: decode one token at a time. (b) DLMs (e.g. Block Diffusion): decode all tokens in a fixed block before moving to the next. (c) SDLM (Ours): dynamically predicts a contiguous subsequence within a fixed block. (d) Performance vs. Speed: MATH-500 results showing trade-off between speed (TPS) and accuracy.

its decoding sequence size according to the difficulty and semantics of future sequences. When the length of the prediction sequence is always 1, NSP degenerates to next-token prediction. This property allows NSP to seamlessly adapt to existing pre-trained ALMs with cheap costs.

Based on the principle of NSP, we propose *Sequential Diffusion Language Models* (SDLMs) with innovative training and inference strategies. As shown in Figure 2, SDLMs are developed based on a pre-trained ALMs, employing a novel 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. Unlike our concurrent work, [Samragh et al. \(2025\)](#) employs gated LoRA with next-token and multi-token prediction losses, whereas we use NTP cross-entropy loss for full supervised fine-tuning. For sampling, we apply bidirectional attention with confidence-based decoding, without extra sampling heads.

To validate our approach, we construct different scales of LLMs and conduct extensive experiments on 13 benchmarks across general, math, knowledge and coding tasks. Experiments show that our SDLMs achieve on-par performance with existing ALMs with much faster speed, *e.g.*, $2.1 \times$ faster than Qwen-2.5-3B ([Team, 2024](#)). Compared to existing DLMs, our SDLMs demonstrate comprehensive advantages in performance, efficiency, and training costs. For example, SDLM-3B significantly outperforms DLMs like Dream-7B ([Ye et al., 2025](#)) and LLaDA-8B ([Nie et al., 2025b](#)) across multiple benchmarks, while requiring far less training compute and yielding substantially higher inference speed. More importantly, the scalability of SDLMs is validated on the larger models, *i.e.*, Qwen-2.5-32B, requiring only 3.5M training [samples](#). In summary, our contributions are three-folds:

- 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, **while providing a practical way to relax the fixed block-size constraint in prior block diffusion models**.
- Based on NSP, we deploy *Sequential Diffusion Language Models* (SDLMs) through a novel 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.

108 • Extensive experiments not only demonstrate the effectiveness and efficiency against existing
 109 ALMs and DLMs, but also confirm its scalability on large-scale models. In particular,
 110 with only 3.5M training samples, our SDLMs achieves comparable performance and nearly
 111 2 \times speedup against Qwen-2.5-32B-SFT.

113 **2 RELATED WORK**

115 **2.1 AUTOREGRESSIVE LLMS AND MULTI-TOKEN PREDICTION**

117 Autoregressive large language models (ALMs), such as GPT (Radford et al., 2018; OpenAI, 2022;
 118 2024), LLaMA series (Touvron et al., 2023a;b; Grattafiori et al., 2024; Chiang et al., 2023), Qwen
 119 series (Bai et al., 2023; Yang et al., 2024; Team, 2024; Yang et al., 2025) and other advanced
 120 LLMs (DeepMind, 2025; Team et al., 2025; xAI, 2025; Anthropic, 2023; Liu et al., 2024), generate
 121 text in a token-by-token manner and have demonstrated strong performance across a wide range
 122 of language tasks, including question answering, code generation, mathematical problem solving
 123 and dialogue systems. However, this strictly sequential decoding process limits generation speed.
 124 To mitigate this, KV Cache has been introduced to store previously computed attention keys and
 125 values, avoiding redundant computations and significantly improving inference efficiency.

126 To address the limitations of serial decoding, multi-token prediction (MTP) (Cai et al., 2024;
 127 Gloeckle et al., 2024; Liu et al., 2024) enables the model predict multiple future tokens in parallel
 128 via multiple output heads. These parallel predictions can be used with speculative decoding (Xia
 129 et al., 2022; Stern et al., 2018) to validate multiple candidates and greatly reduce forward steps. For
 130 example, DeepSeek-V3 shows up to 3 \times faster inference with MTP with speculative decoding.

131 **2.2 DIFFUSION LANGUAGE MODELS**

133 Recent diffusion models have shown increasing potential in language tasks. Masked discrete diffusion
 134 models (MDMs) (Zheng et al., 2023; Gong et al., 2024; Ou et al., 2024; Nie et al., 2024) have
 135 achieved perplexity comparable to ALMs. LLaDA (Nie et al., 2025a) further scales MDMs to 8B parameters,
 136 matching state-of-the-art ALMs. Dream (Ye et al., 2025) adopts shifted prediction and autoregressive
 137 initializes, effectively reducing training costs while also delivering strong performance. Block
 138 Diffusion (Arriola et al., 2025) introduces block-level generation for variable-length decoding
 139 with KV cache reuse. Gemini Diffusion (Google DeepMind, 2025) and Seed Diffusion (Song et al.,
 140 2025) further improve speed while narrowing the gap with ALMs.

141 Although recent acceleration technologies such as dKV-Cache (Ma et al., 2025), Fast-dLLM (Wu
 142 et al., 2025), and dLLM-Cache (Liu et al., 2025) attempt to use approximate KV caching mechanisms
 143 to accelerate DLM inference, these methods still suffer from substantial computational overhead
 144 caused by padding the sequence to the maximum sequence length for each forward computation.

146 **3 METHODS**

149 **3.1 PRELIMINARY AND NOTATION**

151 In autoregressive large language models (ALMs) (OpenAI, 2024; Grattafiori et al., 2024; Yang et al.,
 152 2025; DeepMind, 2025), text generation is typically modeled as a conditional probability chain,
 153 referred to as the next-token prediction paradigm. Given a sequence of input tokens $\{x^1, \dots, x^L\}$,
 154 the objective is to minimize the cross-entropy loss:

$$155 \mathcal{L}_{\text{ALM}}(x; \theta) = -\mathbb{E}_x \left[\sum_{i=1}^L \log P_\theta(x^i | x^{<i}) \right], \quad (1)$$

157 where the model $P_\theta(\cdot | x^{<i})$ aims to maximizes the conditional probability of the current word by
 158 leveraging the preceding context $x^{<i} = x^0, \dots, x^{i-1}$.

