ONE BLOCK AHEAD: PROMOTING SAMPLER STATE IN DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Block-diffusion language models denoise a whole block of token positions in parallel so each forward pass can emit many tokens. However, blocks are still denoised in strict sequence, a cost that within-block acceleration leaves untouched. Here, we investigate decoding blocks concurrently and propose One Block Ahead (OBA), a training-free decoding algorithm that denoises the next block alongside the current one. We demonstrate that for uniform-state diffusion LMs, where a block's denoising progress is not visible in its intermediate tokens, concurrent decoding is advantageous if the next block's tokens are promoted together with the corresponding sampler state, including the self-conditioning signal and the denoising time. We also derive a break-even criterion that relates the benefit of OBA to the relative cost of a paired-block forward pass. On DiffusionGemma, a state-of-the-art diffusion LM, OBA increases the tokens per forward pass by 42.3% (from 15.64 to 22.25) and the single-request denoising throughput by 16.3%, averaged over ten benchmarks, with a small accuracy regression.

1 INTRODUCTION

Diffusion language models generate text by iteratively denoising a whole sequence of token positions, i.e., a block, so a single forward pass can decode many positions at once (Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025; Fu et al., 2026). That decoding parallelism is the principal advantage of diffusion over autoregressive decoding, generating one token at a time. However, there is typically a tradeoff with quality: the more positions a step accepts, the more it disregards the dependencies among them, so accepting tokens aggressively improves the latency at the cost of quality (Ben-Hamu et al., 2025; Fu et al., 2026). How far a decoder can push that quality–throughput frontier is what determines whether diffusion is competitive with autoregression on a single request.

Among recent diffusion language models, DiffusionGemma offers a particularly strong quality–throughput trade-off, reporting around 20 tokens per forward pass (TPF) and roughly 1,500 output tokens per second (TPS) on a single H100 GPU while maintaining strong benchmark performance (DiffusionGemma Team, 2026). Its efficiency comes from several design choices. A block-autoregressive architecture supports KV caching and variable-length generation (Arriola et al., 2025); uniform-state diffusion (Hoogeboom et al., 2021; Schiff et al., 2025) keeps every position revisable until its block is committed, allowing aggressive parallel updates without the irreversible early commitments of masked diffusion decoding; and entropy-bounded sampling (Ben-Hamu et al., 2025), adaptive stopping, and sampler distillation reduce the required denoising steps, with self-conditioning carrying predictions from one step to the next. However, its decoding remains sequential across blocks, which leaves multi-block parallelism unexplored: can decoding successive blocks concurrently further accelerate a model that already achieves such high within-block efficiency?

Recent work has demonstrated the potential of multi-block decoding through distillation, post-training, and concurrent decoding schedules (Wang et al., 2026; Jin et al., 2026; Tian et al., 2026). However, whether these gains extend to DiffusionGemma's combination of native 256-token blocks, high decoding efficiency, and strong model quality remains unestablished. For example, Multi-Block Diffusion (Jin et al., 2026) and FlowBlock (Tian et al., 2026) report their main results with 32-token blocks and baselines averaging 3.47 and 4.62 TPF, respectively. Extending concurrency to DiffusionGemma poses both an algorithmic and a hardware utilization challenge. Its self-conditioning signal and denoising time carry progress that must survive the transition between blocks. Mean-

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

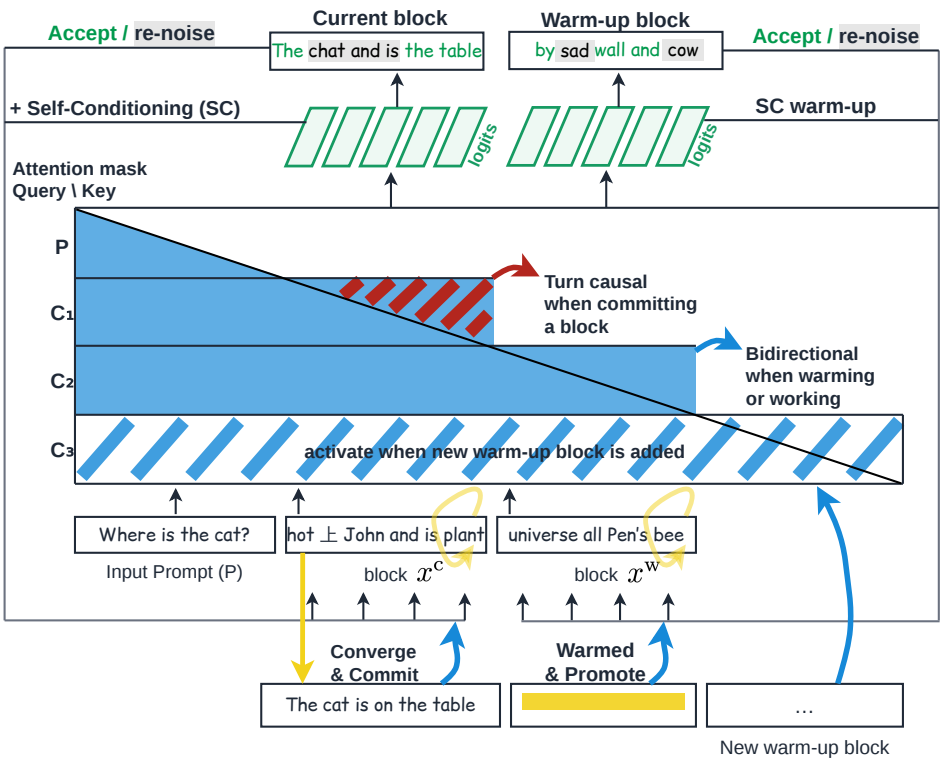


Figure 1: **One Block Ahead (OBA).** The current block x^c and the warm-up block x^w are denoised in one forward pass over the input sequence $[x^c \mid x^w]$. A block-causal attention mask lets x^w attend to x^c but not the reverse, so x^c is denoised as if it were alone. When x^c is committed and encoded under causal attention, x^w is promoted in its place, retaining the tokens and sampler state it has accumulated, and a new warm-up block follows.

while, overlapping two blocks expands a forward pass from 256 to 512 positions. Small blocks can exploit unused GPU compute capacity, but that headroom diminishes as wider blocks approach the compute-bound regime (**?**). Consequently, reducing forward-pass counts does not by itself guarantee higher tokens per second.

In this work, we propose One Block Ahead (OBA), a new training-free multi-block decoding algorithm built upon DiffusionGemma (Figure 1). To the best of our knowledge, this is the first study of concurrent multi-block decoding for a diffusion language model that already achieves around 20 tokens per forward pass with native 256-token blocks. We investigate the design choices impacting the effectiveness of concurrent decoding: how to structure attention between the two blocks (current block and warm-up block), and which components of the warm-up state must be retained when the warm-up block becomes the current block (e.g., tokens, self-conditioning signals, and the denoising time). To compare fairly how gains in tokens per forward pass transfer to throughput, we optimize the precision, attention, mixture-of-experts and sampler components of a denoising step for both the single- and the paired-block input sequence layout (Section 4.4).

Averaged over ten benchmarks on DiffusionGemma, OBA increases TPF from 15.64 to 22.25 (+42.3%) and single-request denoising throughput from 2,375 to 2,761 TPS (+16.3%) on one GB300 GPU, with a small reduction in average accuracy (Section 4). By contrast, the direct way to add parallelism, doubling the block to 512 positions, lowers throughput by 41.4% and reduces accuracy further. Our ablations identify sampler-state promotion as central to the gain: withholding self-conditioning during warm-up lowers throughput 19.9% below the baseline. Together, these results demonstrate that concurrent decoding, supported by sampler-state promotion, can improve throughput even when the baseline already achieves substantial parallelism within each block.

2 BACKGROUND

2.1 DISCRETE DIFFUSION LANGUAGE MODELS

A discrete diffusion language model generates text by reversing a corruption process on token sequences rather than by extending a sequence left to right (Austin et al., 2021a; Hoogeboom et al., 2021). A forward process maps a clean sequence x_0 to progressively noisier x_t, $t \in [0, 1]$, and a learned denoiser is trained to invert it, so each reverse step may revise every position at once. Two corruption processes are in use. *Masked* diffusion replaces a token with a dedicated mask token and reveals positions monotonically (Sahoo et al., 2024; Nie et al., 2025; Ye et al., 2025; Fu et al., 2026); *uniform-state* diffusion replaces it with another token drawn from the vocabulary, so every position holds real text and remains revisable until it is committed (Hoogeboom et al., 2021; Lou et al., 2024; Schiff et al., 2025; Ye et al., 2026). DiffusionGemma is uniform-state (DiffusionGemma Team, 2026), and the distinction is what makes state inheritance a question in Section 3.2.

Fully bidirectional diffusion language models cannot reuse a KV cache or vary their output length, since every position attends to every other. Block Diffusion removes both limits by factorizing generation over blocks, with causal attention across blocks and bidirectional attention within the current block (Arriola et al., 2025). DiffusionGemma adopts this structure and calls each block a *canvas*; to align with the broader diffusion language model literature, we use *block* throughout for this fixed-size group of consecutive positions. Specifically, a causal encoder $\mathrm{Encoder}_\theta$ turns the context h into a KV cache $H = \mathrm{Encoder}_\theta(h)$, and a bidirectional decoder refines a block of length $C = 256$ while cross-attending to H. Once a block is fully denoised, written $\hat{x}_0$, one encoder pass appends its keys and values, $H \leftarrow H \oplus \mathrm{Encoder}_\theta(\hat{x}_0)$, and only then is the next block initialized; committed tokens are therefore encoded once and never re-encoded.

