SDAR: A Synergistic Diffusion-AutoRegression Paradigm for Scalable Sequence Generation

Shuang Cheng[1,2*] **Yihan Bian**[3*] **Dawei Liu**[2,4*] **Yuhua Jiang**[5] **Yihao Liu**[5]
Linfeng Zhang[4] **Qian Yao**[2] **Zhongbo Tian**[2] **Wenhai Wang**[2] **Qipeng Guo**[2]
Kai Chen[2] **Biqing Qi**[2†] **Bowen Zhou**[2†]
[1]Zhejiang University [2]Shanghai AI Laboratory [3]University of Maryland, College Park
[4]Shanghai Jiao Tong University [5]Tsinghua University
{chengshuang, qibiqing}@pjlab.org.cn

Abstract

Autoregressive (AR) language modeling remains the dominant paradigm due to its dense supervision signal and highly optimized serving infrastructure, but its strictly causal, token-by-token decoding limits parallelism and non-causal modeling. While masked diffusion offers a promising path toward parallel generation, it faces two critical bottlenecks: training inefficiency stemming from sparse masked objectives, and high latency caused by iterative whole-sequence denoising. We present a systematic study of blockwise discrete diffusion, a pragmatic middle ground that preserves AR-compatible serving while enabling parallel intra-block generation. Our study proceeds in four steps: (i) a **controlled, compute- and scale-matched comparison** revealing that AR is a more effective backbone for blockwise hybrids than masked diffusion objectives; (ii) a **scalable conversion recipe, SDAR**, validating that AR models spanning 1.7B to 30B parameters can be adapted into block diffusion models with minimal compute while preserving backbone capabilities; and (iii) a **systematic characterization of decoding dynamics**, which reveals a virtuous cycle where larger models enable more aggressive parallel decoding, achieving theoretical speedups over 5× and wall-clock speedups of 2.3× on H200 GPUs in latency-critical regimes; and (iv) an **investigation of local non-causal modeling capabilities**, showing that SDAR's local bidirectional attention overcomes causal bottlenecks in scientific domains (e.g., chemistry) and enables robust test-time scaling. We release the full model suite, the training framework, and our inference engines for further innovation in non-autoregressive generative paradigms.

1 Introduction

Large language models predominantly rely on the autoregressive (AR) paradigm, which factorizes joint probabilities via left-to-right next-token prediction (Bengio et al., 2003; Radford et al., 2019; Achiam et al., 2023; Guo et al., 2025). While AR offers dense supervision signals and supports deployment-critical features like KV caching, its strictly sequential nature imposes two fundamental limitations. First, the token-by-token generation creates a memory-bandwidth bottleneck, where decoding latency scales linearly with sequence length regardless of compute capacity. Second, the unidirectional commitment complicates tasks requiring global planning or bidirectional reasoning, as the model cannot revise earlier tokens based on future context (Hu et al., 2023; Bachmann and Nagarajan, 2024; Liu et al., 2025).

Masked diffusion language models (MDLMs) offer a compelling alternative by iteratively refining partially corrupted sequences, enabling bidirectional context utilization and parallel generation (Li et al., 2025; Yu et al., 2025; Gulrajani and Hashimoto, 2023). However, MDLMs face substantial friction in both training and inference. Unlike the dense supervision of AR, MDLMs employ ELBO masked objectives that provide sparse supervision, leading to poor sample efficiency (Lou et al., 2024; Nie et al., 2024; Arriola et al., 2025). Furthermore, the absence of KV caching and the requirement for iterative whole-sequence denoising often negate the theoretical speedups of parallel decoding, resulting in high wall-clock latency (Nie et al., 2025; Ye et al., 2025).

Blockwise diffusion (BD) models seek to reconcile these paradigms by decomposing generation hierarchically: a global AR structure preserves KV-cache compatibility, while a local diffusion process generates blocks of tokens in parallel (Han et al., 2022; Arriola et al., 2025; Fathi et al., 2025). While theoretically promising, current approaches necessitate expensive end-to-end pretraining or heavy fine-tuning, often doubling the compute cost relative to standard AR base-

*Equal contribution.
†Corresponding Authors.

Findings of the Association for Computational Linguistics: ACL 2026, pages 22058–22075

lines. To make this hybrid paradigm practically viable, we propose SDAR, a streamlined recipe that decouples backbone pretraining from diffusion adaptation. SDAR treats blockwise diffusion as a lightweight capability extension, converting pretrained AR models into block-parallel generators via a short continued-training phase rather than a full retraining.

Guided by this pragmatic perspective, our study unfolds progressively. We begin by establishing AR, rather than MDLM, as the optimal backbone for hybrid modeling via controlled initialization experiments. This foundation enables us to introduce the SDAR conversion recipe, verifying its scalability from 1.7B to 30B parameters. We characterize the decoding dynamics, analyzing how model size and entropy shape the parallel generation process that ultimately determines both theoretical efficiency and wall-clock latency. Finally, we examine this paradigm in structured scientific domains and in test-time scaling settings.

Our contributions are summarized as follows:

- **Establishing AR as the optimal backbone for blockwise diffusion.** We demonstrate that AR NTP objective is significantly more efficient than masked diffusion objectives, providing a superior and cheaper initialization for hybrid models.
- **SDAR: A scalable AR-to-BD conversion recipe.** We introduce a recipe that converts AR models (spanning 1.7B to 30B) into BD models with minimal compute. Crucially, this method retains the AR's capabilities while enabling parallel decoding.
- **Charting the quality-latency frontier.** We reveal a *virtuous cycle of scaling* where larger models enable faster parallel decoding, achieving up to **5**$\times$ **theoretical** and **2.3**$\times$ **wall-clock speedups** in latency-sensitive regimes.
- **Unlocking local bidirectional reasoning.** We show that SDAR's intra-block bidirectional attention overcomes the limitations of causal masking in structured scientific tasks and enables *robust test-time scaling*, where the model's inherent stochasticity significantly boosts reasoning accuracy.

We release the model suite, training framework, and inference engines to facilitate further research into non-autoregressive generation paradigm.

2 Preliminaries: Language Modeling Paradigms

We review autoregressive (AR), masked diffusion (MDLM), and block diffusion (BD) models. Given a vocabulary $\mathcal{V}$ of size V, let $x = (x^1, \ldots, x^L)$ denote a sequence of length L, where each token $x^\ell \in \mathcal{V}$ has a one-hot representation $\mathbf{x}^\ell \in \{0,1\}^V$. Let $\mathrm{Cat}(\cdot; p)$ be a categorical distribution parameterized by probabilities p. We abbreviate expectations as follows:

$$\mathbb{E}_{x_0,t}[\cdot] := \mathbb{E}_{x_0 \sim p_{\text{data}},\, t \sim \mathcal{U}(0,1),\, x_t \sim q(x_t|x_0)}[\cdot], \quad (1)$$

$$\mathbb{E}_{x,b,t}[\cdot] := \mathbb{E}_{x \sim p_{\text{data}},\, b \sim \mathcal{U}[1,B],\, t \sim \mathcal{U}(0,1)}[\cdot]. \quad (2)$$

where t is drawn uniformly from $(0, 1]$ and b indexes a block in BD models.

2.1 Autoregressive Models

AR models decompose the sequence probability via the chain rule. Training minimizes the negative log-likelihood (NLL) for next-token prediction:

$$\mathcal{L}_{\text{AR}}(\theta) = \mathbb{E}_x\Big[-\sum_{\ell=1}^{L} \log p_\theta(x^\ell \mid x^{<\ell}) \Big], \quad (3)$$

where $x^{<\ell} = (x^1, \ldots, x^{\ell-1})$ denotes the history. This factorization requires L sequential steps to sample a sequence, creating a latency bottleneck.

2.2 Masked Diffusion Language Models

MDLM learns to reverse a token-wise corruption process (Austin et al., 2021a). Letting M (the `[MASK]` token) denote the absorbing token with one-hot vector $\mathbf{m}$, the forward process is given by:

$$q(x_t^\ell \mid x_0^\ell) = \mathrm{Cat}\big(x_t^\ell;\, \alpha_t \mathbf{x}_0^\ell + (1-\alpha_t)\mathbf{m}\big), \quad (4)$$

where α_t is a predefined noise schedule.

The generative model p_θ is trained to reverse this corruption by minimizing the negative evidence lower bound (NELBO), yielding a reweighted cross-entropy loss:

$$\mathcal{L}_{\text{Diff}}(\theta) = \mathbb{E}_{x_0,t}\Big[-\frac{1}{t} \sum_{\ell \in \mathcal{M}_t} \log p_\theta(x_0^\ell \mid x_t) \Big], \quad (5)$$

where $\mathcal{M}_t$ denotes masked tokens and the weight $1/t$ derives from the linear schedule $\alpha_t = 1-t$ (Nie et al., 2025; Ye et al., 2025). Compared to the AR objective, Eq. (5) computes the loss solely over masked tokens and optimizes a loose NLL bound.