160 In contrast, diffusion language models (DLMs) (Ou et al., 2024; Nie et al., 2025a; Ye et al., 2025)
 161 generate outputs by progressively denoising from a fully noisy state in parallel. Block Diffusion (Ar-
 162 riola et al., 2025) are a specialized DLM variant that constrains the diffusion operation to proceed

sequentially in blocks. 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 (denoted as $[m]$) within a block of length D , formally defined as:

$$\mathcal{L}_{\text{BD}}(X; \theta) = - \sum_{i=1}^{L/D} \mathbb{E}_{t \sim [0,1]} \mathbb{E}_q \frac{\alpha'_t}{1 - \alpha_t} \log P_\theta(X^i | X^{<i}, X_t^i), \quad (2)$$

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

3.2 SEQUENTIAL DIFFUSION LANGUAGE MODELS

In DLMs, the entire sequence is predicted in parallel based on confidence scores. This can result in premature and inaccurate predictions for later tokens, imposing greater demands on the model's robustness. But predictions for tokens at lower position indices generally benefit from more reliable contextual information and introduce less bias (Wang et al., 2024). Meanwhile, the distribution of certainty and semantics varies across the entire sequence. To this end, we introduce the *Next Sequence Prediction* (NSP) paradigm, which aims to dynamically adjust the size of the decoding sequence at each step based on the difficulty and semantics of the future sequence.

Based on the above understanding, we propose the *Sequential Diffusion Language Models* (SDLM) to reduce error accumulation in diffusion-based generation and improve parallel prediction efficiency. As shown in Figure 1(c), the model adopts bidirectional attention similar to Block Diffusion to understand the semantic information in the future fixed-length noise block X_T^i . Differently, SDLM masks all tokens in the prediction block (masking probability = 1) and is trained by minimizing the cross entropy of all masked tokens. The training objective is formalized as:

$$\mathcal{L}(X; \theta) = -\mathbb{E}_{X, X_T} \left[\frac{1}{D} \sum_i \log P_\theta \left(X^i | x^{<(i-1)}, X_T^i \right) \right], \quad (3)$$

$$X^i = x^{i:(i+D)}, \quad X_T^i = \underbrace{[x^{i-1}, [m], \dots, [m]]}_{D-1},$$

where i denotes a random index within the target sequence, since dynamic length inference makes the decoding start position non-fixed. To better unify next token prediction and block prediction, we continue to employ standard AR's one-position shift between input IDs and labels.

During inference, we introduce *Longest Prefix Decoding*, which uses low-order position priors, to decode the next sequence based on model's confidence. Specifically, at each step, the model perceives history $x^{<(i-1)}$ and produces fixed-length future logits $Z^i = [z_i^1, \dots, z_i^D] \in \mathbb{R}^{D \times |\mathcal{V}|}$ over vocabulary \mathcal{V} , ultimately decoding only the first $\gamma_\tau(Z^i)$ tokens. In the next step, predictions are repeated starting from the previous step's end position. The formalization is as follows:

$$\hat{X}^i = \text{Decode}(Z^i, \gamma_\tau(Z^i)) \quad (4)$$

where $\gamma_\tau(Z^i)$ determines the adaptive sequence length to be decoded (with $1 \leq \gamma_\tau(Z^i) \leq D$), and $\text{Decode}(\cdot)$ denotes extracting the next $\gamma_\tau(Z^i)$ contiguous tokens from Z^i , which we denote as \hat{X}^i . The maximum sequence length function $\gamma_\tau(\cdot)$ is detailed in Section 3.4. This adaptive length mechanism can effectively balance generation efficiency and quality based on text's semantic richness and uncertainty.

3.3 TRAINING

As noted in Section 3.2, when the block size is 1 our model reduces to the autoregressive paradigm, allowing reuse of pretrained ALM weights and cut training costs. From the perspective of instruction fine-tuning, we define the input as $S = [X; Y]$, where X is the prefix and Y the response.

Training. During training, we partition Y into blocks at random positions to train the model's prediction capabilities at different starting positions. As shown in the Equation 3, for a starting position i , we construct a noise block $Y_T^i = [y^{i-1}, [m], \dots, [m]]$ and predict the next fixed-length block

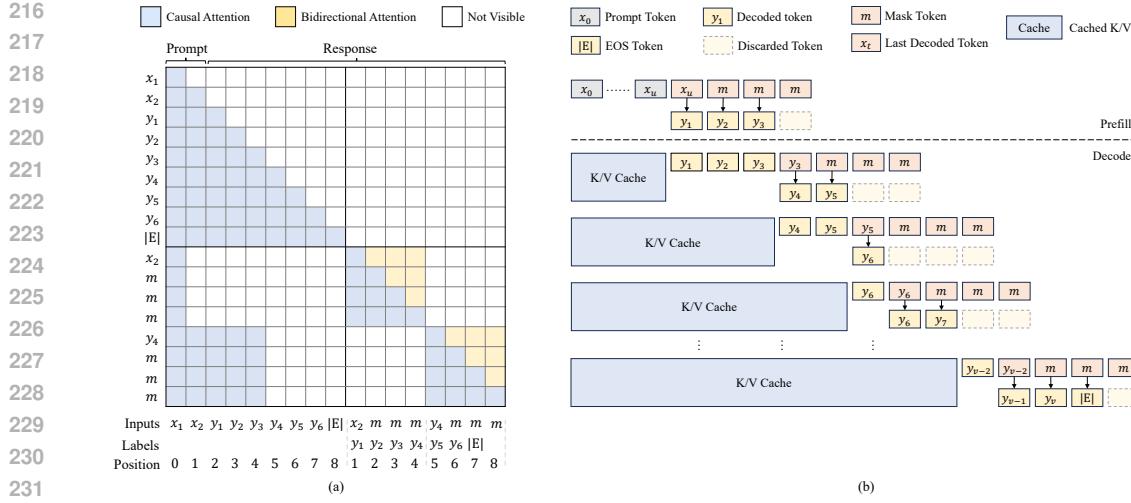


Figure 2: **Structured attention mask for parallel block training and sampling.** (a) Reordered input yields a mask with causal prefix (top-left), visible cross-block prefix (bottom-left), and intra-block bidirectional attention (bottom-right). (b) Confidence-based next sequence prediction with KV reuse. A block of D tokens is predicted with $D-1$ masks. The longest high-confidence subsequence is selected as dynamic output. Cached KV states enable efficient decoding.

$Y^i = y^{i:(i+D)}$ (simplified as $Y^i = [y_i^1, \dots, y_i^D]$) by shifting. A bidirectional attention mechanism is used within the block for feature information, which serves as the basis for decoding dynamic-length sequences. For historical information, we maintain causal attention as ALMs. Therefore, for a single noise block Y_T^i , we can construct a custom attention mask $A \in \{0, 1\}^{(i+D) \times (i+D)}$:

$$A_{uv} = 1_{v \leq u} \oplus 1_{u \geq i \cap v \geq i} \quad (5)$$

This enforces strict causality for $u < i$ and full mutual attention for $u, v \geq i$.