2.2 DIFFUSIONGEMMA

DiffusionGemma is the model we build on: an instruction-tuned mixture-of-experts diffusion language model with 26 billion total and 4 billion active parameters per token (DiffusionGemma Team, 2026). Of the design choices behind its efficiency, three interact with OBA, and we specify them here: the entropy-bounded sampler, adaptive stopping, and the self-conditioning signal.

A block is refined over at most $N = 48$ denoising steps with step size $\Delta t = 1/N$. Let $\mathcal{V}$ be the vocabulary, so that $\mathcal{V}^C$ is the set of token sequences of length C, and let $x_t \in \mathcal{V}^C$ be the block at time $t \in [0, 1]$, starting at $t = 1$ with $x_1 \sim \mathrm{Unif}(\mathcal{V}^C)$. Each step calls the decoder on the block, the cache H and a self-conditioning signal z_t carrying the previous predictions, returning $L_t = \mathrm{Decoder}_\theta(x_t, z_t, t, H)$, from which

$$\hat{p}_0 = \mathrm{Softmax}\left(L_t/\tau_t\right), \qquad z_{t-\Delta t} = \mathrm{FFW}(\hat{p}_0 E), \qquad x_{t-\Delta t} \sim \mathrm{Step}(x_t, \hat{p}_0), \tag{1}$$

with E the embedding matrix, FFW the self-conditioning module, and τ_t falling linearly from 0.8 to 0.4. The sampler acts in Step, deciding position by position which predictions to keep. Positions are accepted in order of increasing predictive entropy until their cumulative entropy reaches a budget $b = 0.1$, which bounds the joint mutual information and so keeps them approximately independent (Ben-Hamu et al., 2025). An accepted position takes a token sampled from $\hat{p}_0$, and every other position is corrupted again by sampling a new token uniformly from $\mathcal{V}$ (Hoogeboom et al., 2021). Denoising stops early once mean entropy falls below $e_{\mathrm{stop}} = 0.005$ with consecutive steps agreeing on the argmax. Acceptance therefore binds for one step only, and no position is fixed while its block is still being refined; that ends at commitment, after which the tokens are committed permanently.

2.3 INFERENCE EFFICIENCY METRICS

Adaptive stopping makes the number of denoising steps a random variable, so DiffusionGemma measures denoising efficiency empirically (DiffusionGemma Team, 2026). Let K be the number of blocks a response spans, C_k the valid tokens produced in block k, counted up to the first end-of-sequence token, F the denoising passes executed, and T_{den} the average GPU time of a denoising pass. Tokens per forward and tokens per second are

$$\mathrm{TPF} \triangleq \frac{\sum_{k=1}^{K} C_k}{1 + (K-1) + F}, \qquad \mathrm{TPS} \triangleq \frac{\sum_{k=1}^{K} C_k}{F\, T_{\text{den}}}, \tag{2}$$

where the TPF denominator counts every forward pass that serves the request: the prompt prefill, the $K - 1$ passes that append each committed block to the cache, and the F denoising passes. TPF counts useful output per invocation and is independent of hardware. Decode TPS (throughput) counts useful output per second of denoising GPU time; it excludes the prefill and cache-append passes and host-side overhead, so that it isolates the cost of denoising (Appendix H gives wall TPS).

3 METHOD

One Block Ahead (OBA) denoises two blocks in each forward pass. The decoder runs over the input sequence $[x^{\mathrm{c}} \mid x^{\mathrm{w}}]$ of $2C$ positions, where x^{c} is the block on its way to being committed and x^{w} is the one after it. As depicted in Figure 1, a block-causal mask lets x^{w} attend to x^{c} but not the reverse. When the current block x^{c} is committed, the warm-up block x^{w} is promoted into its place and a new block fills the vacated half. In the following we present design choices for OBA: Section 3.1 specifies the paired-block layout, Section 3.2 specifies what is inherited at promotion, and Section 3.3 gives the condition under which the OBA schedule raises throughput.

3.1 PAIRED-BLOCK LAYOUT

Rather than running the decoder over the C positions of one block, we lay the two blocks side by side and run it over $2C$ positions, so one pass returns logits for both blocks. A single block still commits C tokens and the cache is still appended once per block, so OBA composes with any method acting inside a block. The block-causal mask defines the attention pattern of each block: queries in x^{c} attend to the committed cache H and to x^{c} itself, while queries in x^{w} attend to H, x^{c} and x^{w}, and attention stays bidirectional inside each half as when a block is denoised alone. No query in x^{c} reads a key in x^{w}, and every remaining operation acts on one position at a time, so in exact arithmetic the logits over x^{c} are those that block would have produced alone (cf. Appendix C).

Although the attention mask prevents the current block from attending to the warm-up block, it does not guarantee numerical equivalence to decoding the current block alone. Processing both blocks in a single grouped mixture-of-experts call changes the computation shape and can introduce floating-point differences that alter subsequent sampling decisions, so we compare configurations on matched benchmarks and seeds rather than asserting identical trajectories; per-benchmark accuracy is given in Appendix H. Appendix B describes the paired-block adaptations, including the shared quantization scales computed from the current block and used by both blocks.

3.2 STATE INHERITANCE AT PROMOTION

Both blocks take a denoising step from every forward pass, so the warm-up block accumulates progress while the current block is still changing. Which part of that progress the warm-up block keeps at promotion, i.e., when the current block commits, is what we call *state inheritance*. We denote the state of a block after its n-th pass as $s_n = (x_n, z_n, t_n, \sigma_n)$, where x_n denotes the block tokens, z_n the self-conditioning signal, t_n the denoising time, and σ_n the adaptive stopping bookkeeping (i.e. the recent argmax predictions and the stability counter). We call (z, t, σ) the *sampler state*. As DiffusionGemma uses uniform-state diffusion, every position holds a real token from the vocabulary from the first step onward, so x alone does not reveal how far a block has progressed and the sampler state cannot be reconstructed solely from x.

Assume the current block is committed after n_k paired denoising passes. Its C tokens are appended to the cache and the warm-up block moves into the current position carrying $(x_{n_k}, z_{n_k}, t_{n_k}, \sigma_{\mathrm{init}})$. The tokens, the signal and the denoising time are promoted, and only σ is reset to its initial value σ_{init}. The reset of σ prevents the promoted block from stopping on a stability judgment formed before its context has been finalized. A new warm-up block then fills the vacated half and the cycle repeats. No warm-up block commits out of order, and the final current block at the end of generation simply discards its warm-up block (cf. Algorithm 1, Appendix A).

3.3 BREAK-EVEN CRITERION

Although the paired-block layout advances two blocks per pass, it does not halve the passes per block, because a warm-up step conditions on a current block that has not yet been committed. Let

$\bar{n}$ be the expected number of denoising passes a block needs under NATIVE (C=256), and n_k the number of passes block k spends as the current block under a concurrent schedule. We model the passes block $k+1$ then needs as the current block as $\bar{n} - \alpha n_k$, so that α is the expected number of current-block passes saved per pass its predecessor spends as the current block. Under OBA, block $k+1$ takes one warm-up step on each of those passes, so α is also the saving per warm-up step; a schedule that starts the warm-up block later, such as FLOWBLOCK (W=2) (Section 4.2), takes fewer warm-up steps per predecessor pass and therefore has a smaller α even if each step is worth as much. We make two simplifying assumptions: (1) α is a constant, independent of n_k and of k, i.e., the saving is linear in the predecessor's passes; and (2) $\bar{n}$ does not depend on k. Assumption (1) is an approximation rather than a property of the sampler; we fit α under it from aggregate counters (cf. Appendix D) and do not test it at the level of individual blocks. We further expect $0 \le \alpha < 1$: a warm-up step should not increase the passes a block needs, and it saves less than a full pass because it conditions on a current block that has not yet been committed. Linearity also makes the recurrence exact in expectation, so under these assumptions the expected passes obey the affine recurrence

$$n_{k+1} = \bar{n} - \alpha\, n_k, \qquad n_1 = \bar{n}, \qquad \text{with unique fixed point} \quad n^\star = \frac{\bar{n}}{1+\alpha}. \tag{3}$$

The map $n \mapsto \bar{n} - \alpha n$ is a contraction of modulus $\alpha < 1$, so $n^\star$ exists and is unique. It is also approached geometrically, since $|n_k - n^\star| = \alpha^{k-1}|\bar{n} - n^\star|$. Let r denote the ratio of T_{den} in Eq. (2) under a schedule to that under NATIVE (C=256), measured at batch size one on the same hardware. Since T_{den} averages over all denoising passes of a request, r is the relative cost of a paired-block pass when every denoising pass is paired, as in OBA, and lies between one and that value when a schedule mixes single- and paired-block passes. At the fixed point, denoising passes per block therefore fall by a factor $1+\alpha$ while a denoising pass takes r times as long, so TPS changes by $(1+\alpha)/r$, and the schedule raises throughput when $\alpha > r - 1$.

4 EXPERIMENTS

We design the experimental evaluation to understand three different aspects of OBA: (1) Does denoising one block ahead raise throughput over single-block and multi-block baselines across diverse tasks, and at what accuracy cost? (2) Which parts of the sampler state carried across promotion make the wider denoising pass pay for itself? (3) How do the attention mask between the two blocks and the time at which the next block starts affect the break-even criterion $\alpha > r - 1$?