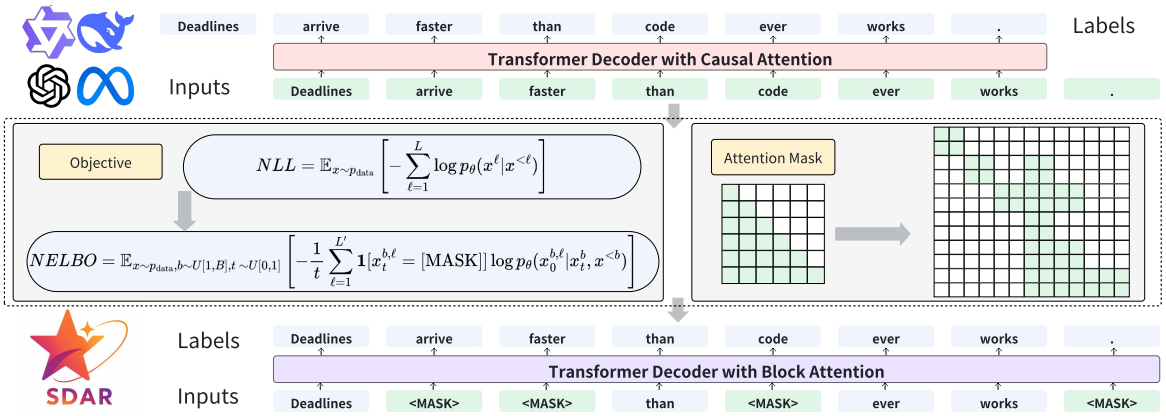

Figure 1: Overview of the SDAR paradigm. We first pre-train a standard AR model with the NTP objective. We then continue training under a blockwise causal attention mask and the blockwise diffusion loss in Eq. (7), converting the model into a blockwise diffusion language model using only 50B additional tokens.

2.3 Block Diffusion Language Models

BD models are introduced to enable variable-length generation and efficient KV caching (Han et al., 2022; Arriola et al., 2025). Partitioning x into B blocks $x^1, \ldots, x^B$ of length L', the model factorizes the sequence probability over blocks:

$$\log p_\theta(x) = \sum_{b=1}^{B} \log p_\theta(x^b \mid x^{<b}), \tag{6}$$

where $x^{<b}$ denotes preceding clean blocks. Each conditional $p_\theta(x^b \mid x^{<b})$ is modeled as a MDLM. Applying the masking strategy in Eq. (4) yields the expected block-wise conditional NELBO:

$$\mathcal{L}_{\text{BD}}(\theta) = \mathbb{E}_{x,b,t}\left[\frac{-1}{t}\sum_{\ell\in\mathcal{M}_t^b}\log p_\theta(x_0^{b,\ell} \mid x_t^b, x^{<b})\right], \tag{7}$$

where x_t^b is the noised version of block b, and $\mathcal{M}_t^b$ is the set of masked positions in block b at noise level t. The model learns to reconstruct each clean block x^b from its corrupted counterpart x_t^b, conditioned on the preceding context $x^{<b}$.

3 Method: The SDAR Paradigm

SDAR does not redesign BD; it makes BD practical on strong AR backbones via short continued training, leaving most compute to standard AR training. Figure 1 shows a schematic overview. Stage (i) trains a standard model θ_{AR} trained with NTP objective in Eq. (3). Stage (ii) converts it to a BD model θ_{SDAR} by modifying only the *attention mask* and *training objective*. Inference uses BD decoding atop the θ_{SDAR} (§3.2).

3.1 Conversion Training

From AR to BD Our recipe converts a pre-trained AR model θ_{AR} by systematically modifying both the data pipeline and the training objective. We first corrupt the input sequence x by masking a subset of tokens based on a uniformly sampled time t (Eq. (4)). As visualized in Figure 1, these perturbed sequences are concatenated with their clean counterparts, governed by a hybrid attention mask that enforces inter-block causality while permitting intra-block bidirectional attention. Under this modified setup, the model is continued-trained by switching the objective from standard NLL (Eq. (3)) to the diffusion NELBO (Eq. (7)).

Training Recipes Our pipeline entails a two-stage regime: Continued Pre-training (CPT) and Supervised Fine-tuning (SFT). CPT first adapts the AR model to the parallel decoding nature of diffusion models. Building on this aligned representation, SFT focuses on unlocking capabilities for complex tasks such as CoT reasoning. During SFT, we adopt a response-only corruption strategy, treating prompts as fixed conditions to maximize the likelihood of the response given the instruction.

3.2 Hierarchical Inference with SDAR

As illustrated in Figure 5, the generation process is *hierarchical*: autoregressive over blocks and non-autoregressive within each block. We begin with a standard prefill pass using a blockwise causal attention mask to populate the KV cache. Subsequently, blocks $x^1, x^2, \ldots$ are generated sequentially via MDLM decoding. Crucially, the current block attends to the KV cache of preceding blocks

but does not update the cache until it is finalized, preserving the blockwise causal attention pattern. To manage the intra-block decoding trajectory, we adopt confidence-based schedules to decide how many tokens to finalize at each denoising step:

- **Static schedule**: For a block of size L' and T steps, we finalize exactly $\lceil L'/T \rceil$ masked positions with the highest confidence (maximum probability) at each step, yielding a fixed T forward passes per block.
- **Dynamic schedule**: Given a confidence threshold $\tau \in (0, 1)$, we finalize masked positions whose probability exceeds τ, while enforcing a minimum number of finalized positions per step to guarantee progress.

Implementation details of our production-ready inference engine are provided in Appendix B.2.

4 Experimental Overview

We structure our experiments into four parts that progressively demonstrate the advantages, scalability, and unique capabilities of SDAR. More implementation details are provided in Appendix C.

Study I: Paradigm Foundations and Training Efficiency We establish the optimal backbone for hybrid modeling through a controlled 2B-parameter study. We train AR and MDLM model *from scratch* using an identical 1T-token corpus and a 4k context length. We apply a 100B-token annealing phase to generate blockwise variants (AR-BD and MDLM-BD) with a block size of 16, followed by SFT on a 4B-token dataset. This setup verifies that AR provides a more compute-efficient foundation for block diffusion than MDLM.

Study II: Scalability and Efficiency of SDAR Adaptation Building on above foundations, we validate the scalability of our recipe by applying it to the Qwen3 family across 1.7B, 4B, 8B, and 30B-A3B. For adaptation, we employ a computationally efficient pipeline: a 50B-token Continued Pretraining (CPT) phase (sampled from the annealing dataset) with a 4k context length and a block size of $L' = 4$, followed by the standard 4B-token SFT. This yields strictly matched AR-Chat and SDAR-Chat baselines to assess scaling laws.

Study III: Systematics of Blockwise Diffusion Scaling We characterize inference dynamics using the models from Study II. We perform a systematic sweep by expanding the block size during SFT with $L' \in \{4, 8, 16, 32, 64\}$ and evaluating both static and dynamic decoding strategies (with confidence thresholds $\tau \in \{0.80, 0.85, 0.90, 0.95\}$). We quantify algorithmic efficiency via Effective Tokens per Forward pass (TPF) and validate wall-clock throughput on our inference engine.

Study IV: Local Non-Causal Modeling Capabilities We investigate the modeling benefits of blockwise diffusion. We target scientific domains to test whether SDAR's local bidirectional attention effectively captures non-causal structures (e.g., molecular graphs). Starting from a science-enhanced Qwen3-30B-A3B checkpoint, we train matched AR-Sci and SDAR-Sci models on 50B science tokens, followed by long-CoT SFT. This validates structural modeling benefits and SDAR's compatibility with extended reasoning.

Evaluation Settings. In Studies I–III, we employ greedy decoding to ensure deterministic comparisons. In Study IV, we evaluate under both greedy and sampling decoding to assess diversity and test-time scaling. For SDAR, we report static decoding results by default. Detailed descriptions of the datasets and metrics used for each benchmark are provided in Appendix C.

5 Study I: Paradigm Foundations and Training Efficiency

In this section, we investigate the fundamental training efficiency of autoregressive (AR) and masked diffusion (MDLM) paradigms. By strictly controlling for model architecture, training data, and compute budget (FLOPs), we isolate the impact of the objective function on downstream performance. Furthermore, we evaluate the efficacy of these paradigms when serving as backbones for blockwise diffusion. Table 2 presents the results of our matched-compute 2B controlled study.

AR training is significantly more compute-efficient. The AR baseline consistently outperforms MDLM, establishing a substantial lead on logic-intensive tasks (e.g., MATH +17.3, HumanEval +20.8). This gap stems from intrinsic objective differences: unlike AR's dense supervision on exact likelihood, MDLM optimizes a loose variational bound (NELBO) subject to train-inference mismatch. Consequently, AR converts FLOPs into capability more effectively than the sparse signals provided by masked diffusion objectives.