Parallel Training. To enable efficient parallel training, we construct the sequence by interleaving noise blocks and target blocks as:

$$S_T = \text{concat}(X, \underbrace{\mathbb{I}_1 \cdot Y_T^1, Y^1, \dots, \mathbb{I}_i \cdot Y_T^i, Y^i, \dots}_{\text{Block } i}) \quad (6)$$

where $\mathbb{I}_i \in \{0, 1\}$ is a random indicator variable that controls whether a noise block Y_T^i is inserted at the current starting position i to predict the ground-truth block Y^i . Each noise block Y_T^i attends only within itself, while Y^i is visible as prefix to later blocks but not vice versa, ensuring causality through attention constraints and positional encodings.

Since transformers rely on positional encodings, by rearranging S , the attention mask forms three parts as shown in Figure 2: (1) causal attention (top-left), (2) visible prefixes for each block (bottom-left), and (3) bidirectional attention within blocks (bottom-right). To improve training efficiency, we can concatenate any number of noise blocks after the target sequence within max sequence length. The sparse attention structure allows flex attention (Dong et al., 2024) to accelerate training.

3.4 INFERENCE

As described in Equation 4, we introduce the Longest Prefix Decoding method for dynamic length decoding based on low-order position priors. We primarily rely on the model’s confidence in its inferences as the basis to refine the length function $\gamma(\cdot)$, and design two types of decoding strategies:

270 **Greedy Decoding.** We implement γ_τ through a confidence-based stopping rule that identifies the
 271 longest prefix satisfying:

$$273 \quad 274 \quad 275 \quad \gamma_\tau(Z^i) = \max \left(\left\{ j \in \{1, 2, \dots, D\} \mid \prod_{k=1}^j p(z_i^k) \geq \tau \right\} \cup \{1\} \right) \quad (7)$$

276 where $p(z_i^k)$ quantifies confidence at position k (where $z_i^k \in \mathbb{R}^{|\mathcal{V}|}$ is the position- k logit vector), and
 277 τ is a predefined threshold. This approach greedily decodes at most j tokens ($j \geq 1$) whose cumu-
 278 lative product of confidence scores is greater than τ . We explore two distinct confidence functions:

279 (1) **Logit Value Confidence.** This metric uses the softmax probability of the decoded token v at
 280 position k , to capture the model’s per-token confidence in its top prediction:

$$282 \quad p_{\text{logit}}(z_i^k) = \text{softmax}(z_i^k)_v \quad (8)$$

284 (2) **Entropy-Normalized Confidence.** While p_{logit} provides a pointwise confidence signal, it over-
 285 looks distributional ambiguity. Inspired by [Wang et al. \(2025\)](#) that higher predictive entropy corre-
 286 lates with forking behavior during generation, we employ an entropy-based confidence score:

$$288 \quad 289 \quad 290 \quad p_{\text{entropy}}(z_i^k) = 1 - \frac{H_p}{\log |\mathcal{V}|}, \quad \text{where} \quad H_p = -\sum_n p_n \log p_n \quad (9)$$

291 Here, p_n is the softmax probability of the n -th word. Then, the entropy H_p is normalized by $\log |\mathcal{V}|$.
 292 Lower entropy indicates higher confidence, while higher entropy reflects more uncertainty.

293 **Self-Speculative Decoding.** Following the speculative decoding ([Stern et al., 2018](#)), we decode
 294 multiple tokens in parallel and verify their correctness through self-consistency checks. In each step,
 295 the model produces D speculative tokens $\hat{Y}^i = [\hat{y}_i^1, \dots, \hat{y}_i^D]$ (where \hat{y}_i^k denote the k -th decoded
 296 token of block i) in an initial forward pass. To validate them, D verification inputs are constructed
 297 by progressively extending prefixes of the sampled tokens, appending mask $[m]$ at the first unverified
 298 position and padding to form a batch. A second forward pass then yields corresponding predictions
 299 $\tilde{Y}^i = [\tilde{y}_i^1, \dots, \tilde{y}_i^D]$. The decoding sequence length is determined by the consistency-driven function:

$$301 \quad 302 \quad \gamma_{\text{verify}}(Z^i) = \max \left(\left\{ j \in \{1, 2, \dots, D\} \mid \hat{y}_i^j = \tilde{y}_i^j \right\} \cup \{1\} \right) \quad (10)$$

303 Compared to confidence-based truncation via γ_τ , which relies on local heuristics, self-speculative
 304 decoding performs explicit consistency checks for self-verification without external models, offering
 305 greater reliability at the cost of an additional forward pass.

4 EXPERIMENTS

4.1 SETTING

311 To ensure a fair comparison, we fine-tune the Qwen-2.5 base model ([Team, 2024](#)) with all open-
 312 source instruction datasets (3.5 million samples, 2.3 billion tokens), covering math, code, and
 313 instruct-following. We compare SDLM against same-scale ALMs (Qwen2.5-3B/32B-Instruct, fine-
 314 tuned verison of Qwen2.5-3B/32B under the same setting), and larger DLMs like Dream-7B-Instruct
 315 and LLaDA-8B-Instruct across benchmarks spanning general, mathematics, science, and coding
 316 tasks. All evaluated with OpenCompass ([Contributors, 2023](#)) under standardized settings. Details
 317 about training and evaluating can be found in Appendix A.

4.2 MAIN RESULTS

321 Table 1 shows the performance and inference efficiency of our SDLM, trained in a single epoch on
 322 only 3.5M samples. SDLM-32B attains 92.4 on GSM8K, 74.2 on MATH-500, and 78.6 on IFEval,
 323 while remaining competitive on coding tasks. The SDLM-3B performs on par with or even surpasses
 Qwen-2.5-3B-SFT, and significantly outperforms larger DLMs such as LLaDA-8B and Dream-7B.

324
 325 **Table 1: Performance of instruct models across 8 long-form tasks.** Numbers in parentheses (#)
 326 denote the speedup ratio: average tokens per pass vs. ALMs (1 token per pass). Results marked by
 327 \dagger and \ddagger are from [Team \(2024\)](#) and [Ye et al. \(2025\)](#) respectively. “–” indicates unknown data.