4.1 EXPERIMENTAL SETUP

Model and benchmarks. We evaluate DiffusionGemma across ten benchmarks on a single NVIDIA GB300 at batch size one: GSM8K, MATH-500, MBPP, HumanEval, GPQA, IFEval, MMLU, LiveCodeBench C++, AIME 2024 and AIME 2025 (Cobbe et al., 2021; Hendrycks et al., 2021b; Austin et al., 2021b; Chen et al., 2021; Rein et al., 2024; Zhou et al., 2023; Hendrycks et al., 2021a; Jain et al., 2025). We report the main comparison averaged over three generation seeds; Appendix E gives the details about the evaluated population, graders, seeds and serving limits. While the proposed paired-block layout applies to any block-diffusion model, uniform-state diffusion is what makes state inheritance at promotion a design question. Existing uniform-state models, e.g., Sumi (Ye et al., 2026), UDLM (Schiff et al., 2025) and Duo (Sahoo et al., 2025), are trained and sampled over a full sequence in a single diffusion trajectory, so no state has to cross a block commitment boundary. DiffusionGemma is, to our knowledge, the first publicly released large-scale model that is both uniform-state and block-autoregressive.

Baselines. NATIVE (C=256) is the default DiffusionGemma inference configuration, denoising one 256-token block at a time. NATIVE (C=512) applies the same algorithm to a single 512-token block and commits all 512 positions at once, the natural way to increase parallelism by widening a block. FLOWBLOCK (W=2) is a concurrent, training-free multi-block decoder built on the LLaDA2.1 token-editing model (Tian et al., 2026; Bie et al., 2026); we apply it by replacing its masked diffusion model forward with DiffusionGemma's (cf. Appendix F). It decodes up to W blocks concurrently, and we use W=2, the window of its main results. It starts the next block once its readiness gate exceeds a threshold, and since blocks are decoded concurrently, that block keeps its tokens, self-conditioning signal and denoising time when its predecessor commits. It therefore differs from OBA

Table 1: **Decoding efficiency across ten benchmarks.** Tokens per forward pass (TPF) and tokens per second (TPS). TPS counts denoising GPU time only (Eq. (2)). All metrics are averaged over three generation seeds. Best in bold. OBA achieves higher TPF and TPS than all baselines on all benchmarks. See benchmark accuracy and wall TPS in Appendix H.

	NATIVE (C=256)		NATIVE (C=512)		FLOWBLOCK (W=2)		OBA	
Benchmark	TPF	TPS	TPF	TPS	TPF	TPS	TPF	TPS
GSM8K	17.59	2800	19.93	2128	18.33	2816	**22.87**	**2983**
MATH-500	17.96	2798	17.52	1503	18.26	2742	**23.05**	**2911**
MBPP	16.43	2485	16.64	1208	16.89	2516	**22.41**	**2733**
HumanEval	18.93	2963	21.98	1996	22.04	3447	**29.22**	**3858**
GPQA	13.13	1919	13.92	1018	14.30	2036	**20.38**	**2435**
IFEval	15.87	2433	18.51	1787	17.43	2566	**23.78**	**3000**
MMLU	13.16	1997	15.72	1430	14.05	2061	**20.77**	**2611**
LCB C++	14.37	2090	14.21	902	14.34	2054	**19.37**	**2230**
AIME 2024	14.59	2159	14.26	1023	15.50	2247	**19.95**	**2410**
AIME 2025	14.38	2101	14.01	917	16.42	2339	**20.72**	**2442**
Average	15.64	2375	16.67	1391	16.76	2482	**22.25**	**2761**

mainly in when the next block starts, and we use it as our comparison for the start time of the warm-up block. We exclude other multi-block methods (Wang et al., 2026; Jin et al., 2026), which target masked diffusion models that cannot revise a revealed token. We further isolate OBA's mechanism through ablations of state inheritance and attention between the two blocks. For fair comparison, all configurations above run with kernels specialized to their own input sequence shape.

4.2 DECODING EFFICIENCY

TPF and throughput. We investigate the decoding efficiency of the proposed OBA, with the main results summarized in Table 1. OBA has higher TPF and TPS than all baselines on all ten benchmarks. Averaged over the ten benchmarks, OBA increases TPF by 42.3% over NATIVE (C=256) (22.25 vs. 15.64) and TPS by 16.3% (2,761 vs. 2,375), even though each denoising pass costs 1.26× as much. The margin over NATIVE (C=256) is largest on MMLU, HumanEval and GPQA, where TPS rises 31%, 30% and 27% and TPF 58%, 54% and 55%, and smallest on MATH-500, where TPS rises 4%. MMLU and GPQA are the benchmarks where a block needs the most denoising passes under NATIVE (C=256), so the current block occupies the input sequence for longer and the warm-up block accumulates more steps before it is promoted. These gains come with a 1.37-percentage-point reduction in average accuracy over the ten benchmarks (cf. Table 6).

Comparison with FlowBlock. FLOWBLOCK (W=2) raises average TPF over NATIVE (C=256) by 7.2% (15.64 to 16.76) and TPS by 4.5% (2,375 to 2,482), against 42.3% and 16.3% for OBA. Since it retains the same sampler state, the difference comes from when it starts the next block, which moves both sides of the break-even criterion of Section 3.3: waiting for the readiness threshold keeps 92.2% of denoising passes 256 positions wide, lowering r to 1.03, but forfeits most warm-up steps, lowering α to 0.09. On DiffusionGemma, the gated start thus lowers α by more than it lowers $r-1$, so the margin $\alpha-(r-1)$ is 0.06 under FLOWBLOCK (W=2) against 0.28 under OBA, which starts the warm-up block at the earliest possible pass. The readiness gate guards against drafting on an unsettled prefix, but under uniform-state diffusion an early draft is not locked in: even under NATIVE (C=256), only 5.1% of positions keep their first-pass token until commitment (Figure 2), since every position not accepted is renoised. The gate does retain 0.73 points of accuracy over OBA (83.43 against 82.70), but at the cost of most of the throughput gain. Widening FLOWBLOCK's window to W=3 or 4 also makes little difference, with TPF and TPS within 1% of W=2 (cf. Appendix F).

Comparison with a wider block. A block-diffusion decoder raises parallelism by committing a wider block, but DiffusionGemma's 256-token block is already close to the point where a single-request forward pass saturates the arithmetic intensity of the GPU (**?**), so the extra positions are no longer nearly free. NATIVE (C=512) raises TPF from 15.64 to 16.67, but a denoising pass over 512 positions costs roughly 1.84× one over 256 positions, so its throughput is 41.4% below NATIVE (C=256). The paired-block layout of OBA occupies the same 512 positions at 1.26× the reference

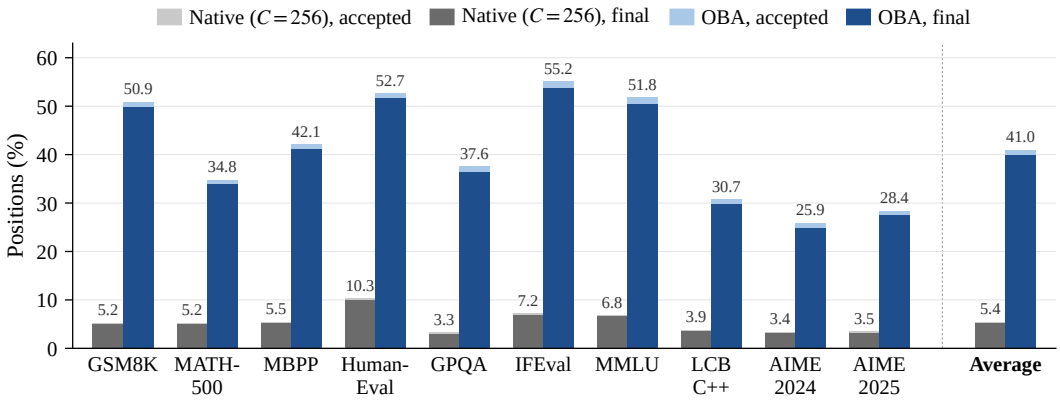


Figure 2: **Warm-up progress of promoted and newly initialized blocks.** For each benchmark, the grey bar corresponds to NATIVE (C=256) and the blue bar to OBA. The full height of a bar is the fraction of the 256 positions accepted on a block's first pass as the current block, and its dark portion is the fraction whose token at that pass is retained until commitment, so the light segment on top is the fraction of first-pass acceptances later revised. Under OBA the first pass is the one after promotion, and every first block, which has no predecessor, is excluded. Fractions are averaged over three generation seeds. On average, a promoted block under OBA begins its first pass as the current block with 39.9% of its positions already holding their committed tokens, against 5.1% for a newly initialized block under NATIVE (C=256). Per-benchmark values are given in Table 5.

cost, largely because its attention is block-causal rather than dense; Section 4.3 ablates the mask directly. Simply widening the block is therefore not the axis to scale up.

Warm-up progress at promotion. Figure 2 measures what a block has settled on its first pass as the current block, which under OBA is the first pass after promotion, conditioned on the newly committed context, and under NATIVE (C=256) the first pass after initialization from noise. Averaged over the ten benchmarks, 41.0% of the 256 positions of a promoted block are accepted on that pass, against 5.4% for a newly initialized block, and 39.9% against 5.1% keep that pass's token until commitment, so a promoted block arrives with more than a third of its positions already holding their committed tokens. The fraction of first-pass acceptances retained until commitment is similar under both schedules, 97% and 94%. The state inherited from warm-up therefore lets many more positions clear the entropy budget on the first pass under the committed context without making those acceptances less reliable, so promotion raises how many positions settle early rather than how reliably they settle. The effect varies with the task. More than half of the positions are accepted on the first pass on IFEval, HumanEval, MMLU and GSM8K, against 26–31% on AIME and LiveCodeBench C++, where a block plausibly depends more on the exact content of its predecessor.