Table 1: Performance comparison of AR and SDAR models. † indicates that the models are derived from the Qwen3 Base models via identical CPT and SFT. The best result within a given scale (e.g., 1.7B, 30B) is shown in **bold**.

Model	AR-Chat Model				SDAR-Chat Model			
scale	Qwen3-1.7B†	Qwen3-4B†	Qwen3-8B†	Qwen3-30BA3B†	SDAR-1.7B	SDAR-4B	SDAR-8B	SDAR-30BA3B
Reasoning, Knowledge & Instruction Following								
ARC-C	81.0	89.5	91.9	93.9	**85.4** (+4.4)	**90.5** (+1.0)	**91.9** (0.0)	93.2 (-0.7)
TriviaQA	45.0	57.9	68.2	66.5	42.6 (-2.4)	57.2 (-0.7)	**69.2** (+1.0)	**75.9** (+9.4)
MMLU	63.8	74.8	77.5	82.2	62.9 (-0.9)	**74.9** (+0.1)	**78.6** (+1.1)	**82.8** (+0.6)
MMLU-Pro	39.0	52.8	56.6	63.8	37.0 (-2.0)	50.9 (-1.9)	**56.9** (+0.3)	61.5 (-2.3)
GPQA-diamond	28.6	31.4	37.0	37.3	**29.8** (+1.2)	**33.0** (+1.6)	**40.2** (+3.2)	36.7 (-0.6)
IFEval	43.3	58.4	60.8	57.7	**43.4** (+0.1)	56.6 (-1.8)	**61.4** (+0.6)	**60.6** (+2.9)
Mathematics								
GSM8K	81.1	90.7	92.8	92.7	80.1 (-1.0)	89.9 (-0.8)	91.3 (-1.5)	91.4 (-1.3)
Math500	62.0	74.1	78.4	76.8	**63.2** (+1.2)	72.8 (-1.3)	**78.6** (+0.2)	**77.8** (+1.0)
MathBench	60.5	70.6	73.5	78.4	**63.6** (+3.1)	**74.7** (+4.1)	**76.9** (+3.4)	**79.3** (+0.9)
AIME-24	7.1	12.9	10.0	15.4	**10.0** (+2.9)	**10.0** (-2.9)	10.0 (0.0)	**16.7** (+1.3)
AIME-25	3.3	5.0	7.5	10.8	2.1 (-5.9)	**7.5** (+2.5)	**10.0** (+2.5)	10.8 (+0.0)
LMB-Hard	9.2	11.6	15.7	15.5	6.6 (-2.6)	6.9 (-4.7)	8.9 (-6.8)	13.7 (-1.8)
Code Generation								
HumanEval	65.9	73.4	80.7	84.8	61.6 (-4.3)	72.8 (-0.6)	78.7 (-2.0)	**87.2** (+2.4)
MBPP	61.9	67.1	75.1	75.1	61.1 (-0.8)	65.4 (-1.7)	72.0 (-3.1)	71.6 (-3.5)
HumanEval-X	47.0	60.9	63.5	63.5	45.2 (-1.8)	**62.3** (+1.4)	**64.9** (+1.4)	**66.3** (+2.8)
LCB-v6	8.6	10.3	20.5	23.4	5.7 (-2.9)	**13.1** (+2.8)	16.6 (-3.9)	21.7 (-1.7)
Language								
MMMLU-lite	36.7	44.4	55.2	52.9	40.9 (+4.2)	**50.7** (+6.3)	**55.3** (+0.1)	57.2 (+4.3)
CMMLU	60.7	72.6	76.2	82.0	60.2 (-0.5)	71.3 (-1.3)	75.7 (-0.5)	81.0 (-1.0)

Table 2: **Comparison of 2B models under matched compute.** AR significantly outperforms MDLM, while AR-BD largely retains performance, validating AR as the superior backbone for BD.

Benchmark	AR	MDLM	AR-BD	MDLM-BD
BBH	35.7	32.2	**35.9**	32.9
MMLU	48.7	47.0	**50.9**	47.5
MATH	**29.9**	12.6	26.8	23.3
GSM8K	61.8	57.9	59.4	**64.7**
HumanEval	**42.1**	21.3	40.0	39.0
MBPP	**44.4**	27.2	42.9	33.5
GPQA	26.3	26.3	28.2	**29.8**
Average	**41.3**	32.1	40.6	38.7

SDAR adaptation is nearly lossless. Converting the AR backbone to BD (AR-BD) preserves the base model's strong performance (40.6 vs. 41.3 avg), with gains on knowledge-heavy benchmarks offsetting minor regressions in symbolic reasoning. This confirms that pretraining with AR does not preclude diffusion-based decoding; rather, the generative mechanism can be effectively switched *post-hoc* with minimal adaptation cost.

AR is the optimal initialization for block diffusion. AR-BD consistently outperforms MDLM-BD, confirming AR's superior efficiency persists after conversion. Given the intrinsic inefficiency of native diffusion training, the optimal strategy is to decouple the paradigm—using AR for **capability acquisition** and adapting to diffusion *post hoc*.

6 Study II: Scalability and Efficiency of SDAR Adaptation

In this section, we validate the SDAR conversion recipe across model sizes ranging from 1.7B to 30B parameters. We aim to verify whether the blockwise diffusion adaptation preserves the general capabilities of the AR backbone as model size increases and whether this conversion remains computationally efficient.

Scaling with Lossless Adaptation. Table 1 presents a comparison of SDAR against its AR progenitors. We observe that SDAR exhibits robust scaling laws, preserving the performance benefits of the AR backbone across all model sizes. Crucially, the conversion is largely *lossless* or even *accretive*; at the 30B scale, SDAR achieves parity with or surpasses the baseline on the majority of benchmarks. Specifically, the model demonstrates improvements in instruction following (IFEval: +2.9) and key coding and math evaluations (e.g., HumanEval: +2.4, Math500: +1.0). This capability retention allows SDAR to significantly outperform native diffusion baselines (e.g., LLaDA) as detailed in Appendix Table 5, validating the efficiency of our blockwise adaptation recipe.

Efficiency of the Conversion Recipe. SDAR is exceptionally data-efficient. Unlike Dream (Ye et al., 2025) which requires 580B tokens, our ap-

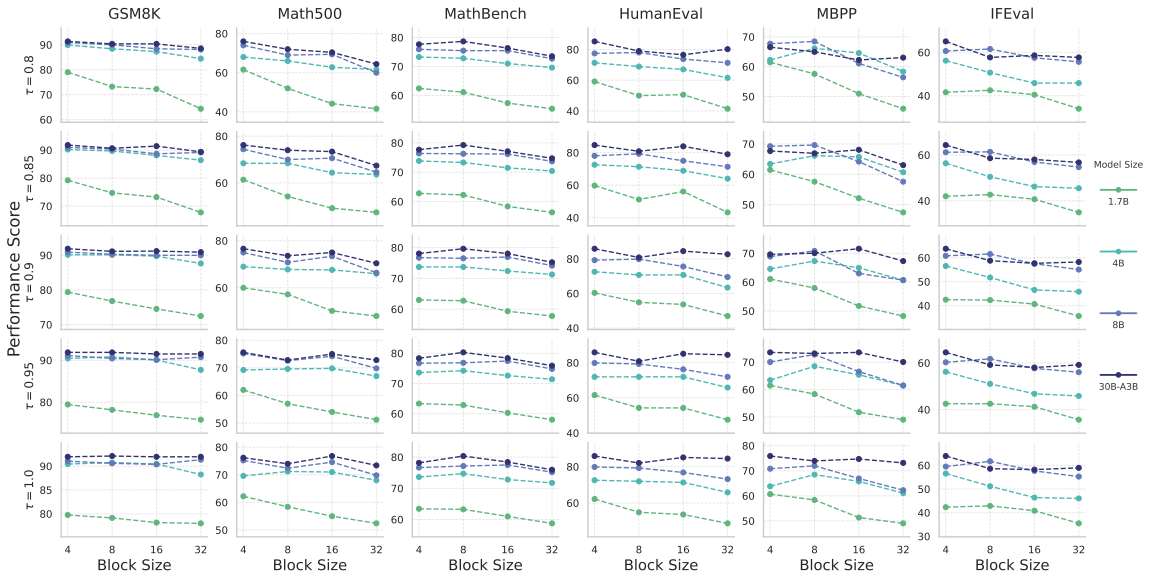

Figure 2: Performance on various benchmarks as a function of architectural block size (L') across different model sizes and various confidence thresholds (τ).

proach converges with only **50B tokens**. Using under 1% of the original pretraining compute, it unlocks blockwise generation without degrading backbone capabilities. This establishes SDAR as a viable pathway for equipping large-scale AR models with parallel decoding.