Model Name	GSM8K	MATH	GPQA	HumanEval	HumanEval+	MBPP	MBPP+	IFEval	Avg.	
ALMs										
Qwen-2.5-3B \dagger	86.7	65.9	30.3	74.4	–	72.7	–	58.2	–	
Qwen-2.5-3B-SFT	85.8	59.8	27.8	73.8	60.4	68.5	42.6	62.1	60.1	
Qwen-2.5-32B \dagger	95.9	83.1	49.5	88.4	–	84.0	–	79.5	–	
Qwen-2.5-32B-SFT	93.2	74.8	33.8	82.9	76.2	82.1	59.0	76.5	72.3	
DLMs										
LLaDA-8B \ddagger	78.6	26.6	31.8	47.6	–	34.2	–	59.9	–	
Dream-7B \ddagger	81.0	39.2	33.0	55.5	–	58.8	–	62.5	–	
SDLM-3B ($D = 4$)	$\tau = .98$	84.6 (2.15)	60.8 (2.18)	28.3 (2.26)	67.1 (1.91)	59.8 (1.76)	65.4 (1.66)	40.5 (1.78)	57.1 (1.38)	57.9 (1.89)
SDLM-3B ($D = 4$)	$\tau = .82$	84.5 (2.75)	57.8 (2.73)	28.3 (2.66)	66.5 (2.53)	60.4 (2.25)	65.0 (2.30)	40.0 (2.29)	55.8 (1.58)	57.3 (2.39)
SDLM-32B ($D = 4$)	$\tau = .98$	92.4 (2.15)	74.2 (2.35)	36.4 (2.34)	81.1 (2.05)	73.8 (2.29)	80.9 (1.56)	58.2 (1.51)	78.6 (1.25)	71.9 (1.94)
SDLM-32B ($D = 4$)	$\tau = .82$	92.3 (2.71)	73.0 (2.88)	36.9 (2.61)	79.9 (2.82)	73.2 (2.72)	80.9 (2.17)	57.1 (2.25)	78.2 (1.43)	71.4 (2.45)

343
 344 In terms of generation efficiency, SDLM generate about 2 tokens per forward pass, reducing latency
 345 to about two-thirds of comparable ALMs. Taking GSM8K as an example, SDLM-32B at $\tau = 0.98$
 346 achieves accuracy 92.4 (vs. 93.2 for its same-scale SFT counterpart) while generating 2.15 tokens
 347 per step. Lowering τ to 0.82 further increases token output to 2.71 with only a 0.1 pct accuracy drop,
 348 highlighting an attractive speed-accuracy tradeoff. SDLM-3B follows a similar trend on GSM8K
 349 with minimal performance drop as τ is lowered. This trend holds across all benchmarks, where
 350 lowering τ consistently increases token generation while maintaining competitive performance. The
 351 effect and robustness of different τ values are ablated in Section 4.3.

352 In terms of short-answer benchmarks
 353 shown in Table 2, SDLM-32B performs
 354 within 1 ptc of its autoregressive counter-
 355 part across MMLU, Winogrande, and
 356 Hellaswag, while SDLM-3B matches
 357 Qwen-2.5-3B-SFT on these benchmarks.
 358 This demonstrates that SDLM
 359 retains the semantic and reasoning abil-
 360 ies of the base ALMs while enabling
 361 more efficient parallel decoding, con-
 362 firming that our diffusion training pre-
 363 serves the base model’s NTP capability.

364 Overall, SDLM delivers “near-SFT ac-
 365 curacy with significant inference accel-
 366 eration” at both 3B and 32B scales, proving that NSP generation can stably converge in large-model
 367 regimes and providing a solid foundation for future work with larger parameters, longer training,
 368 and wider blocks.

370 4.3 TRADE-OFF BETWEEN SPEEDUP AND PERFORMANCE

371 Existing DLMs ([Nie et al., 2025a](#); [Ye et al., 2025](#)) exploit parallel token generation but face a key
 372 trade-off: generating one token per step maintains quality, while producing multiple tokens often
 373 degrades it. Moreover, the reliance on fixed-length noise sequences constrains flexibility and limits
 374 practical efficiency gains over ALMs. In contrast, SDLM only concatenate a block-length masks
 375 per step, incurring minimal overhead compared to NTP inference.

377 Figure 3 shows the speed-performance trade-off with varying confidence threshold τ across
 378 GSM8K, MATH-500 and HumanEval+. As τ decreases, SDLM generates more tokens per step,

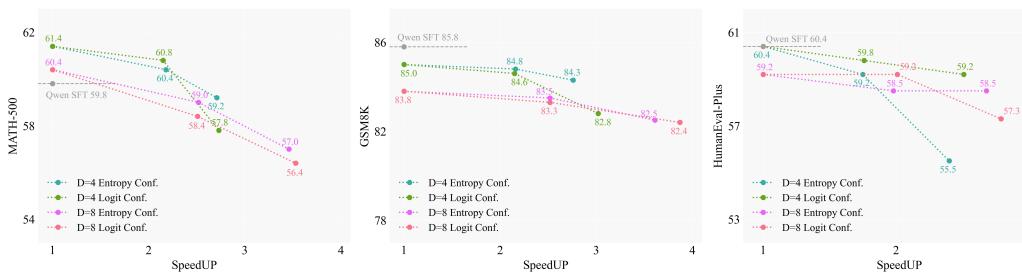


Figure 3: **Trade-off between performance and speed under different inference setting for SDLM-3B ($D = 4$) and SDLM-3B ($D = 8$).** Adjusting τ allows a controllable trade-off between speed and performance. SpeedUp denotes the average number of tokens output per forward pass.

Table 3: **SDLM-3B ($D = 8$) with larger block size and sampling with self-speculative decoding.** (a) Larger blocks yield higher throughput with only minimal performance degradation. (b) With self-speculative decoding, the average accepted tokens per step (in green) significantly exceeds greedy decoding with threshold (Conf. τ).