4.3 ABLATION STUDY

We ablate what a warm-up block inherits and how it is formed in Table 2. KEEP ALL retains the full state and is OBA; DROP TIME drops the denoising time, KEEP SIGNAL keeps the self-conditioning signal alone, and KEEP TOKENS keeps the tokens alone. Up to its start threshold, KEEP TOKENS matches the original FlowBlock (Tian et al., 2026), where a block carries only its tokens across the commitment boundary, as under masked diffusion. Three further policies change the warm-up process rather than the inherited state: NO ACCEPT keeps the self-conditioning signal but accepts no position before promotion, NO SIGNAL accepts positions but withholds self-conditioning, and DENSE MASK replaces the block-causal mask with full bidirectional attention between the two blocks. We estimate α for each configuration by fitting the recurrence of Eq. (3) on each benchmark to its numbers of denoising passes and blocks and averaging over benchmarks; Appendix D gives the details. Since the cache grows with each response, r also depends on response length; the six paired-layout policies have r between 1.26 and 1.27.

Table 2: **Inheritance policies.** α, relative pass cost r, efficiency, and accuracy for seven policies, averaged over the ten benchmarks. Retains gives the inherited state in the (x, z, t) decomposition; Before promotion marks the three policies that instead change what a block does before promotion; ΔTPS is against NATIVE (C=256).

Policy	Retains	Before promotion	α	r	TPF	TPS	ΔTPS %	Acc.
NATIVE (C=256)	–	–	0	1.00	15.64	2375	–	84.07
KEEP ALL (OBA)	x, z, t	full	0.54	1.26	22.25	2761	+16.3	82.70
DROP TIME	x, z	full	0.41	1.26	20.76	2558	+7.7	82.69
KEEP TOKENS	x	full	0.25	1.26	18.83	2309	−2.8	83.55
KEEP SIGNAL	z	full	0.32	1.27	19.61	2401	+1.1	83.93
NO ACCEPT	z	no acceptance	0.32	1.27	19.62	2403	+1.2	84.95
NO SIGNAL	x	no self-conditioning	0.02	1.27	15.87	1902	−19.9	84.11
DENSE MASK	x, z, t	dense mask	0.38	1.70	20.05	1878	−20.9	80.29

Inherited state. Recall that α is the number of current-block passes saved per warm-up step. Promoting the tokens alone, as a masked diffusion decoder would, gives $\alpha = 0.25$, below the break-even value $r - 1 = 0.26$, and changes TPS by -2.8% relative to NATIVE (C=256). Adding the self-conditioning signal raises α to 0.41 and TPS by 7.7%, and adding the time raises α to 0.54 and TPS by 16.3%, since a block that restarts its temperature schedule spends passes re-deciding positions it had already settled. Each component of the sampler state therefore adds to the saving per warm-up step, and together they account for the entire throughput gain. Carried alone, the signal is worth more than the tokens: KEEP SIGNAL reaches $\alpha = 0.32$ against 0.25 for KEEP TOKENS.

Warm-up process. NO ACCEPT and KEEP SIGNAL both reach $\alpha = 0.32$, with TPS changing by $+1.2\%$ and $+1.1\%$ relative to NATIVE (C=256). Once the self-conditioning signal is propagated during warm-up, whether some positions were allowed to settle makes little difference. On the other hand, NO SIGNAL lowers α to 0.02, and changes TPS by -19.9% relative to NATIVE (C=256). Without self-conditioning, the warm-up does no useful work while the schedule still pays for the wider pass, falling below the break-even criterion of Section 3.3.

Attention between blocks. DENSE MASK keeps the promoted state and the commitment cadence of OBA but lets each block attend to the other, which removes the guarantee of Proposition 1. It still raises TPF over NATIVE (C=256), 20.05 against 15.64, at $\alpha = 0.38$, so the warm-up block does useful work under mutual attention, though less than under the block-causal mask. However, a pass under DENSE MASK costs $1.70\times$ against $1.26\times$ for the block-causal one, so break-even would require $\alpha > 0.70$, and throughput falls 20.9% below NATIVE (C=256). The gap to the $1.84\times$ of NATIVE (C=512), which shares its input sequence shape and kernels, reflects the context lengths of their responses rather than the mask. It also lowers average accuracy relative to OBA, 80.29 against 82.70, since the current block now conditions on a block that is still near noise.

4.4 KERNEL IMPLEMENTATION

Shared kernels. Every configuration runs the same implementation, which we optimize for the GB300 in each component of a denoising step (Eq. (1)). In precision, the mixture-of-experts layers use a grouped matrix multiplication with 4-bit weights and activations (NVFP4) in place of 4-bit weights with 16-bit activations, the dense projections use FP8, the rotary tables are kept in FP32, and amax reduction, scale finalization and row quantization are fused into one pass. In attention, the kernels are fused and additionally parallelized over the cache H, the ragged masked softmax of the fallback path is vectorized, and the sliding-window keys and values are held in a ring buffer. In the mixture-of-experts and dense layers, the input producer is fused, the combine stage is sharded over the hidden dimension, the routing scan is parallelized in a deterministic accumulation order, and the two branches of a layer execute concurrently on separate streams. In the sampler, the self-conditioning module FFW is vectorized. Table 1 therefore compares decoding algorithms at a common level of optimization, and the gains are over an already optimized single-block decoder.

Per-configuration kernels. Each configuration additionally selects the kernels specialized to its own input sequence shape. NATIVE (C=256) uses the 256-position attention kernels; NATIVE (C=512) and DENSE MASK use matrix-multiplication and mixture-of-experts plans tuned for 512

positions, together with a dispatch correction without which those attention kernels are disabled at that length; OBA adds offset-aware attention that skips the masked-out half of the paired-block layout and a fused producer over both halves, neither of which applies to a dense input sequence; FLOWBLOCK (W=2) uses the NATIVE (C=256) kernels for single-block passes and the OBA kernels for paired ones. The one numerical difference across configurations is that OBA computes its activation-scale from the current block and shares it with the warm-up half. A denoising pass takes 7.08 ms under NATIVE (C=256), 8.94 ms under OBA and 13.06 ms under NATIVE (C=512); averaged over benchmarks, a paired pass costs $r = 1.26$ against $r = 1.84$ for a dense pass over the same 512 positions. Appendix B gives the full details.

5 RELATED WORK

Within-block acceleration. One line of work reduces the number of denoising passes a single block requires. Fast-dLLM (Wu et al., 2026) pairs a block-wise approximate KV cache with confidence-thresholded acceptance, entropy-bounded sampling (Ben-Hamu et al., 2025) replaces the per-position threshold with a cumulative-entropy budget that bounds the joint mutual information the acceptance discards, and dParallel (Chen et al., 2026) distills a model along its own decoding trajectory while forcing predictive certainty, so that fewer steps suffice. Editing reaches the same end by a different route: the token-to-token stream that LLaDA2.1 (Bie et al., 2026) adds to LLaDA2.0 (Bie et al., 2025) overwrites a revealed token once an alternative becomes sufficiently confident, which permits a lower acceptance threshold because a premature decision can be repaired rather than carried into the output. Uniform-state diffusion has that property natively, since every position holds a real token and stays revisable until its block commits. All of these act within the block being denoised, whereas OBA changes how many blocks a pass advances; DiffusionGemma already carries several of them, so OBA starts from a baseline at around 20 TPF (DiffusionGemma Team, 2026) rather than the low single digits at which prior concurrent decoding methods are evaluated.

Multi-block decoding. Another line of work focuses on serialization across blocks. D2F (Wang et al., 2026) and MBD-LM (Jin et al., 2026) post-train a masked diffusion LM to denoise a block from an incomplete prefix and decode a bounded window of active blocks; FlowBlock (Tian et al., 2026) does so without training, admitting a downstream block once most of its predecessor's remaining masked positions are confident enough to be unmasked, and relying on token-to-token editing to repair tokens drafted against a stale prefix. Three properties separate this line from OBA. First, each targets masked diffusion, where the mask ratio reports a block's progress and a promoted block carries nothing beyond its revealed tokens; under uniform state progress is invisible in x, and the sampler state (z, t, σ) must cross the commitment boundary (Section 3.2). Second, all use block sizes close to 32, where the extra positions are close to free; at DiffusionGemma's 256-token block a paired-block pass costs $r = 1.26$, so the schedule pays only when $\alpha > r - 1$ (Section 3.3). Third, OBA is training-free and needs no threshold to decide when to start the warm-up block; applied to DiffusionGemma with the same sampler state, FlowBlock's threshold delays the next block enough that its throughput gain over NATIVE (C=256) is 4.5%, against 16.3% for OBA (Section 4.2).