7 Study III: Systematics of Blockwise Diffusion Scaling

Having established the training efficiency of the AR backbone (Study I) and the SDAR scalability (Study II), we now characterize the inference dynamics. We systematically analyze the interplay between model size (N), block size (L'), and decoding policy (static vs. dynamic). Our goal is to chart the Pareto frontier between generation quality and latency, bridging theoretical algorithmic speedup with realized industrial throughput.

7.1 Performance Analysis

We first examine how model size influences robustness to parallelization. Figure 2 illustrates downstream performance as a function of block size $L' \in \{4, \ldots, 64\}$ across four model sizes.

The Virtuous Cycle of Scaling. We observe a critical phase transition governed by model capacity. Smaller models (1.7B, 4B) are highly sensitive to block size, exhibiting sharp performance degradation for $L' > 8$. In contrast, larger models (8B, 30B) demonstrate significant *robustness*, maintaining AR-parity performance even at $L' = 32$ or $L' = 64$. This implies a virtuous cycle: larger models possess stronger contextual modeling capabilities, allowing them to cohere longer sequences in parallel. Consequently, **larger models not only generate better text but effectively unlock larger speedups** by tolerating larger block sizes.

Non-Monotonic Performance and Inductive Bias. Performance does not monotonically decay with block size. On complex reasoning tasks (e.g., GSM8K, MathBench), intermediate block sizes ($L' \in \{8, 16\}$) occasionally outperform short blocks ($L' = 4$). Moderate blockwise generation introduces a beneficial inductive bias, encouraging the model to plan locally coherent segments rather than over-attending to immediate token-level probabilities. Furthermore, larger models tolerate more aggressive dynamic decoding thresholds ($\tau < 0.9$), whereas smaller models require conservative thresholds to avoid error propagation.

7.2 Algorithmic Efficiency Analysis

We quantify algorithmic efficiency using *Effective Tokens per Forward Pass* (TPF), representing the theoretical speedup upper bound per decoding step. Figure 3 plots TPF against block size across scales.

Scale Reduces Entropy, Increasing Speed. A counter-intuitive finding emerges: for a fixed block size and threshold, **larger models achieve higher TPF**. As illustrated in Figure 3, on datasets like Math500 and HumanEval, the TPF can exceed **5** with a block size of 64. Crucially, a distinct per-

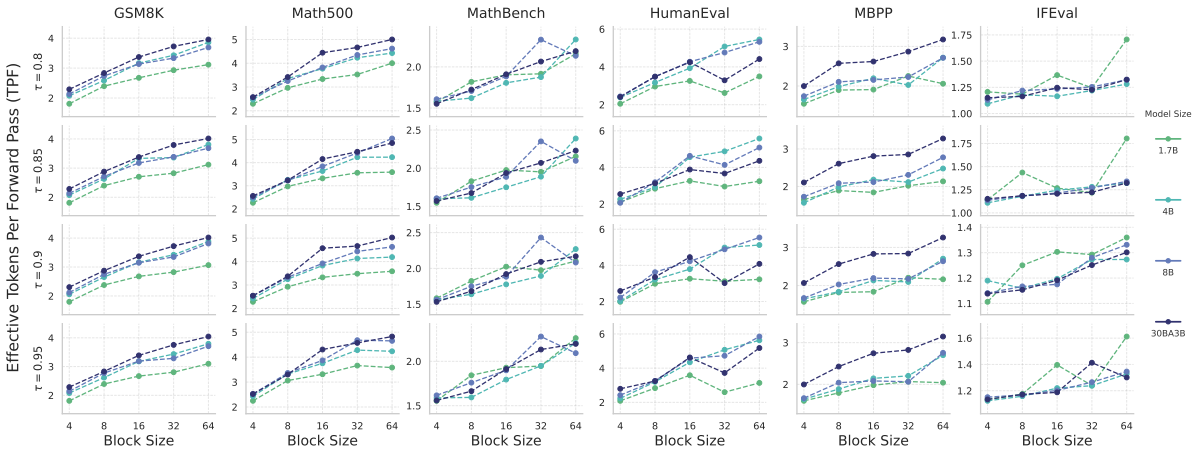

Figure 3: Algorithmic speedup, measured in Effective Tokens Per Forward Pass (TPF), as a function of architectural block size (L') across different model sizes and various confidence thresholds (τ).

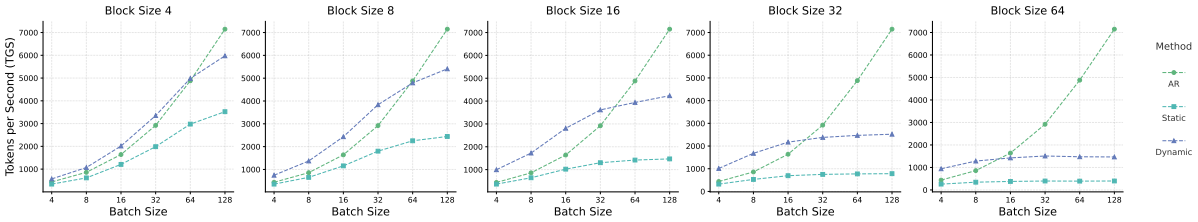

Figure 4: Throughput (tokens/s) vs. batch size for AR and SDAR (blocksize $L' \in \{4, 8, 16, 32, 64\}$) on SDAR-8B-Chat under static and dynamic regimes.

formance gap exists between scales; for instance, on GSM8K, the 30B model consistently achieves significantly higher TPF. Exceptionally, the 1.7B model shows higher TPF on IFEval, which we attribute to SFT data mismatch that induces repetitive degeneration and thus entropy collapse (inflated TPF). While standard AR inference speed typically degrades with model size due to memory bandwidth and compute costs, under the SDAR paradigm, larger models exhibit lower predictive entropy and higher confidence. This allows the dynamic decoding algorithm to accept more tokens per step. This suggests that techniques reducing output entropy, such as knowledge distillation or reinforcement learning, may serve as direct inference accelerators in this framework.

7.3 Wall-Clock Speed Analysis

We evaluate the wall-clock performance of the SDAR-8B model on an industrial inference engine powered by an NVIDIA H200 GPU. Figure 4 reports the Tokens Generated Per Second (TGS) across varying block sizes (L') and batch sizes, comparing our dynamic and static strategies against the standard AR baseline.

Latency-Sensitive Regime (Small Batch). SDAR is most beneficial in latency-sensitive settings (batch size $\leq$ 32) typical of interactive serving. In this memory-bound regime, AR decoding is constrained by memory bandwidth due to single-token generation, whereas SDAR amortizes weight loading by generating multiple tokens per forward pass. As shown in Figure 4, SDAR consistently outperforms AR at small batch sizes, achieving up to 2.3× speedup with $L' = 64$ and a batch size of 4.

Throughput-Bound Regime (Large Batch). Our implementation achieves peak throughputs, exceeding 6,000 TGS at high concurrency. However, as the batch size scales beyond 32, the system shifts from a memory-bound to a compute-bound regime. Due to the higher arithmetic intensity required to process L' candidates per step, SDAR reaches the GPU's compute saturation point earlier than the baseline. As a result, the speed advantage narrows, with AR eventually overtaking SDAR in pure throughput as the batch size approaches 128. **For a comprehensive profiling of the transition between memory- and compute-bound regimes**, please refer to Appendix D.2.

Table 3: **Comprehensive Performance of SDAR vs. AR.** Evaluated across scientific and reasoning benchmarks, SDAR achieves substantial gains in structured domains (e.g., Chemistry, GPQA) while maintaining parity or improving on general tasks. **Bold** indicates the best result.

Model	Structured Scientific Domains				General Reasoning					
	GPQA	Chembench	Physics	Protein	MMLU-Pro	AIME24	AIME25	LMB-H	LCB-v5	LCB-v6
AR-Sci	61.2	60.5	39.0	59.5	78.3	74.9	60.7	55.4	51.5	46.3
SDAR-Sci (G)	**66.7** (+5.5)	72.3(+11.8)	37.9 (-1.1)	**59.9** (+0.4)	78.1 (-0.2)	**76.7** (+1.8)	**60.0** (-0.7)	**60.7** (+5.3)	40.7 (-10.8)	42.3 (-4.0)
SDAR-Sci (S)	66.0 (+4.8)	**72.8**(+12.3)	**38.2** (-0.8)	59.6 (+0.1)	77.9 (-0.4)	73.4 (-1.5)	59.2 (-1.5)	58.7 (+3.3)	49.1 (-2.4)	**51.4** (+5.1)

Table 4: **Test-Time Scaling Dynamics.** Comparison of AR and SDAR variants (based on 30B-A3B). SDAR exhibits strong synergy with test-time scaling, generating diverse reasoning trajectories that lead to dramatic gains under Majority Voting and Pass@k ($k = 32$).