Model Name	GSM8K	MATH	HumanEval+	MBPP	MBPP+	Avg.
Qwen-2.5-3B-SFT (AR)	85.8	59.8	60.4	68.5	42.6	63.4
SDLM-3B ($D = 4$)	84.6 (2.15)	60.8 (2.18)	59.8 (1.76)	65.4 (1.66)	40.5 (1.78)	62.2 (1.91)
	85.1 (3.62)	61.2 (3.54)	58.4 (3.40)	65.8 (3.29)	40.5 (3.23)	62.2 (3.42)
SDLM-3B ($D = 8$)	83.3 (2.52)	58.4 (2.51)	59.2 (2.01)	64.2 (1.71)	39.7 (2.16)	61.0 (2.18)
	83.6 (5.99)	60.2 (5.73)	57.3 (5.18)	64.2 (4.84)	39.4 (5.33)	60.9 (5.41)

achieving up to $3.5\times$ speed-up. On math tasks like MATH-500, accuracy remains stable ($61.4 \rightarrow 59.2$) as long as tokens per step stay under 3. Code tasks like HumanEval+ are more sensitive, with performance remaining high at around 1.7 tokens per step ($60.4 \rightarrow 59.8$).

Furthermore, we compare the effects of generation block size D and confidence functions (Logit *vs.* Entropy). Results show that $D = 4$ generally yields slightly better accuracy, while the new trained model SDLM-3B ($D = 8$) enables greater acceleration due to larger parallel generation capacity. Both confidence function-based schemes maintain good performance. The threshold τ provides a flexible balance between speed and performance across various tasks and configurations.

4.4 ABLATION STUDY

We further explore larger D and self-speculative decoding (Table 3), alternative model designs (Appendix C), and comparisons with other speed-up methods (*e.g.*, MTP; Appendix D). We retrain all variants under the same settings to assess their impact on both performance and efficiency.

Block Size. We investigate the impact of larger block sizes on SDLM-3B in Table 3, focusing on the new trained model SDLM-3B ($D = 8$). Compared to $D = 4$, the $D = 8$ configuration delivers substantially higher throughput with comparable model performance. Under Conf. $\tau = 0.98$, the average number of output tokens per step increases from 1.9 ($D = 4$) to 2.2 ($D = 8$), with only a 1.2-point drop in overall accuracy. Increasing D from 4 to 8 boosts the accepted tokens with only a small quality drop, suggesting potential for further throughput gains.

Self-Speculative Decoding. We further evaluate self-speculative decoding in Table 3. In the *Speculative* rows, with $D = 4$ and $D = 8$, SDLM accepts an average of 3.4 and 5.4 tokens per step, corresponding to roughly 85% and 68% of the proposal budget, respectively. Model performance remains comparable across settings. However, this method incurs additional validation overhead, differing

```

432
433 User: What fraction of 2 feet is 3 inches? Express your answer as a common fraction.
434 Please reason step by step, and put your final answer within \boxed{}.
435
436 SDLM: To find the fraction of 2 feet that is 3 inches , we need to convert the feet
437 to inches and then divide the by the total number.
438 1 foot = 12 inches
439 2 feet = 2 * 12 = 24 inches
440 Now, we can the 3 inches by the fraction of 24 inches:
441 
$$\boxed{\frac{3}{24}} = \frac{1}{8}$$

442 So, the answer is 1/8.
443 
$$\boxed{\frac{1}{8}}$$
 38 step with 113 tokens
444
445 User: Write a Fibonacci function in Python.
446
447 SDLM: def fibonacci(n):
448     if n == 0:
449         return 0
450     elif n == 1:
451         return 1
452     else:
453         return fibonacci(n-1) + fibonacci(n-2)  
#(fib
454
455
456

```

Figure 4: **Visualization of the sampling process.** Where each blue block indicates a subsequence generated in a single decoding step.

in experimental setup compared to the other two decoding methods. Despite this, it substantially enhances the model’s responsiveness, demonstrating its potential under specific conditions.

Case Study. Figure 4 illustrates SDLM’s flexible decoding, where the generated sequence length adapts to local context. In fluent or structured regions (e.g. math expressions, structured code segments, common phrases), it confidently emits longer sequences at once. While facing uncertainty or forking, it slows down with shorter outputs. This adaptive strategy balances speed with precision.

5 CONCLUSION

In conclusion, we propose *Next Sequence Prediction* (NSP), a unified framework bridging autoregressive and diffusion decoding. Building on NSP, we develop *Sequential Diffusion Language Models* (SDLMs) that adapt pretrained ALMs via parallel block training and dynamic decoding. SDLM matches SFT-tuned ALMs in performance while decoding faster, offering a stronger speed–performance trade-off. We hope this work inspires further exploration of unified sequence generation.

471 REPRODUCIBILITY & ETHICS STATEMENT

472 All data are publicly available, and code is provided in an anonymous repository (<https://anonymous.4open.science/r/SDLM-112358/readme.md>) for full reproducibility. 473 This work follows the ICLR Code of Ethics, upholding integrity, fairness, and respect for privacy.

477 REFERENCES

478 Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo,
479 Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav,
480 Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo
481 Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and
482 Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model,
483 2025. URL <https://arxiv.org/abs/2502.02737>.

484 Anthropic. Introducing Claude, 2023. URL <https://www.anthropic.com/index/introducing-claude>.

486 Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Sub-
 487 ham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autore-
 488 gressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

489

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

493

494 Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge,
 495 Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

496

497 Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri
 498 Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv
 preprint arXiv:2401.10774*, 2024.

499

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

503

504 Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake,
 505 and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The Inter-
 506 national Journal of Robotics Research*, pp. 02783649241273668, 2023.

507

508 Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng,
 509 Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An
 510 open-source chatbot impressing GPT-4 with 90%* ChatGPT quality, March 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>.

511

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

515

516 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
 517 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to
 518 solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

519

520 OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models.
 521 <https://github.com/open-compass/opencompass>, 2023.

522

523 Google DeepMind. Gemini 2.5, 2025. URL <https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/>.

524

525 Yuyang Ding, Xinyu Shi, Xiaobo Liang, Juntao Li, Qiaoming Zhu, and Min Zhang. Unleash-
 526 ing reasoning capability of llms via scalable question synthesis from scratch. *arXiv preprint
 527 arXiv:2410.18693*, 2024.

528

529 Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A
 530 programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*,
 531 2024.

532

533 Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Syn-
 534 naeve. Better & faster large language models via multi-token prediction. *arXiv preprint
 535 arXiv:2404.19737*, 2024.

536

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

540

541 Google DeepMind. <https://blog.google/technology/google-deepmind/gemini-diffusion/>. <https://blog.google/technology/google-deepmind/gemini-diffusion/>, 2025.
 542 Accessed: 2024-07-24.

540 Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad
 541 Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd
 542 of models. *arXiv preprint arXiv:2407.21783*, 2024.