6 CONCLUSION

We presented One Block Ahead (OBA), a training-free decoding algorithm that denoises the next block alongside the current one in a single forward pass. A block-causal mask leaves the current block denoised as if it were alone, and at commitment the warm-up block is promoted with its tokens and sampler state. Such a schedule raises throughput when the passes saved per warm-up step, α, exceed the extra cost of a paired-block pass, $r - 1$. On DiffusionGemma, averaged over ten benchmarks, OBA meets this condition with $\alpha = 0.54$ against $r - 1 = 0.26$, raising tokens per forward pass (TPF) from 15.64 to 22.25 and single-request denoising throughput (TPS) by 16.3% at a cost of 1.37 accuracy points. Ablations attribute this gain to the sampler state: promoting the tokens alone gives $\alpha = 0.25$ and lowers throughput, and adding the self-conditioning signal and then the denoising time raises α to 0.41 and 0.54. More broadly, these results show that parallelism across blocks remains available to a model that already decodes one block efficiently, and that it is unlocked by promoting the sampler state across the commitment boundary.

7 LIMITATIONS

We evaluate at batch size one, where the latency advantage of diffusion over autoregressive decoding is most pronounced, and on a single GPU type. We evaluate a single model, as DiffusionGemma is to our knowledge the only diffusion language model that is both uniform-state and block-autoregressive. The promotion mechanism is specific to that setting: under masked diffusion a block's progress is readable from its mask and its revealed tokens are the only state to inherit.

AI USE STATEMENT

In this work, we used generative AI coding agents (Claude Code and Codex) to implement methods: the OBA decoding algorithm, the baseline configurations including the FLOWBLOCK adaptation, the inference kernels described in Appendix B, and the evaluation pipeline, and to run the experiments. We have not used generative AI tools to propose research ideas or hypotheses, to develop the conceptual framework, to design the methodology or experiments, to formulate the mathematical claims or write the proofs in Section 3.3 and Appendix C, or to interpret results; these were done by the authors. Generating synthetic data, cleaning or reformatting datasets, translation, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to edit parts of the manuscript for readability. We have reviewed all AI-assisted work, and every AI-assisted component was verified by at least two authors: the generated code was reviewed, the kernel-level changes were validated against reference implementations for numerical agreement and unchanged accuracy, the reported numbers were checked against the per-case run records, and every claim, equation, and citation in the manuscript was checked. We take responsibility for the final content of this work, including text, claims, code, and experimental results produced with the aid of generative AI.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. All experiments use publicly available benchmarks and a pretrained model (DiffusionGemma), with our contributions centered on methodological and algorithmic advancements, in particular the development of the One Block Ahead (OBA) decoding algorithm. The method is training-free and introduces no new capabilities beyond those of the underlying model. The study does not involve human subjects, personal data, or other sensitive information, and no applications with a high likelihood of causing harm are considered. We have carefully examined potential risks and broader societal impacts of this research and did not identify significant ethical concerns.

REPRODUCIBILITY STATEMENT

OBA is training-free and requires no change to model weights. The decoding algorithm is specified in Section 3 and in full as Algorithm 1 in Appendix A; the attention mask, the promoted state, and the reset at promotion are given in Sections 3.1 and 3.2. The assumptions behind the break-even criterion are stated in Section 3.3. Section 4.1 lists the model, the ten benchmarks, the generation seeds, the hardware, and the baseline configurations, and all sampler hyperparameters are the DiffusionGemma defaults reported in Section 2.2. Appendix E gives the seed offsets and serving limits, Appendix F gives the FLOWBLOCK thresholds, and Appendix B describes the kernel-level adaptations needed to reproduce the reported throughput, since it, unlike TPF, depends on the implementation.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, 2025.

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

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

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S. Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems*, 2025.

Tiwei Bie, Maosong Cao, Kun Chen, et al. LLaDA2.0: Scaling up diffusion language models to 100B. *arXiv preprint arXiv:2512.15745*, 2025.

Tiwei Bie, Maosong Cao, Xiang Cao, et al. LLaDA2.1: Speeding up text diffusion via token editing. *arXiv preprint arXiv:2602.08676*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dParallel: Learnable parallel decoding for dLLMs. In *International Conference on Learning Representations*, 2026.

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

DiffusionGemma Team. DiffusionGemma technical report. *arXiv preprint arXiv:2608.00146*, 2026.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, Chenhan Yu, Ye Yu, Weijia Chen, Sajad Norouzi, Jingyu Liu, Shiyi Lan, Ligeng Zhu, Jin Wang, Jindong Jiang, Morteza Mardani, Mehran Maghoumi, Song Han, Ante Jukić, Nima Tajbakhsh, Jan Kautz, and Pavlo Molchanov. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. Technical report, NVIDIA, 2026. arXiv:2607.05722.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021b.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems*, 2021.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida I. Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025.

Yijie Jin, Jiajun Xu, Yuxuan Liu, Chenkai Xu, Yi Tu, Jiajun Li, Dandan Tu, Xiaohui Yan, Kai Yu, Pengfei Liu, and Zhijie Deng. Multi-block diffusion language models. *arXiv preprint arXiv:2606.29215*, 2026.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, 2024.

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander M. Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T. Chiu, and Volodymyr Kuleshov. The diffusion duality. In *International Conference on Machine Learning*, 2025.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-Torre, Bernardo P. de Almeida, Alexander M. Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. In *International Conference on Learning Representations*, 2025.

Bing Tian, Haikun Liu, Xiaocheng Zhong, Zhuohui Duan, Zhaokai Luo, Huayi Jin, Zhiyong Wang, and Xiaofei Liao. FlowBlock: Wavefront-parallel decoding for self-correcting diffusion language models. *arXiv preprint arXiv:2607.17652*, 2026.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, Kai Yu, and Zhijie Deng. Diffusion LLMs can do faster-than-AR inference via discrete diffusion forcing. In *International Conference on Learning Representations*, 2026.

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

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

Mengyu Ye, Keito Kudo, Wataru Ikeda, Ryosuke Matsuda, Keisuke Sakaguchi, and Jun Suzuki. Sumi: Open uniform diffusion language model from scratch. *arXiv preprint arXiv:2606.19005*, 2026.

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

A COMPLETE DECODER STATE MACHINE

Algorithm 1 gives the full decoding loop, with the additions to the NATIVE (C=256) sampler marked in blue. The state machine below describes the same loop in words, for $C = 256$.

Algorithm 1 DiffusionGemma sampling with one-block-ahead decoding

Require: Context h; block length $C = 256$; maximum denoising steps $N = 48$
Require: Entropy budget $b = 0.1$; stopping threshold $e_{\text{stop}} = 0.005$
Require: Temperature endpoints $\tau_{\max} = 0.8$, $\tau_{\min} = 0.4$; output stopping rule
Require: Warm-up block x^{w} and paired input sequence $[x^{\text{c}} \mid x^{\text{w}}]$

function DENOISESTEP(L, t)
  $\tau \leftarrow (\tau_{\max} - \tau_{\min})t + \tau_{\min}, \quad \hat{p}_0 \leftarrow \text{Softmax}(L/\tau)$
  $\hat{x}_0 \leftarrow \arg\max L, \quad e^i \leftarrow \text{Entropy}(\hat{p}_0^i)$
  $U \leftarrow \text{LowEntropySet}((e^i)_{i=1}^C, b)$ ▷ Entropy-bounded accept set
  Draw $x_D^i \sim \hat{p}_0^i$ and $\tilde{x}^i \sim \text{Unif}(\mathcal{V})$
  $x_U^+ \leftarrow x_{D,U}, \quad x_{\bar{U}}^+ \leftarrow \tilde{x}_{\bar{U}}, \quad z^+ \leftarrow \text{FFW}(\hat{p}_0 E)$
  return $(\hat{x}_0, e, x^+, z^+)$
end function
$H \leftarrow \text{Encoder}_\theta(h), \quad \Delta t \leftarrow 1/N, \quad Y \leftarrow ()$
$x^{\text{c}} \sim \text{Unif}(\mathcal{V}^C), \quad z^{\text{c}} \leftarrow \mathbf{0}$
$x^{\text{w}} \sim \text{Unif}(\mathcal{V}^C), \quad z^{\text{w}} \leftarrow \mathbf{0}$
$n_{\text{inh}} \leftarrow 0, \quad n_{\text{w}} \leftarrow 0$ ▷ n_{inh}: steps inherited by x^{c}; n_{w}: steps accumulated by x^{w}
loop
  $\hat{x}_0^{\text{c,prev}} \leftarrow \perp$ ▷ Reset σ after promotion
  $c_{\text{c}} \leftarrow \min\{n_{\text{inh}}, N-1\}$ ▷ Reserve one committed-context step
  for $n = 1, \ldots, N - c_{\text{c}}$ **do**
    $t_{\text{c}} \leftarrow 1 - (n - 1 + c_{\text{c}})\Delta t$
    $t_{\text{w}} \leftarrow 1 - (n-1)\Delta t$
    $(L^{\text{c}}, L^{\text{w}}) \leftarrow \text{Decoder}_\theta([x^{\text{c}} \mid x^{\text{w}}], [z^{\text{c}} \mid z^{\text{w}}], (t_{\text{c}}, t_{\text{w}}), H)$ ▷ x^{c} sees H and itself; x^{w} also sees x^{c}
    $(\hat{x}_0^{\text{c}}, e^{\text{c}}, x^{\text{c},+}, z^{\text{c},+}) \leftarrow \text{DENOISESTEP}(L^{\text{c}}, t_{\text{c}})$
    $(\hat{x}_0^{\text{w}}, e^{\text{w}}, x^{\text{w},+}, z^{\text{w},+}) \leftarrow \text{DENOISESTEP}(L^{\text{w}}, t_{\text{w}})$ ▷ No stopping test for the warm-up block
    $(x^{\text{w}}, z^{\text{w}}, n_{\text{w}}) \leftarrow (x^{\text{w},+}, z^{\text{w},+}, n_{\text{w}} + 1)$ ▷ Keep the update even if the current block stops
    $\text{stop}_{\text{c}} \leftarrow (\hat{x}_0^{\text{c,prev}} \neq \perp) \wedge (\text{mean}(e^{\text{c}}) \leq e_{\text{stop}}) \wedge (\hat{x}_0^{\text{c}} = \hat{x}_0^{\text{c,prev}})$
    if $\text{stop}_{\text{c}} \vee n = N - c_{\text{c}}$ **then break**
    $(x^{\text{c}}, z^{\text{c}}, \hat{x}_0^{\text{c,prev}}) \leftarrow (x^{\text{c},+}, z^{\text{c},+}, \hat{x}_0^{\text{c}})$
  end for
  $y \leftarrow \text{Publish}(\hat{x}_0^{\text{c}}), \quad Y \leftarrow Y \,\|\, y$ ▷ Commit the current block only
  if output stopping rule is met then return Y
  $H \leftarrow H \oplus \text{Encoder}_\theta(y)$
  $(x^{\text{c}}, z^{\text{c}}, n_{\text{inh}}) \leftarrow (x^{\text{w}}, z^{\text{w}}, n_{\text{w}})$ ▷ Promote x^{w} with its denoising time
  $x^{\text{w}} \sim \text{Unif}(\mathcal{V}^C), \quad z^{\text{w}} \leftarrow \mathbf{0}, \quad n_{\text{w}} \leftarrow 0$
end loop