Model	GPQA	AIME24	AIME25	LMB-H
AR-Sci	61.2	74.9	60.7	55.4
SDAR-Sci (G)	66.7 (+5.5)	76.7 (+1.8)	60.0 (-0.7)	60.7 (+5.3)
SDAR-Sci (S)	66.0 (+4.8)	73.4 (-1.5)	59.2 (-1.5)	58.7 (+3.3)
+ Majority	68.2 (+7.0)	86.7(+11.8)	80.0(+19.3)	71.1(+15.7)
+ Pass@k	84.3(+23.1)	93.3(+18.4)	86.7(+26.0)	87.5(+32.1)

8 Study IV: Local Non-Causal Modeling Capabilities

Standard AR models are constrained by strict left-to-right causality. SDAR mitigates this via blockwise diffusion, introducing a **local non-causal** property where tokens within a block attend bidirectionally. By adapting a science-specialized backbone (Qwen3-30B-A3B-Sci), we isolate the impact of this relaxation on structured domains and analyze its synergy with test-time scaling.

8.1 Structured and General Capabilities

Scientific data (e.g., molecular graphs) involves non-linear dependencies challenging causal masking. We posit SDAR's local bidirectionality offers superior inductive bias for these structures. Table 3 reports performance across 10 benchmarks.

Local bidirectionality captures structural dependencies. SDAR shows distinct advantages in structure-rich tasks, achieving significant gains on ChemBench (+12.3) and the expert-level GPQA-diamond (+5.5). We attribute this to the **relaxation of token-level causality**: intra-block bidirectional attention allows the model to capture local topology (e.g., molecular bonds) without the constraints of sequential generation order.

Global sequential logic is preserved. Crucially, this structural enhancement does not compromise general reasoning. As shown in Table 3 (right), SDAR remains strong on logic-heavy benchmarks (MMLU-Pro, AIME) and outperforms AR on complex tasks like LiveMathBench-Hard (+5.3) and LiveCodeBench (+5.1). This indicates that modifying the local attention mechanism enhances structural modeling without disrupting the global chain-of-thought logic learned in pretraining.

8.2 Diversity and Test-Time Scaling

We further examine the interaction between diffusion-based stochasticity and test-time scaling (Majority Voting and Pass@k) in Table 4.

Diffusion noise maximizes ensemble efficiency. SDAR shows exceptional synergy with scaling strategies. Beyond strong single-sample performance, ensemble inference yields outsized gains: +19.3 on AIME25 and +15.7 on LiveMathBench-hard. Similarly, Pass@32 metrics reach 93.3 on AIME24. These results suggest that the non-causal generation produces a greater diversity of valid reasoning paths. Consequently, SDAR excels at "System 2" inference, effectively converting test-time compute into significant accuracy improvements.

9 Conclusion

In this work, we introduced SDAR to reconcile the training efficiency of autoregressive models with the parallel decoding of diffusion. Our study establishes that AR pretraining provides a more compute-efficient foundation than native masked diffusion. Leveraging this, we converted standard AR models up to 30B into blockwise diffusion generators with minimal adaptation cost (50B tokens), retaining strong capabilities while unlocking parallel generation. Crucially, we identified a *virtuous cycle of scaling*: larger models demonstrate greater robustness to parallelization, enabling 2.3× wall-clock speedups on H200 GPUs. Finally, we showed that SDAR's local bidirectionality effectively captures structured scientific data and maximizes test-time compute efficiency. We hope the released models serve as a foundation for future research in non-autoregressive generation.

10 Limitations

While SDAR offers a pragmatic path to efficient parallel generation, several limitations remain for future exploration:

Alignment Beyond Supervised Fine-Tuning. Our current study validates the synergy between SDAR and Long-CoT reasoning within a Supervised Fine-Tuning (SFT) framework. However, we have not yet explored alignment via Reinforcement Learning (RL) or preference optimization (e.g., DPO, PPO). Given that modern reasoning models rely heavily on RL to refine logic and safety, adapting these objectives to the blockwise diffusion loss landscape remains an open research direction.

Engineering Gap Between Theory and Practice. There remains a gap between our theoretical algorithmic speedup (over $5\times$) and realized wall-clock speedup ($2.3\times$). Although our custom inference engine, developed atop LMDeploy, achieves competitive throughput (e.g., 6,600 TGS on an 8B model), it represents a preliminary implementation. Specifically, our current pipeline performs a dedicated forward pass to populate the KV cache for a finalized block. A fully optimized implementation would integrate this update into the generation of the subsequent block. This would effectively hide the cache update overhead and further reduce the total number of forward passes.

Compute-Bound Throughput Constraints. As analyzed in §7.3, SDAR increases the arithmetic intensity of generation. While this is advantageous in memory-bound regimes (small batch sizes, latency-critical applications), it becomes a bottleneck in compute-bound regimes. At very high batch sizes (e.g., $>$ 128), the additional FLOPs required for iterative denoising outweigh the memory bandwidth savings, potentially resulting in lower pure throughput compared to highly optimized AR baselines.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 6250076080, the China Postdoctoral Science Foundation under Grant No. 2025M771537, and Shanghai Artificial Intelligence Laboratory.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AIME. 2025. AIME problems and solutions.

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

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Gregor Bachmann and Vaishnavh Nagarajan. 2024. The pitfalls of next-token prediction. In *International Conference on Machine Learning*, pages 2296–2318. PMLR.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Aadyot Bhatnagar, Sarthak Jain, Joel Beazer, Samuel C Curran, Alexander M Hoffnagle, Kyle Ching, Michael Martyn, Stephen Nayfach, Jeffrey A Ruffolo, and Ali Madani. 2025. Scaling unlocks broader generation and deeper functional understanding of proteins. *bioRxiv*, pages 2025–04.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman.

2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Nima Fathi, Torsten Scholak, and Pierre-André Noël. 2025. Unifying autoregressive and diffusion-based sequence generation. *arXiv preprint arXiv:2504.06416*.

Kaiyue Feng, Yilun Zhao, Yixin Liu, Tianyu Yang, Chen Zhao, John Sous, and Arman Cohan. 2025. Physics: Benchmarking foundation models on university-level physics problem solving. *arXiv preprint arXiv:2503.21821*.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, and 1 others. Scaling diffusion language models via adaptation from autoregressive models. In *The Thirteenth International Conference on Learning Representations*.

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*.

Ishaan Gulrajani and Tatsunori B Hashimoto. 2023. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36:16693–16715.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. 2022. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. 2022. Learning inverse folding from millions of predicted structures. *ICML*.

Edward J Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, Yoshua Bengio, and Nikolay Malkin. 2023. Amortizing intractable inference in large language models. *arXiv preprint arXiv:2310.04363*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023. Cmmlu: Measuring massive multitask language understanding in chinese. *Preprint*, arXiv:2306.09212.

Tianyi Li, Mingda Chen, Bowei Guo, and Zhiqiang Shen. 2025. A survey on diffusion language models. *arXiv preprint arXiv:2508.10875*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, and 1 others. 2022. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. 2024a. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6884–6915.

Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. 2024b. Are your llms capable of stable reasoning? *arXiv preprint arXiv:2412.13147*.

Ke Liu, Shuaike Shen, and Hao Chen. 2025. From sentences to sequences: Rethinking languages in biological system. *arXiv preprint arXiv:2507.00953*.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios

of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32819–32848.

Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos Jr, Caiming Xiong, Zachary Z Sun, Richard Socher, and 1 others. 2023. Large language models generate functional protein sequences across diverse families. *Nature biotechnology*, 41(8):1099–1106.

Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alexander Rives. 2021. Language models enable zero-shot prediction of the effects of mutations on protein function. *bioRxiv*.

Adrian Mirza, Nawaf Alampara, Sreekanth Kunchapu, Martiño Ríos-García, Benedict Emoekabu, Aswanth Krishnan, Tanya Gupta, Mara Schilling-Wilhelmi, Macjonathan Okereke, Anagha Aneesh, Amir Mohammad Elahi, Mehrdad Asgari, Juliane Eberhardt, Hani M. Elbeheiry, María Victoria Gil, Maximilian Greiner, Caroline T. Holick, Christina Glaubitz, Tim Hoffmann, and 16 others. 2024. Are large language models superhuman chemists? *arXiv preprint arXiv: 2404.01475*.

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

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

Erik Nijkamp, Jeffrey A Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. 2023. Progen2: exploring the boundaries of protein language models. *Cell systems*, 14(11):968–978.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. Improving language understanding by generative pre-training.

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*.

Yiqing Shen, Zan Chen, Michail Mamalakis, Luhan He, Haiyang Xia, Tianbin Li, Yanzhou Su, Junjun He, and Yu Guang Wang. 2024. A fine-tuning dataset and benchmark for large language models for protein understanding. In *2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2390–2395. IEEE.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and 1 others. 2022. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. 2024a. Dplm-2: A multimodal diffusion protein language model. *arXiv preprint arXiv:2410.13782*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2024. Beyond autoregression: Discrete diffusion for complex reasoning and planning. *arXiv preprint arXiv:2410.14157*.