543

544 Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob
 545 Steinhardt. Measuring massive multitask language understanding. In *ICLR*. OpenReview.net,
 546 2021a.

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

550

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

553

554 Siming Huang, Tianhao Cheng, Jason Klein Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang,
 555 J. H. Liu, Chenchen Zhang, Linzheng Chai, Ruiyuan Yuan, Zhaoxiang Zhang, Jie Fu, Qian Liu,
 556 Ge Zhang, Zili Wang, Yuan Qi, Yinghui Xu, and Wei Chu. Opencoder: The open cookbook for
 557 top-tier code large language models. 2024. URL <https://arxiv.org/pdf/2411.04905.pdf>.

558

559 Ivan Kapelyukh, Vitalis Vosylius, and Edward Johns. Dall-e-bot: Introducing web-scale diffusion
 560 models to robotics. *IEEE Robotics and Automation Letters*, 8(7):3956–3963, 2023.

561

562 Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E.
 563 Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model
 564 serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating
 Systems Principles*, 2023.

565

566 Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brah-
 567 man, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik,
 568 Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm,
 569 Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu
 570 3: Pushing frontiers in open language model post-training. 2024.

571

572 Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman,
 573 Dongmei Zhang, and Surajit Chaudhuri. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv
 574 preprint arXiv:2310.09263*, 2023.

575

576 Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference
 577 acceleration of large language models via training-time test. In *Annual Conference on Neural
 578 Information Processing Systems*, 2025.

579

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

583

584 Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chat-
 585 gpt really correct? rigorous evaluation of large language models for code generation. *Advances
 586 in Neural Information Processing Systems*, 36:21558–21572, 2023.

587

588 Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, and
 589 Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching,
 590 2025. URL <https://github.com/maomaocun/dLLM-cache>.

591

592 Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion
 593 language models. *arXiv preprint arXiv:2505.15781*, 2025.

594

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

597

598 Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai
 599 Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint
 600 arXiv:2502.09992*, 2025a.

594 Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai
 595 Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint*
 596 *arXiv:2502.09992*, 2025b.

597

598 OpenAI. Introducing ChatGPT, 2022. URL <https://openai.com/index/chatgpt/>.

599

600 OpenAI. Hello GPT-4o, 2024. URL <https://openai.com/index/hello-gpt-4o/>.

601 Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan
 602 Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data.
 603 *arXiv preprint arXiv:2406.03736*, 2024.

604

605 Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language under-
 606 standing by generative pre-training. Technical report, OpenAI, 2018.

607

608 Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language
 609 models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

610

611 David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Di-
 612 rani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a bench-
 613 mark. In *First Conference on Language Modeling*, 2024.

614

615 Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-
 616 resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF confer-
 617 ence on computer vision and pattern recognition*, pp. 10684–10695, 2022.

618

619 Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An ad-
 620 versarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, 2021.

621

622 Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and
 623 Mehrdad Farajtabar. Your llm knows the future: Uncovering its multi-token prediction potential.
 624 *arXiv preprint arXiv:2507.11851*, 2025.

625

626 Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang
 627 Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with
 628 high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

629

630 Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autore-
 631 gressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

632

633 Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej,
 634 Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical
 635 report. *arXiv preprint arXiv:2503.19786*, 2025.

636

637 Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL <https://qwenlm.github.io/blog/qwen2.5/>.

638

639 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
 640 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and
 641 efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

642

643 Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko-
 644 lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open founda-
 645 tion and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

646

647 David Wadden, Kejian Shi, Jacob Morrison, Aakanksha Naik, Shruti Singh, Nitzan Barzilay, Kyle
 648 Lo, Tom Hope, Luca Soldaini, Shannon Zejiang Shen, et al. Sciriff: A resource to enhance
 649 language model instruction-following over scientific literature. *arXiv preprint arXiv:2406.07835*,
 650 2024.

651

652 Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen,
 653 Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive
 654 effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

648 Weiyun Wang, Zhe Chen, Wenhui Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Jinguo Zhu,
 649 Xizhou Zhu, Lewei Lu, Yu Qiao, et al. Enhancing the reasoning ability of multimodal large
 650 language models via mixed preference optimization. *arXiv preprint arXiv:2411.10442*, 2024.

651

652 Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song
 653 Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache
 654 and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025.

655 xAI. Grok 3 beta — the age of reasoning agents, 2025. URL <https://x.ai/news/grok-3>.

656

657 Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative de-
 658 coding: Exploiting speculative execution for accelerating seq2seq generation. *arXiv preprint*
 659 *arXiv:2203.16487*, 2022.

660 An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li,
 661 Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint*
 662 *arXiv:2407.10671*, 2024.

663

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

667 Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng
 668 Kong. Dream 7b, 2025. URL <https://hkunlp.github.io/blog/2025/dream>.

669

670 Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a ma-
 671 chine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

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

674

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

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

A DETAILS OF TRAINING

We show the training hyperparameters in Table 4.

Table 4: Training Hyperparameters for SDLM.

Parameter	SDLM-3B	SDLM-32B
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

The training corpus comprises with: Tulu-3-SFT-Mixture (Lambert et al., 2024), Table-GPT (Li et al., 2023), SciRIFF (Wadden et al., 2024), SmolTalk (Allal et al., 2025), OPC-SFT-Stage2 (Huang et al., 2024), and ScaleQuest-Math (Ding et al., 2024), with a combined total of 3.5 million samples (~ 2.3 billion tokens).

To comprehensively evaluate the capabilities of SDLM, we conduct evaluations across a diverse set of benchmarks encompassing:

General Tasks. MMLU (Hendrycks et al., 2021a)(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).

Mathematics & Science Tasks. GSM8K (Cobbe et al., 2021) (0-shot), MATH-500 (Hendrycks et al., 2021b)(0-shot), GPQA (Rein et al., 2024) (0-shot).

Coding Tasks. 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).