1. Encode the prompt causally to form H. Initialize the current block x^{c} and the warm-up block x^{w} from independent noise streams, with null self-conditioning, initial denoising time, and fresh stopping bookkeeping. Hold them as one input sequence $[x^{\text{c}} \mid x^{\text{w}}]$ of 512 positions.
2. On every denoising iteration, evaluate $[x^{\text{c}} \mid x^{\text{w}}]$ in one block-causal forward pass. Queries in x^{c} cannot read keys in x^{w}, and queries in x^{w} read H, x^{c} and x^{w}. This is one forward pass and one denoising step for each half.
3. Apply the same token transition and self-conditioning update to both halves. Only x^{c} is tested against the stopping rule. The warm-up block is not committed, but its tokens, self-conditioning signal and position on the denoising time advance.
4. When x^{c} stops, commit its $C = 256$ tokens up to the first end-of-sequence token. If that slice terminates the request, discard the warm-up half. Otherwise append the committed tokens to H through the ordinary causal encoder pass, counted once per block as the $K - 1$ term of Eq. (2) and not as a denoising pass.
5. After the append, move the warm-up half into the current positions, preserving its tokens x^{w}, its signal z^{w} and its time t^{w}, and reset σ, so that the promoted block is reconsidered under the newly committed context.

6. Seed a new 256-position warm-up block in the vacated half and resume the same 512-position loop. Promotion never creates a third live block, since the old current half has moved into H.

A promoted block inherits at most $N-1$ of its N steps, the $c_{\mathrm{c}} \leftarrow \min\{n_{\mathrm{inh}}, N-1\}$ line above, so at least one pass always runs under the committed context. NATIVE (C=256) uses the same causal append after each committed 256-token slice, and so do DENSE MASK, which differs from OBA only in its mask. NATIVE (C=512) commits and appends 512 tokens at once and has no midpoint append.

B KERNEL IMPLEMENTATION

This appendix expands Section 4.4. All five configurations of the three-seed comparison run on one inference engine, which we optimize for a single NVIDIA GB300 so that a difference in throughput reflects a difference in decoding algorithm rather than in kernel quality. DENSE MASK and FLOWBLOCK (W=2) run from later builds of that engine that add their decoding loops; each build was qualified by reproducing every generated token, finish reason and forward count of NATIVE (C=256), OBA and NATIVE (C=512) on a set of archived GSM8K cases. All configurations share the model export, sampling parameters, entropy-bounded sampler, stopping rule, batch size and plan source for the dense kernels, and each receives a fresh kernel profile for its own input sequence shape, so no configuration is measured on a code path specialized for another. Appendices B.1–B.4 describe the shared kernels, organized by the component of a denoising step (Eq. (1)) they act on; Appendix B.5 describes what each configuration selects for its own block layout; Appendix B.6 gives the measurement protocol behind T_{den} and r. We describe each at the level a reimplementation requires rather than by internal kernel name.

B.1 PRECISION AND QUANTIZATION

The mixture-of-experts layers use a grouped matrix multiplication with 4-bit weights and activations in the NVFP4 format, replacing an implementation with 4-bit weights and 16-bit activations; routing, gather, activation quantization, the GELU/GLU stage and the combine stage keep their original structure, and the combine stage preserves a deterministic accumulation order over route slots. The dense projections use FP8 and the rotary tables are stored in FP32, with a load-time check that their storage type matches the type the compiled kernels expect. Amax reduction, scale finalization and row quantization are fused into one pass, which removes the small scale-conversion nodes that otherwise separate them, and consumers are launched with a dependent launch on the producer edges we validate individually.

B.2 ATTENTION

The decoder uses fused attention kernels, with a separate causal entry for prompt and context prefill, and is additionally parallelized over the cache H so that a single request at batch size one occupies more of the device. The ragged masked softmax of the fallback path is vectorized, and its scalar variant is retained for exact cross-variant validation. Keys and values of the sliding-attention layers are held in a ring buffer, which reduces their allocation and is what makes two live blocks affordable at the context lengths we evaluate.

B.3 MIXTURE-OF-EXPERTS AND DENSE LAYERS

The mixture-of-experts input producer is fused, the combine stage is sharded over the hidden dimension, the routing bin-prefix scan is parallelized, the GELU/GLU stage is vectorized, and the gather-and-quantize stage issues 128-bit loads. The dense and mixture-of-experts branches of a layer are data-independent until their outputs are combined, so we execute them concurrently on separate streams; the transformation is bit-exact and improves throughput in an A/B test. Tuned plans are provided for both the 256- and the 512-position shapes.

B.4 SAMPLER

The output pass of the self-conditioning module FFW is vectorized, and its specialization is compiled at two blocks per multiprocessor, which doubles occupancy without spilling to local memory. The entropy-bounded acceptance rule of Section 2.2 is computed in one reduction, shared by both halves of a paired pass.

B.5 KERNELS PER BLOCK LAYOUT

NATIVE (C=256) denoises 256 positions and uses the corresponding attention kernels. NATIVE (C=512) and DENSE MASK denoise 512 positions under a dense mask; both use the matrix-multiplication and mixture-of-experts plans tuned for that shape, and both require a dispatch correction, without which the fused attention kernels are disabled whenever the input sequence holds 512 positions and the slower fallback path is taken silently. Their T_{den} therefore differ only through the context lengths their responses reach (Appendix B.6). FLOWBLOCK (W=2) switches between a 256-position pass, which uses the NATIVE (C=256) kernels, and a paired 512-position pass once its next block has started, which uses the OBA kernels below. OBA denoises the same 512 positions under the block-causal mask of Section 3.1 and adds three specializations, whose guards require batch size one, a 512-position input sequence, a 256-token commitment, the block-causal layout and the inherited timescale, and which fail closed outside that contract.

Offset-aware paired attention. The sliding-attention layers see the input sequence as two 256-query blocks with different visibility: x^{c} attends to H and to itself, and x^{w} additionally to x^{c}. One dispatch consumes the two device offsets and lengths over their overlapping cached key and value intervals rather than issuing two calls, and skips the masked-out quadrant. Neither NATIVE (C=256) nor NATIVE (C=512) has two visibility views.

Full-sequence producers with current-block quantization scales. Producer sequences shared by both halves run over all 512 positions, while ownership of the dynamic low-precision scales stays with the leading 256 positions, as the commitment contract requires; the per-sequence expert maxima this path produces feed the descriptor builder for x^{c} directly, avoiding a second scan of the leading half. The mixture-of-experts operators consequently process the paired-block as one grouped call rather than two independent 256-position calls.

Block-parallel completion metadata. A batch-one completion kernel assigns one thread per committable position, performs fixed-order end-of-sequence reductions, preserves both halves when x^{c} completes, and commits only the leading 256 positions.

None of these changes the state transition, the random draws, the stopping rule or the forward accounting; they change how the input sequence is dispatched. Two consequences are worth stating. First, a paired pass computes its activation scale from x^{c} alone and shares it with x^{w}, so the scale a position sees does not depend on the warm-up half; a scale reduced over both halves would make the logits over x^{c} depend on x^{w} through quantization, which the attention mask cannot prevent. Second, grouped execution over 512 rows changes the reduction shape of the matrix multiplications, so the logits over x^{c} are not bitwise identical to those of a single-block pass even under that ownership rule, and a floating-point difference can change which positions the sampler accepts at a step. This is why the comparison uses matched prompts and seeds rather than asserting identical trajectories (Proposition 1). Committed keys and values are written by the separate encoder pass counted as the $K - 1$ term of Eq. (2), rather than from the denoising pass itself, which we tested and found nondeterministic under load.