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

Runpeng Yu, Qi Li, and Xinchao Wang. 2025. Discrete diffusion in large language and multimodal models: A survey. *arXiv preprint arXiv:2506.13759*.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. 2023. Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5673–5684.

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 Related Work

Our work, SDAR, resides at the intersection of autoregressive and diffusion-based language modeling, aiming to synthesize their respective strengths while mitigating their inherent weaknesses. We structure our review by first examining these two dominant paradigms, then analyzing prior attempts to unify them, and finally contrasting our approach with orthogonal methods for inference acceleration.

A.1 The Autoregressive Paradigm: Dominance and Inherent Constraints

AR models, which factorize the joint probability of a sequence into a left-to-right product of conditionals, represent the de facto standard in large-scale language modeling (Radford et al., 2019; Achiam et al., 2023; Grattafiori et al., 2024; Yang et al., 2025). The success of this paradigm is rooted in its strict causal inductive bias, which aligns naturally with the sequential structure of human language (Bengio et al., 2003; Vaswani et al., 2017; Radford et al., 2018). This alignment, combined with the stable and efficient cross-entropy training objective, has enabled unprecedented scaling and performance on a wide array of general-purpose tasks (Kaplan et al., 2020; Brown et al., 2020).

However, the very causal bias that underpins the AR model's success becomes its **Achilles' heel** for tasks demanding non-local or holistic reasoning. This fundamental mismatch has been shown to impede performance on problems where solutions depend on global constraints, such as mathematical puzzles (e.g., Sudoku, 24-point game), Boolean satisfiability, and long-term planning (Hu et al., 2023; Bachmann and Nagarajan, 2024; Ye et al., 2024). The model's rigid, token-by-token generation process struggles to reason about future dependencies or revise earlier decisions in light of global information. For instance, recent studies demonstrate that masked generative diffusion models can dramatically outperform strong AR baselines on combinatorial problems like Sudoku (e.g., 91.5 vs. 45.8 accuracy (Ye et al., 2024)), underscoring a clear paradigm-level advantage.

This limitation also manifests in scientific domains. Modeling chemical formulas (e.g., SMILES), for instance, requires bidirectional context. Similarly, modeling protein sequences to resolve structural motifs and functional domains is inherently at odds with the unidirectional autoregressive (AR) process, a foundational challenge for generative models like the ProGen series (Madani et al., 2023; Nijkamp et al., 2023; Bhatnagar et al., 2025). This has motivated a shift towards architectures that can leverage global, bidirectional context. Examples include powerful representation learners like the ESM series (Meier et al., 2021; Hsu et al., 2022; Lin et al., 2022) and non-autoregressive generators such as discrete diffusion models dplm2 (Wang et al., 2024a). As argued by Liu et al. (2025), the standard sequential paradigm is fundamentally suboptimal for biological sequences due to their critical long-range dependencies. Our own experiments affirm this hypothesis: by locally relaxing this causal constraint, SDAR achieves a 72.8 accuracy on the challenging Chembench benchmark (Mirza et al., 2024), significantly outperforming its AR counterpart (60.5).

A.2 Diffusion Models: Holistic Modeling at a Prohibitive Cost

As an alternative, discrete diffusion models circumvent the strict causal constraints of AR models by treating sequence generation as a holistic denoising process, which learns to reverse a gradual corruption of the data (Strudel et al., 2022; Han et al., 2022; Gulrajani and Hashimoto, 2023; Shi et al., 2024; Lou et al., 2024). This architectural freedom offers two profound advantages: (1) a natural capacity for parallel decoding, directly addressing the latency bottleneck of AR inference, and (2) a flexible, bidirectional inductive bias that is better suited for the aforementioned non-local reasoning tasks.

Despite this promise, the practical application of diffusion language models has been severely hampered by their prohibitive training cost (Nie et al., 2024; Gulrajani and Hashimoto, 2023). This inefficiency stems from two primary sources. First, the training objectives, such as the Evidence Lower Bound (ELBO), are often more difficult to optimize and converge slower than the standard cross-entropy loss used in AR models (Nie et al., 2024; Arriola et al., 2025). Second, the learning task of reconstructing a fully structured sequence from varying levels of noise is information-theoretically

more challenging than next-token prediction. Empirical studies quantify this gap, showing that masked discrete diffusion models can require up to $16\times$ more FLOPs to match the validation NLL of an AR equivalent (Nie et al., 2024), a disparity that widens to as much as $64\times$ for continuous diffusion models (Gulrajani and Hashimoto, 2023).

A.3 Hybrid and Conversion Models: Bridging the Gap

Recognizing the complementary nature of AR and diffusion models, several lines of research have explored hybrid architectures.

Block-wise Hybrid Models. A prominent approach fuses the paradigms by employing an autoregressive structure at a global, inter-block level while using a parallel, diffusion-based mechanism for intra-block generation (Arriola et al., 2025; Han et al., 2022; Fathi et al., 2025). This strategy elegantly preserves macroscopic causal flow, enabling variable-length generation and KV-caching. However, these models have traditionally adopted a monolithic training regime, where the AR and diffusion components are trained jointly from scratch. This subjects them to the full training inefficiency of the diffusion objective, compounded by the complexity of a hybrid loss function, presenting a significant barrier to scaling.

AR-to-Diffusion Conversion Models. Another strategy leverage the efficiency of AR pre-training by first training a standard LLM and then adapting it to a non-autoregressive or diffusion-based objective. Works like DiffuLLaMA (Gong et al.) and Dream 7B (Ye et al., 2025) follow this path. While conceptually appealing, this conversion has proven to be computationally expensive (e.g., requiring 580B tokens for adaptation in Dream 7B) and can result in notable performance degradation compared to the original AR model. In stark contrast, SDAR's **decoupled training and inference paradigm** and block-level adaptation requires a minimal fine-tuning budget (e.g., 50B tokens) to unlock parallel decoding, while fully preserving, and on specialized tasks enhancing, the performance of the foundational AR model.

B Infrastructures

B.1 Training Infrastructure

The training of the SDAR model requires specialized attention masks and training objectives that differ from standard autoregressive language models. As a result, widely adopted kernels such as `FlashAttention` for attention computation and fused cross-entropy kernels from projects like `Liger-Kernel` cannot be directly applied.

To address this, we adopt `FlexAttention`[1], which achieves significant speedups compared to the scaled dot-product attention provided in PyTorch. Besides, we modify the fused cross-entropy loss to support element-wise division of the per-token loss terms, matching the diffusion ELBO form. This design avoids fully materializing the logits tensor—a highly memory-intensive step—thereby reducing GPU memory usage and improving overall efficiency.

In terms of efficiency, the raw throughput, measured in tokens processed per GPU per second, is comparable to conventional autoregressive model training. However, due to SDAR's training paradigm, which requires processing both clean and noised versions of each sample, the effective training speed is approximately halved relative to standard AR models.

B.2 Inference Infrastructure

Unlike many diffusion-based models that rely on approximate or specialized cache mechanisms, SDAR maintains a mathematically identical key-value (KV) cache to that of standard autoregressive models. This compatibility allows us to leverage established cache management techniques, including PagedAttention (Kwon et al., 2023) as implemented in vLLM. Figure 5 summarizes the SDAR inference workflow that our system targets, highlighting the block-wise generation pattern and KV-cache reuse behavior.

We design a custom attention kernel to handle the specialized attention mask required during the prefill stage. For the decoding stage, we integrate PagedAttention, ensuring inference efficiency competitive with state-of-the-art autoregressive inference engines. Our implementation extends autoregressive infrastructure with block-oriented optimizations, employing block-aligned memory allocation that provisions $n/(n+1)$ blocks to support n forward passes. The inference pipeline operates through iterative resource allocation, batched block-level inference, and state updates. Key optimizations include unified treatment of denoising and cache-filling operations dur-

[1] https://pytorch.org/blog/flexattention/

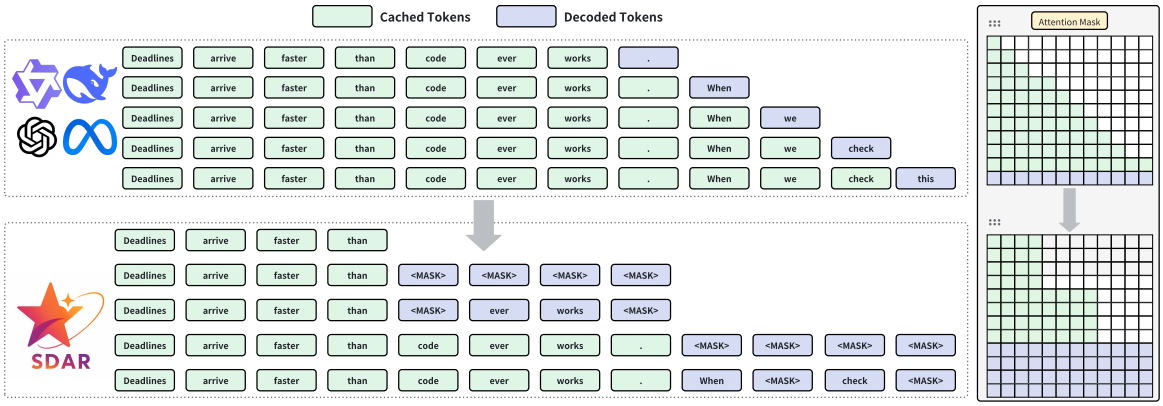

Figure 5: Contrasting inference paradigms between AR and SDAR. SDAR adopts a block-wise causal attention mechanism to enable inter-block causal autoregression and intra-block parallel diffusion. The KV cache from previously generated blocks is reused, while the block currently being decoded does not store KV cache.

ing forward passes, fully batched unmasking operations to avoid device synchronization, and deferred stop condition evaluation to handle mid-block termination sequences.