B PSEUDOCODE FOR SDLM

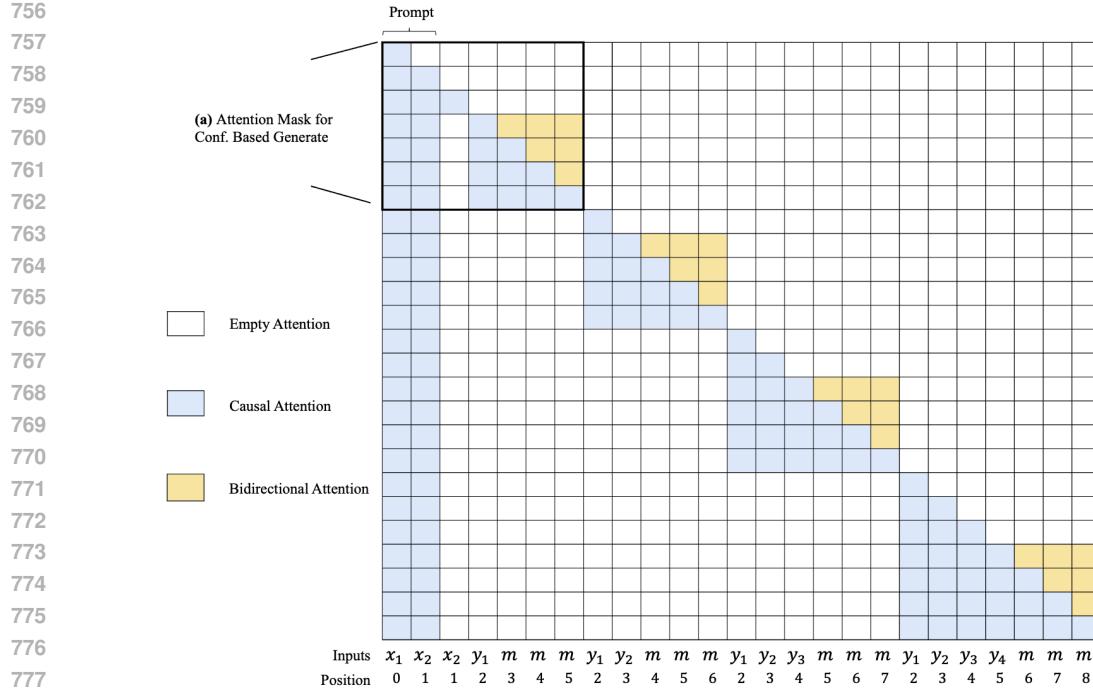
We now include pseudocode for SDLM training (Algorithm 1), confidence-based decoding (Algorithm 2), and self-speculative decoding (Algorithm 3). The corresponding input IDs, position IDs, and attention masks are shown in Figure 2 (a) (training) and Figure 5 (inference). For confidence-based decoding, the relevant mask is the bold upper-left block in Figure 5.

Algorithm 1 Supervised Fine-Tuning of SDLM

Require: Model \mathcal{M} , Sequence x , block size D

Ensure: Parameters θ

- 1: **repeat**
- 2: $S_T \leftarrow$ construct input via Eq. 3, Eq. 6
- 3: $S_0 \leftarrow$ ground-truth labels
- 4: PE \leftarrow block-wise position ids aligned with S_T
- 5: A \leftarrow Fig. 2 (a)
- 6: $\hat{S}_0 = \mathcal{M}(S_T, \text{PE}, \text{A})$
- 7: $\mathcal{L}(S_T, S_0) = \text{CE}_{\text{block}}(\hat{S}_0, S_0)$
- 8: $\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L}$
- 9: **until** converged
- 10: **return** θ



779 Figure 5: **Structured attention mask for sampling.** (a) The bold upper-left block shows the mask
 780 used during confidence-based generation; the full mask shows the mask used in self-speculative
 781 decoding.

783 **Algorithm 2** Conf Generate

784 **Require:** Model \mathcal{M} , prompt x , block size D
 785 **Ensure:** Generated sequence y

786 1: Initialize KV-cache; $y \leftarrow x$; $L \leftarrow |x|$
 787 2: **while** not EOS **do**
 788 3: $x_{in} \leftarrow [y, y_{-1}, \text{MASK}^{\times(D-1)}]$ ▷ prepare diffusion block
 789 4: $\text{PE}_{in} \leftarrow \text{range}(0, L) \parallel [L-1] \parallel \text{range}(L-1, L+D-2)$ ▷ revise pe for
 790 5: $(\text{logits}, \text{KV}_{\text{new}}) \leftarrow \mathcal{M}(x_{in}, \text{KV}_{\text{cache}}, \text{PE}_{in})$
 791 6: $\text{KV}_{\text{cache}} \leftarrow \text{truncate KV}_{\text{new}} \text{ to positions } 0:L$ ▷ only update causal KV
 792 7: $\hat{d}_{1:D} \leftarrow \text{last } D\text{-token predictions from logits}$ ▷ candidate block
 793 8: $k \leftarrow \max\{j \in [1, D] \mid \text{confidence}(\hat{d}_{1:j}) \geq \tau\}$ ▷ longest prefix accepted
 794 9: $y \leftarrow y \parallel \hat{d}_{1:k}$; $L \leftarrow L + k$ ▷ append accepted tokens
 795 10: **if** $\hat{d}_{1:k}$ contains EOS **then**
 796 11: **break**
 797 12: **end if**
 798 13: **end while**
 799 14: **return** y

800
801
802
803
804
C MORE ABLATION STUDY OF MODEL DESIGN

805 **No Shift Prediction.** To verify the effectiveness of shift prediction, we employ a method similar
 806 to LLaDA to directly predict the original tokens at the mask location. As shown in Figure 6, under
 807 the same training cost, this method leads to a noticeable decline in model performance, with Hu-
 808 manEval+ scores dropping by approximately 14 points. After log analysis, we find that the model
 809 has more repeated outputs. This indicates that the shift prediction method exploits the strong ability
 of ALMs to predict the first token and provides a stable starting point for diffusion decoding.



Figure 6: Ablation on attention mask type and prediction shift strategy. We conduct the following ablation experiments: (1) No shift: predicting x_t instead of x_{t+1} ; (2) Leisure precautions: using a causal mask instead. The left image shows its model performance, while the right image shows the acceleration ratio.

Causal Attention. As shown in Figure 6, we replace bidirectional attention inside each block with a causal (unidirectional) masking. 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 decreases from 1.88 to 1.82, indicating that bidirectional attention enlarges the local receptive field during decoding and improves parallel generation efficiency.

864
 865 **Table 5: SDLM with larger block size and sampling with self-speculative decoding.** SC.:
 866 Performance; SuP (SpeedUp \times): average accepted tokens per step; TPS: actual throughput, denotes
 867 as Equation 11. In addition, BD denotes the length of draft tokens (num of heads in MTPs and
 868 decoding window size of SDLM).