B.6 MEASUREMENT PROTOCOL

We obtain T_{den} from the main comparison itself rather than from a separate microbenchmark. For each configuration and benchmark, we divide the GPU time spent in denoising passes, summed over the cases and three seeds at batch size one, by the number of denoising passes $\sum_i F_i$ (Appendix D), and report the unweighted mean of these per-benchmark values. The relative cost r is the ratio of a configuration's T_{den} to that of NATIVE (C=256) on the same benchmark, averaged over benchmarks, so it need not equal the ratio of the averaged T_{den}. Table 3 reports both, and Tables 10 and 12 give r

per benchmark. Because H grows with the committed length of each response, T_{den} averages over the context lengths each configuration produces, not the cost at a single fixed H, so a configuration whose responses run longer pays more per pass. This is why the two dense 512-position rows of Table 3 differ even though NATIVE (C=512) and DENSE MASK use the same input sequence shape and kernels: DENSE MASK generates shorter responses (27.3M against 30.3M tokens, 85 against 240 reaching the output ceiling), so fewer of its passes attend over a long cache, and its T_{den} is 8% lower. The paired layout saves on attention, the one component whose cost the block-causal mask reduces, while the matrix multiplications and the mixture-of-experts layers are unaffected by the mask. This is the main reason $r = 1.26$ rather than the 1.84 of a dense pass.

Table 3: **Cost of a denoising pass.** T_{den}, the GPU denoising time per denoising pass, and the relative cost r for each block layout on GB300. Both are unweighted means over the ten benchmarks of values pooled over cases and three seeds within each benchmark. A paired pass costs less than a dense pass over the same 512 positions, which sets the value of r used in the break-even criterion $\alpha > r - 1$ of Section 3.3. DENSE MASK shares the shape and kernels of NATIVE (C=512); its lower T_{den} reflects shorter responses and hence fewer long-context passes.

Layout	Mask	T_{den} (ms)	r
256 positions (NATIVE (C=256))	—	7.08	1.00
512 positions (NATIVE (C=512))	dense	13.06	1.84
512 positions (DENSE MASK)	dense	12.06	1.70
512 positions (OBA)	block-causal	8.94	1.26
256/512 positions (FLOWBLOCK (W=2))	block-causal	7.29	1.03

C PROOFS

Proposition 1 (Block-triangular isolation)**.** *Consider one denoising pass in exact arithmetic under the mask of Section 3.1, and assume every non-attention operation either acts on one position at a time or reduces over positions of x^{c} only. Let the current block enter the pass with state $(x^{\text{c}}, z^{\text{c}}, t_{\text{c}})$ and cache H, and compare it with a C-position* NATIVE *(C=256) pass from the same state and cache. Then (i) the logits over x^{c} are independent of the warm-up block state $(x^{\text{w}}, z^{\text{w}}, t_{\text{w}})$ and equal those of the* NATIVE *(C=256) pass; and (ii) if, in addition, the sampler acting on x^{c} receives the same random variates as under* NATIVE *(C=256), the resulting state $(x^{\text{c},+}, z^{\text{c},+})$ and stopping decision are equal as well.*

Proof. (i) The input representations of the current-block positions depend only on $(x^{\text{c}}, z^{\text{c}}, t_{\text{c}})$, so they are identical in the paired and the NATIVE (C=256) evaluation. Suppose they remain identical through layer ℓ. At layer $\ell + 1$, a current-block query attends to exactly the NATIVE (C=256) key set, H and x^{c}, and to no warm-up key, so its attention output is identical. Every remaining operation at that layer acts on one position at a time or reduces over x^{c} only, so it cannot introduce a dependence on x^{w}. By induction, the current-block hidden states and logits are identical at every layer and independent of the warm-up state. (ii) DENOISESTEP and the stopping test are deterministic functions of $(L^{\text{c}}, t_{\text{c}})$, the previous argmax and the random variates. By (i), identical logits and identical variates give identical $(x^{\text{c},+}, z^{\text{c},+})$ and an identical stopping decision. □

The proposition is a structural statement in exact arithmetic. Shape-specific reductions and the grouped mixture-of-experts call need not be bitwise identical across the single-block and paired layouts (Appendix B.5), so the empirical comparison uses matched prompts and seeds rather than asserting identical generated trajectories.

D ESTIMATION OF α

This appendix details how the values of α in Tables 2, 10 and 11 are obtained. We estimate α separately on each benchmark from the aggregate counters of each configuration, summed over the benchmark's cases and the three generation seeds, under the two assumptions of Section 3.3, and report the unweighted mean over benchmarks.

Denoising passes and blocks. For one benchmark and configuration, let R be the number of responses, three times the number of cases, Y the generated tokens and Q the serving forward passes. Each response takes one prompt-prefill pass and $K_i - 1$ cache-append passes (Eq. (2)), so with $K = \sum_i K_i$ the number of blocks, the denoising passes are $F = Q - R - (K - R) = Q - K$. We approximate the number of blocks from the generated tokens, counting the partially filled last block of a response as half a block on average:

$$K \approx \frac{Y}{C} + \frac{R}{2}, \qquad F = Q - K, \qquad n = \frac{F}{K}, \tag{4}$$

with $C = 256$, or $C = 512$ for NATIVE (C=512). For NATIVE (C=256), every block starts from noise, so its mean passes per block on the same benchmark is the reference $\bar{n}$ of Eq. (3).

Recurrence fit. Under Eq. (3) with constant α, a response of K_i blocks has $n_k = n^\star + (-\alpha)^{k-1}(\bar{n} - n^\star)$, and since $\bar{n} - n^\star = \alpha n^\star$,

$$\sum_{k=1}^{K_i} n_k = n^\star \Big[K_i + \frac{\alpha\big(1 - (-\alpha)^{K_i}\big)}{1 + \alpha} \Big]. \tag{5}$$

Summing over responses and neglecting $(-\alpha)^{K_i}$ gives

$$F \approx \frac{\bar{n}}{1 + \alpha} \Big[K + \frac{R\,\alpha}{1 + \alpha} \Big], \tag{6}$$

one equation in α, which we solve numerically on $[-\frac{1}{2}, 1)$ with $\bar{n}$, F and K from Eq. (4); it has a single root there for every benchmark and configuration. The second term in the bracket accounts for the first block of each response, which has no predecessor and pays $\bar{n}$, and for the transient that follows it.

Comparison with a mean over blocks. Taking $\bar{\alpha} = \bar{n}/n - 1$ with n the mean over all blocks understates α, because every first block pays $\bar{n}$; dropping the first block of each response overstates it, because the second block is promoted with $n_1 = \bar{n} > n^\star$ warm-up steps and so needs fewer than $n^\star$ passes, after which n_k oscillates toward $n^\star$. Eq. (6) models both effects. Averaged over benchmarks, OBA has $\bar{\alpha} = 0.47$ against the fitted $\alpha = 0.54$.

Relation to measured throughput. Per block, a configuration spends n denoising passes plus, on average, one prefill or cache-append pass, so its TPF is proportional to $1/(n + 1)$ at a fixed number of tokens per block. This is why the TPF ratio of OBA to NATIVE (C=256), 1.43 averaged over benchmarks, falls short of $1 + \bar{\alpha} = 1.47$: the paired layout reduces the denoising passes but not the cache-append pass of each block. TPS in Eq. (2) counts denoising time only, so on each benchmark the TPS ratio of a configuration to NATIVE (C=256) is $(\bar{n}/n)(\rho/r)$ by construction, with ρ the ratio of tokens per block Y/K; this is an identity rather than a test, since r is computed from the same pass counts. Averaged over benchmarks it gives 1.17 for OBA and 0.79 for DENSE MASK. The fixed-point prediction $(1 + \alpha)/r$ at the averaged α and r of Table 2, 1.22 and 0.81, is higher in both cases because the first block of each response pays $\bar{n}$ and gains nothing from promotion, which $\bar{\alpha}$ counts and the fixed point does not. Under both estimates, OBA clears the break-even $\alpha > r - 1$ and DENSE MASK does not.

Limitations of the estimate. The fit inherits the two assumptions of Section 3.3, a constant α and a $\bar{n}$ that does not depend on the block index. The block count of Eq. (4) is an approximation, and neglecting $(-\alpha)^{K_i}$ in Eq. (6) biases the fit on responses of few blocks; both are coarsest on GSM8K and MMLU, whose responses average about three and five blocks.

E EVALUATION PROTOCOL

Population, grading and seeds. The evaluated population is 4,736 cases: GSM8K (1,319), MATH-500 (500), MBPP (500), HumanEval (164), GPQA (198), IFEval (541), the first 1,000 MMLU examples, LiveCodeBench C++ (454), AIME 2024 (30) and AIME 2025 (30). This is the configured population, not every example in each original release. Accuracy is the fraction accepted

by a benchmark's primary grader. GSM8K, MATH-500, GPQA, MMLU and the two AIME sets use deterministic parsers or verifiers, IFEval uses strict all-instructions grading, and LiveCodeBench, MBPP and HumanEval use execution-based grading in restricted subprocesses, with grader failures and timeouts recorded separately from semantic failures. Each benchmark starts a fresh server with seed stride one and a per-benchmark base seed, and case identities, prompt-token hashes and seeds are matched across configurations. The main comparison, covering NATIVE (C=256), NATIVE (C=512), FLOWBLOCK (W=2), DENSE MASK and OBA, repeats every case under three generation seeds, obtained by adding offsets of 0, 10,000 and 20,000 to the per-case seed, for $3 \times 4{,}736 = 14{,}208$ responses per configuration and 71,040 in total. Every seed, including offset zero, is freshly generated, and within a seed a case receives the same seed under every configuration. Accuracy is single-generation accuracy, not pass@3 or majority voting, averaged over the three seeds. Separate noise streams per block keep warm-up initialization from shifting the current-block stream by construction. The serving limits are 4,096 input, 32,768 generated and 37,888 cached tokens, and a case is counted as truncated when it reaches the output or context ceiling. The ablations of Section 4.3 follow the same protocol with a single generation seed, except DENSE MASK, which is part of the three-seed comparison.