Beyond efficiency, we develop a lightweight, modular inference-engine framework built on top of the open-source `Nano-vLLM`[2]. It is designed for ease of modification and educational use, facilitating rapid prototyping of new sampling strategies and RL-based algorithms. We also validate SDAR on an industrial-grade inference engine that supports prefix-based KV-cache sharing across requests, reducing both compute and memory overhead for interactive generation.

C Experiment Setup

C.1 Experiment Setup for Study I

Training Setup. We pretrain two 2B-parameter models from scratch on an identical 1T-token corpus (general web, STEM, and code), using the same transformer architecture, tokenizer, optimizer, and context length (4096 with packed sequences). The AR model uses variable-length causal attention; the MDLM uses full bidirectional attention, which is standard and effective for diffusion-style training(Nie et al., 2025).

After pretraining, we perform an identical 100B-token annealing training phase to obtain four Base models:

1. **AR-2B-Base:** The pretrained AR model is continually trained with its original autoregressive objective.
2. **SDAR-2B-Base:** The pretrained AR model is converted to the Block Diffusion paradigm, with a block size of 16. This is our proposed SDAR paradigm.
3. **MDLM-2B-Base:** The pretrained MDLM is continually trained with its original masked diffusion objective.
4. **MDLM-BD-2B-Base:** The pretrained MDLM is converted to the blockwise diffusion, with a block size of 16.

Each Base model then undergoes SFT on the same 4B-token dataset, using the same strategy as in its annealing phase, producing four chat variants: AR, AR-BD, MDLM, and MDLM-BD.

For MDLM training, we use packed sequences with a global non-causal attention mask during CPT. During SFT, to prevent attention leakage across packed sequences, we switch to a **variable-length non-causal attention** that restricts attention within each packed sequence. Similarly, for BD training, we use packed sequences with a global block-wise attention mask during CPT. During SFT, we adopt a **sequence-aware block-wise attention mask** to mimic variable-length attention behavior.

Evaluation Setup. We evaluate the four variants on downstream tasks that probe reasoning, mathematics, and code:

- **Reasoning & Knowledge**: BBH (Suzgun et al., 2022) (3-shot), MMLU (Hendrycks et al., 2021) (5-shot), GPQA-Diamond (Rein et al., 2024) (0-shot).

[2] https://github.com/GeeeekExplorer/nano-vllm

- **Mathematics**: MATH (Lightman et al., 2023) (4-shot, CoT), GSM8K (Cobbe et al., 2021) (4-shot, CoT).
- **Coding Tasks**: HumanEval (Chen et al., 2021) (0-shot), MBPP (Austin et al., 2021b) (3-shot).

All four variants are evaluated using greedy decoding. The BD variants (SDAR and MDLM-BD) use static decoding with block length $L' = 16$ and 16 denoising steps. The MDLM-2B-Chat follows LLaDA (Nie et al., 2025) protocol, using static decoding and semi-autoregressive remasking strategy with hyper-parameters set according to the original publication.

C.2 Experiment Setup for Study II

Training Setup. We start from the Qwen3 base models at 1.7B, 4B, 8B, and 30B-A3B scales and adapt them with a two-stage recipe. We run 50B-token CPT on a subset sampled from the 1T-token corpus in Appendix C.1 (4096-token length, packed sequences), followed by SFT on the same 4B-token instruction-following dataset. We use the same attention-mask settings for CPT and SFT as those described in Appendix C.1.

Evaluation Setup. We conduct a comprehensive evaluation of our models on a suite of downstream benchmarks designed to assess key capabilities in reasoning, mathematics, code generation, and instruction following. The evaluation datasets are organized as follows:

- **Reasoning & Knowledge:** , MMMLU-lite (Hendrycks et al., 2021) (0-shot), TriviaQA (Joshi et al., 2017) (1-shot), MMLU (Hendrycks et al., 2021) (5-shot), CMMLU (Li et al., 2023) (0-shot), MMLU-Pro (Wang et al., 2024b) (0-shot, CoT), and GPQA-diamond (Rein et al., 2024) (0-shot).
- **Mathematics:** GSM8K (Cobbe et al., 2021) (0-shot, CoT), MATH-500 (Lightman et al., 2023) (0-shot, CoT), MathBench (0-shot, CoT) (Liu et al., 2024a), and the challenging competition-level benchmarks AIME-2024, AIME-2025 (AIME, 2025) and LiveMathBench-Hard (LMB-Hard) (Liu et al., 2024b).
- **Code Generation:** HumanEval (Chen et al., 2021), HumanEval-X (Zheng et al., 2023), MBPP (Austin et al., 2021b), and LiveCodeBench (LCB) (Jain et al., 2024), all evaluated in a zero-shot setting.
- **Instruction Following:** IFEval (Zhou et al., 2023) (0-shot).

For our evaluation protocol, we employ greedy static decoding for the SDAR-Chat models, with both the block length and the number of denoising steps set to 4. All AR models are evaluated under standard greedy decoding.

C.3 Experiment Setup for Study III

Training Setup. Starting from the SDAR base models ($L' = 4$) in Appendix C.2, we run SFT on the same 4B-token dataset while varying the block length $\{4, 8, 16, 32, 64\}$. We keep the model architecture and initialization fixed, and only modify the blockwise attention mask accordingly. All SFT runs use packed sequences with a 4K length.

Evaluation Setup. We use greedy decoding throughout. Additionally, we investigate two distinct decoding strategies:

- **Static Decoding:** The generation of each block involves a full sequence of L' iterative denoising steps, where L' is the model's architectural block size. This exhaustive refinement process maximizes output quality, thereby establishing the *performance ceiling* for the given architecture. All dynamic, early-exit strategies are thus evaluated as trade-offs against this quality benchmark.
- **Dynamic Decoding:** To enable a finer-grained trade-off, this mode dynamically adjusts the number of tokens generated per step, k (where $1 \leq k \leq L'$), based on a confidence threshold τ. The number of denoising steps equals the chosen block size for that step. We evaluate a range of thresholds, $\tau \in \{0.80, 0.85, 0.90, 0.95\}$, to explore the spectrum between aggressive and conservative parallel generation.

C.4 Experiment Setup for Study IV

Training Setup. The development of our proposed science-oriented models follows a three-stage pipeline:

1. **Extensive Domain Pre-training (AR Objective):** We continue pretrain Qwen3-30B-A3B

Table 5: Performance comparison of SDAR, AR, and Diffusion models. The best result in each column is highlighted in **bold**.

Scale	Model	MMLU	GSM8K	Math500	HumanEval	MBPP	IFEval
AR Models							
1.7B	AR-Chat	63.8	81.1	62.0	65.9	61.9	43.3
	Qwen3-Base	62.6	75.4	43.5	–	55.4	–
30BA3B	AR-Chat	82.2	**92.7**	76.8	84.8	**75.1**	57.7
	Qwen3-Base	81.4	91.8	59.0	–	74.4	–
Diffusion Models							
8B	LLaDA	65.9	78.6	37.3	47.6	34.2	59.9
7B	Dream	69.5	81.0	38.7	55.5	58.8	**62.5**
SDAR Models							
1.7B	SDAR-Chat	62.9	80.1	63.2	61.6	61.1	43.4
4B		74.9	89.9	72.8	72.0	65.4	56.6
8B		78.6	91.3	**78.6**	78.7	72.0	61.4
30BA3B		**82.8**	91.4	77.8	**87.2**	71.6	60.6

on a 1T-token corpus in two phases: 500B tokens from a general-domain mixture with scientific data, followed by 500B tokens of high-quality scientific annealing data. This yields a science-oriented AR base checkpoint.