Model	GSM8K			MATH			HumanEval+			MBPP			Avg.			
	SC.	SuP	TPS	SC.	SuP	TPS	SC.	SuP	TPS	SC.	SuP	TPS	SC.	SuP	TPS	
ALMs																
Vicuna-7B-v1.5 ¹	Vanilla	11.8	1.00	40.00	2.0	1.00	40.26	16.5	1.00	39.01	38.1	1.00	40.21	—	1.00	39.87
Qwen2.5-3B-SFT	Vanilla	86.0	1.00	30.95	60.8	1.00	31.22	61.0	1.00	31.32	68.5	1.00	31.00	69.1	1.00	31.12
Qwen2.5-3B-SFT	vLLM	85.6	1.00	138.15	62.2	1.00	138.29	59.2	1.00	137.05	70.0	1.00	132.46	69.3	1.00	136.49
MTPs																
Medusa-V1.0-7B ²	BD=6	10.4	3.64	119.15	2.0	3.82	128.40	16.5	3.90	127.74	26.1	4.25	103.40	—	3.90	119.67
Qwen2.5-3B-Eagle3	BD=16	77.41	5.78	142.29	57.60	6.13	182.31	59.15	5.71	146.37	67.70	5.24	139.39	65.45	5.72	152.59
Qwen2.5-3B-Eagle3	BD=32	78.51	8.01	256.97	59.60	8.65	190.39	58.54	7.73	161.99	68.48	7.18	157.33	66.28	7.89	191.67
DLMs																
SDLM-3B ($D=4$)	BD=4	85.0	3.62	98.52	60.4	3.57	98.58	59.2	3.44	93.49	65.4	3.35	86.11	67.5	3.50	94.18
	BD=8	85.5	5.34	143.30	60.0	5.12	137.49	59.8	4.88	129.43	65.4	4.51	111.31	67.7	4.96	130.38
SDLM-3B ($D=8$)	BD=8	83.6	5.99	161.00	59.6	5.73	155.53	57.9	5.37	141.47	64.2	5.09	125.27	66.3	5.55	145.82
	BD=16	84.2	7.30	176.44	60.2	6.82	166.83	59.2	6.33	152.69	65.4	5.68	126.39	67.3	6.53	155.59

881 D COMPARISON WITH MULTI-TOKEN PREDICTION

882
 883 SDLM can be viewed through the lens of multi-token prediction (MTP) as well. Both SDLM and
 884 MTP parallelize autoregressive generation by predicting multiple tokens in a single forward pass.
 885 For a prediction horizon of D tokens, MTP use D separate output heads, with the i -th head predicting
 886 the token at position $m + i$. Similarly, SDLM uses D positions in the input sequence: the last token
 887 (at position m) and $D - 1$ mask tokens. The prediction at the last token position corresponds
 888 to the token at $t + 1$ (equivalent to MTP’s first head), and the prediction at the j -th mask token
 889 ($1 \leq j \leq D - 1$) corresponds to the token at $m + 1 + j$ (equivalent to MTP’s $(j + 1)$ -th head).

890 However, SDLM introduces two advantages. First, the predictions are generated within a local bi-
 891 directional attention window, enabling joint context utilization across the predicted tokens. This con-
 892 trasts with MTP’s isolated head (Cai et al., 2024; Gloeckle et al., 2024) or left-to-right attention (Liu
 893 et al., 2024). Second, extending the prediction horizon requires no architectural modification: ap-
 894 pending additional mask tokens suffices, while MTP necessitates adding new output heads.

895 To ensure a fair comparison with MTP style methods with speculative decoding, we evaluate the
 896 following models:

- 897 • Autoregressive: both vanilla HuggingFace `transformers.generate()` (PyTorch
 898 backend with pre-allocated KV-cache) and the vLLM Kwon et al. (2023) acceleration
 899 framework;
- 900 • Medusa Cai et al. (2024): A method accelerating generation by employing a few additional
 901 decoding heads; evaluated using Medusa-V1.0-7B and its base model, Vicuna-7B-v1.5;
- 902 • Eagle-3 Li et al. (2025): A SoTA multi-token prediction with speculative decoding; eval-
 903 uated using trained Qwen2.5-3B-Eagle3 with our training data;
- 904 • SDLM (Ours): Our method of self-speculative decoding with enhanced KV-cache support.

905 We compute TPS with actual generated tokens and wall-clock inference time as follows:

$$906 \quad \text{TPS} \approx \frac{\sum_{\text{num_samples}} \text{actual_generate_tokens}}{\text{wall-clock inference time}} \quad (11)$$

913 As shown in Table 5, we first compare SDLM with the SFT baseline model trained under the same
 914 setting, SDLM exhibits only a small drop in performance across the four benchmarks, but already
 915 achieves about a $3 \times$ end-to-end speedup over vanilla decoding, when both the training and inference

916 ¹<https://huggingface.co/lmsys/vicuna-7b-v1.5>

917 ²<https://huggingface.co/FasterDecoding/medusa-v1.0-vicuna-7b-v1.5>

918 windows size are set to 4. Furthermore, for SDLM-3B ($D = 8$), when we increase the inference
 919 window size to 16, we observe an even longer effective decoding length without noticeable performance
 920 degradation, and the actual speedup reaches about $5\times$ compared with vanilla decoding, while
 921 remaining highly comparable to vLLM across the four evaluated tasks.

922 For comparison with MTP approaches, Medusa achieves nearly a $3\times$ speedup over its base model,
 923 and the state-of-the-art speculative decoding model Eagle-3 also demonstrates close to a $5\times$ speedup
 924 (see their Table 1 on GSM8K for a fair comparison). In our evaluation, when using a block window
 925 of 16, SDLM achieves acceleration comparable to Eagle-3 (TPS 155 vs. 152), while exhibiting
 926 slightly lower speedup compared with Eagle-3 using a larger block size of 32.

927 These results indicate that SDLM can achieve excellent acceleration when combined with speculative
 928 decoding. Moreover, SDLM naturally generalizes to larger speculative windows that are never
 929 seen during training, and the bidirectional attention design may further enhance modeling capacity.
 930 We believe that with more carefully optimized training objectives and attention kernels, SDLM can
 931 become even faster and more accurate in the future.

932 All results in Table 5 are evaluated using the standard OpenCompass settings, and the wall-clock
 933 inference time is measured by simply placing a timer around the `generate` call.

936 E USE OF LARGE LANGUAGE MODELS

937 We used large language models (LLMs) solely for assisting with language polishing and minor
 938 writing support. All research ideas and analyses were conceived and developed by the authors.

940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 970
 971