2. **Paradigm Conversion to SDAR:** Starting from the science-oriented AR checkpoint, we continue training under the blockwise diffusion objective on 50B tokens sampled from the 500B annealing corpus, using a sequence length of 20480 tokens, yielding SDAR-30B-A3B-Sci-Base.

3. **SFT for Reasoning:** We fine-tune both the science-aware AR base checkpoint and SDAR-30B-A3B-Sci-Base on the same 0.5B-token collection of high-quality long-CoT instruction data, using 20480 context length, yielding AR-30B-A3B-Sci and SDAR-30B-A3B-Sci, respectively.

Evaluation Setup. We evaluate our science-oriented models on a curated suite of frontier benchmarks covering complex reasoning, mathematics, specialized scientific domains, and code generation. The evaluation datasets are organized as follows:

- **Expert Reasoning & Knowledge:** MMLU-Pro (Wang et al., 2024b) and GPQA-diamond (Rein et al., 2024).
- **Competition-Level Mathematics:** AIME-2024, AIME-2025 (AIME, 2025), and LiveMathBench-hard (LMB-hard) (Liu et al., 2024b).
- **Specialized Scientific Domains:** Chembench (Mirza et al., 2024), PHYSICS (Feng et al., 2025), and ProteinLMBench (Shen et al., 2024).
- **Code Generation:** LiveCodeBench-v5 and LiveCodeBench-v6 (Jain et al., 2024).

For our SDAR-Sci model, we use a fixed block length of 4 and static decoding strategy. We report results under two sampling strategies: **(G)** greedy decoding and **(S)** sampling-based decoding with `temperature` = 1.0, `top_p` = 1.0, and `top_k` = 0. Meanwhile, AR-Sci is evaluated using a recommended sampling-based approach with `temperature` = 0.6, `top_p` = 0.95, and `top_k` = 20 (Yang et al., 2025). For all sampling-based evaluations, we report the mean performance over multiple runs on key benchmarks: 8 runs for GPQA-diamond, and 32 runs for AIME-2024, AIME-2025, and LiveMathBench-hard.

D Additional Results and Analysis

D.1 Study II: Comparison with SOTA Baselines

In Table 5, we further contextualize these findings by comparing SDAR with the original Qwen3-Base models and other prominent non-autoregressive architectures like LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025). Our

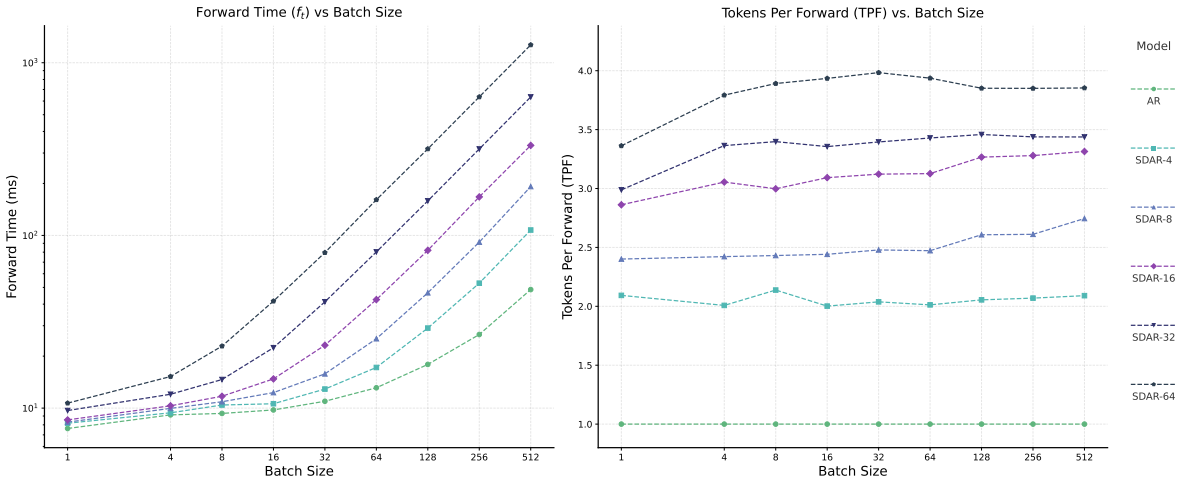

Figure 6: **Left:** Single forward pass time (t_f) as a function of batch size for AR and SDAR models. The transition from a flat curve to a linear slope marks the shift from memory-bound to compute-bound regimes. **Right:** Realized Tokens Per Forward Pass (TPF) calculated from actual inference throughput (Eq. 11). The consistency of TPF across batch sizes confirms that algorithmic efficiency remains stable even when hardware saturates.

SDAR models substantially outperform these prior diffusion-based approaches across all key benchmarks. For example, our SDAR-8B model achieves 78.6% on MMLU, a significant leap over LLaDA-8B (65.9%) and Dream-7B (69.5%). This substantial performance gap highlights that SDAR represents a major advancement for non-autoregressive generation, making it truly competitive with top-tier AR models for the first time. Furthermore, the SDAR-Chat models demonstrate enhanced instruction-following capabilities over their AR-Chat counterparts (e.g., a +2.9% gain on IFEval at the 30B scale), suggesting the block-wise approach may better reinforce alignment. Taken together, these results validate that our two-stage training strategy provides a robust foundation and positions SDAR as a leading paradigm, not only advancing beyond previous diffusion models but also establishing itself as a powerful and capable competitor to autoregressive generation.

D.2 Study III: Deployment Efficiency Analysis

In the main text (Section 7.3), we presented the wall-clock throughput of SDAR. To further understand the underlying hardware behaviors and the transition between memory-bound and compute-bound regimes, we provide profiling of the inference process on the NVIDIA H200 GPU.

D.2.1 Analytical Framework

To decouple algorithmic efficiency from hardware constraints, we formalize the relationship between throughput and execution time. Let α denote the batch size, β the block size (L'), and γ the average number of forward passes per block generation.

For standard Autoregressive (AR) decoding, the Tokens Generated Per Second (TGS) is governed by the time per forward pass (t_f):

$$\text{TGS}_{\text{AR}} = \frac{\alpha}{t_f} \tag{8}$$

$$t_f = \frac{\alpha}{\text{TGS}_{\text{AR}}} \tag{9}$$

For our SDAR implementation on the inference engine, which requires one additional forward pass for KV cache updates per generation step, the throughput is formulated as:

$$\text{TGS}_{\text{SDAR}} = \frac{\alpha\beta}{(\gamma+1)t_f} \tag{10}$$

From this, we derive the **Realized Tokens Per Forward Pass (Realized TPF)**. This metric allows us to verify whether the theoretical algorithmic speedup is achieved in practice, independent of hardware saturation:

$$\text{Realized TPF} = \frac{\beta}{\gamma} = \left(\frac{\alpha}{\text{TGS}_{\text{SDAR}} \times t_f} - \frac{1}{\beta}\right)^{-1} \tag{11}$$

D.2.2 Profiling Hardware Regimes

Figure 6 presents the dual analysis of execution time and algorithmic efficiency.

Transition from Memory-Bound to Compute-Bound. The left panel of Figure 6 plots t_f against batch size, revealing two distinct operating regimes:

- **Memory-Bound Regime (Small Batch):** At low concurrency ($\alpha \leq 32$), t_f remains relatively constant. In this regime, latency is dominated by the memory bandwidth required to load model weights rather than computation. SDAR excels here by amortizing the expensive weight loading over multiple generated tokens (β) per forward pass, leading to significant speedups.
- **Compute-Bound Regime (Large Batch):** As the batch size scales, t_f exhibits linear growth, indicating that the GPU's arithmetic units are fully saturated. Since SDAR processes β candidate tokens per step, it entails a higher arithmetic intensity than AR. Consequently, SDAR reaches this compute saturation point at smaller batch sizes compared to the baseline.

This explains the phenomenon observed in the main text: while SDAR offers up to $2.3\times$ speedup in latency-sensitive scenarios, the advantage narrows in throughput-oriented scenarios as the system becomes bound by raw FLOPS.

Validation of Algorithmic Efficiency. The right panel of Figure 6 displays the Realized TPF derived from the actual end-to-end throughput using Eq. 11. Crucially, this metric remains consistent across varying batch sizes and aligns closely with the empirical TPF measurements observed in our generation experiments (Figure 3). This consistency confirms that the diminishing wall-clock speedup at large batch sizes is *not* due to a degradation in SDAR's acceptance rate or algorithmic failure, but is strictly a consequence of hardware compute limits.

Summary. These findings suggest that SDAR is particularly well-suited for interactive, low-latency applications (e.g., real-time chatbots) where batch sizes are typically small. For high-throughput offline batch processing, the trade-off shifts towards compute capacity. Future optimizations, such as kernel fusion to eliminate the separate KV-cache overhead, could further reduce the arithmetic overhead and extend SDAR's advantage into higher batch regimes.