Averaging over benchmarks. Every figure reported over the ten benchmarks, including α, r and T_{den}, is an unweighted mean of per-benchmark values, so each benchmark counts equally; weighting by case count would let GSM8K and MMLU, 49% of the cases, dominate. Averaged over the ten benchmarks and three seeds, OBA scores 82.70 against 84.07 for NATIVE (C=256), a loss of 1.37 points.

Forward accounting. For case i, the serving forward passes are $Q_i = 1+(K_i-1)+F_i = F_i+K_i$: one prompt prefill, $K_i - 1$ cache appends and F_i denoising passes, where one paired $[x^{\text{c}} \mid x^{\text{w}}]$ call counts as one pass and NATIVE (C=512) has one append per 512 committed tokens. Within a benchmark, TPF divides the generated tokens $\sum_i Y_i$ by $\sum_i Q_i$ and TPS divides them by the GPU time of the $\sum_i F_i$ denoising passes, with sums over cases and seeds; the average over benchmarks is the unweighted mean of these per-benchmark ratios. The split of $\sum_i Q_i$ into denoising passes and blocks used to estimate α is given in Appendix D.

F FLOWBLOCK ADAPTATION

FlowBlock (Tian et al., 2026) decodes up to W blocks concurrently on a masked, token-editing model, admitting the next block once its predecessor is ready and repairing tokens drafted against an uncommitted prefix by editing. We keep its window, its admission rule and its in-order commitment, and replace only its model forward with DiffusionGemma's, which supplies the token transition, entropy-bounded acceptance, renoising and self-conditioning. FlowBlock defines the readiness of the frontier block as the fraction of its still-masked positions whose confidence already clears the unmasking threshold, and as one when no masked position remains (Tian et al., 2026, Eq. (2)). Under uniform-state diffusion no position is masked, so we derive the analogous quantity from DiffusionGemma's sampler: the still-masked set becomes the positions rejected, and hence renoised, at the previous step, and clearing the threshold becomes acceptance by the entropy-bounded sampler at the current step. Readiness is therefore the fraction of the preceding block's positions rejected at the previous step that are newly accepted at the current one, and one when none were rejected. Adhering to the configurations in FlowBlock, the next block is admitted once readiness exceeds 0.6 on the general and mathematical benchmarks and 0.8 on the coding benchmarks like MBPP, HumanEval and LiveCodeBench C++. As in FlowBlock, readiness is computed from the current-step logits, so a newly admitted block runs at least one pass before it can admit a successor (Tian et al., 2026, Algorithm 1). As blocks are decoded concurrently, a block keeps its tokens, self-conditioning signal and denoising time when its predecessor commits, with the same $N - 1$ inheritance cap, stopping-bookkeeping reset and post-promotion pass as OBA. At W=2, the setting of FlowBlock's main results, it therefore differs from OBA in when it starts the next block and in a pass width that switches between 256 and 512 positions. Widening the window does not help (Table 4): the readiness gate admits a successor rarely enough that more than 92% of denoising passes run a single block at every width, three blocks are live on at most 0.63% of passes, and average TPF and TPS stay within 1% of W=2.

Table 4: **FLOWBLOCK on DiffusionGemma across window widths** W, over three generation seeds. Accuracy, TPF and TPS are unweighted means over the ten benchmarks, as in Table 1. Live blocks gives the share of denoising passes that run one, two, three or four blocks. r is the cost of a denoising pass relative to NATIVE (C=256), as in Table 3.

				Live blocks (%)				
Setting	Acc. (%)	TPF	TPS	1	2	3	4	r
NATIVE (C=256)	84.07	15.64	2375	100.0	0.0	0.0	0.0	1.00
W=2	83.43	16.76	2482	92.2	7.8	0.0	0.0	1.03
W=3	83.58	16.77	2475	92.1	7.2	0.6	0.0	1.03
W=4	83.58	16.68	2463	92.2	7.2	0.4	0.2	1.03

G POSITION-LEVEL STABILITY OF PROMOTED BLOCKS

Table 5 reports the per-benchmark values plotted in Figure 2. For each benchmark, Accepted is the fraction of the 256 positions accepted on a block's first pass as the current block, and Final is the fraction whose token at that pass is unchanged at commitment. Final is a subset of Accepted, so their ratio is the fraction of first-pass acceptances that survive to commitment: 97% under OBA and 94% under NATIVE (C=256) on average.

Table 5: **Position-level stability of promoted and newly initialized blocks.** Accepted: fraction of the 256 positions accepted on a block's first pass as the current block. Final: fraction whose token at that pass is unchanged at commitment, a subset of Accepted. Fractions are computed over complete blocks and averaged over three generation seeds.

	NATIVE (C=256)		OBA	
Benchmark	Accepted (%)	Final (%)	Accepted (%)	Final (%)
GSM8K	5.2	5.0	50.9	49.8
MATH-500	5.2	5.0	34.8	33.9
MBPP	5.5	5.2	42.1	41.1
HumanEval	10.3	10.0	52.7	51.6
GPQA	3.3	3.0	37.6	36.3
IFEval	7.2	6.9	55.2	53.7
MMLU	6.8	6.6	51.8	50.5
LCB C++	3.9	3.5	30.7	29.7
AIME 2024	3.4	3.1	25.9	24.9
AIME 2025	3.5	3.2	28.4	27.4
Average	5.4	5.1	41.0	39.9

H PER-BENCHMARK RESULTS

Tables 6 and 7 give per-benchmark accuracy and wall TPS for the main comparison, and Tables 8 and 9 give the same for the inheritance policies and the attention ablation of Section 4.3; per-benchmark TPF and TPS are in Table 1. Tables 10, 11 and 12 give the per-benchmark α (Appendix D) and r (Appendix B.6); NATIVE (C=256) has $\alpha = 0$ and $r = 1$ by definition. Wall TPS divides the generated tokens by the summed server request wall time, which includes prefill, cache appends and host-side overhead but excludes model loading, idle time between requests and grading. All values are over three generation seeds, except the five inheritance policies other than KEEP ALL in Tables 8–12, which use one.

Table 6: **Per-benchmark accuracy (%) for the main comparison.**

Benchmark	NATIVE (C=256)	NATIVE (C=512)	FLOWBLOCK (W=2)	OBA
GSM8K	96.34	94.90	96.03	96.06
MATH-500	96.13	91.87	95.80	95.20
MBPP	94.87	91.07	94.87	93.80
HumanEval	97.36	96.75	96.75	96.75
GPQA	71.55	62.12	70.54	69.02
IFEval	94.09	89.28	92.67	91.44
MMLU	88.13	84.97	87.83	86.90
LCB C++	66.67	61.60	62.04	63.44
AIME 2024	74.44	62.22	74.44	71.11
AIME 2025	61.11	52.22	63.33	63.33
Average	84.07	78.70	83.43	82.70

Table 7: **Per-benchmark wall TPS for the main comparison.** Best in bold.

Benchmark	NATIVE (C=256)	NATIVE (C=512)	FLOWBLOCK (W=2)	OBA
GSM8K	2530	1971	2516	**2624**
MATH-500	2566	1427	2495	**2605**
MBPP	2296	1155	2301	**2454**
HumanEval	2690	1865	3057	**3338**
GPQA	1809	980	1895	**2216**
IFEval	2252	1684	2349	**2683**
MMLU	1862	1358	1906	**2351**
LCB C++	1956	869	1899	**2026**
AIME 2024	2026	985	2081	**2199**
AIME 2025	1974	884	2147	**2213**
Average	2196	1318	2265	**2471**

Table 8: **Per-benchmark accuracy (%) for the ablations** of Table 2. KEEP ALL is OBA.

Benchmark	KEEP ALL	DROP TIME	KEEP SIGNAL	NO ACCEPT	KEEP TOKENS	NO SIGNAL	DENSE MASK
GSM8K	96.06	96.36	96.21	95.98	95.45	96.29	95.32
MATH-500	95.20	95.20	96.20	95.40	96.60	97.20	93.20
MBPP	93.80	92.40	93.40	94.60	94.80	94.40	91.40
HumanEval	96.75	96.34	98.78	99.39	97.56	96.95	97.36
GPQA	69.02	69.70	73.74	70.71	72.73	70.20	66.50
IFEval	91.44	89.65	91.68	91.68	92.98	92.79	91.13
MMLU	86.90	86.90	88.10	87.90	87.30	88.30	87.17
LCB C++	63.44	63.66	64.54	63.88	61.45	64.98	59.69
AIME 2024	71.11	76.67	66.67	80.00	83.33	73.33	65.56
AIME 2025	63.33	60.00	70.00	70.00	53.33	66.67	55.56
Average	82.70	82.69	83.93	84.95	83.55	84.11	80.29

Table 9: **Per-benchmark wall TPS for the ablations** of Table 2. KEEP ALL is OBA. Best in bold.

Benchmark	KEEP ALL	DROP TIME	KEEP SIGNAL	NO ACCEPT	KEEP TOKENS	NO SIGNAL	DENSE MASK
GSM8K	**2624**	2484	2371	2344	2294	1996	2037
MATH-500	**2605**	2468	2341	2367	2310	2013	1853
MBPP	**2454**	2309	2063	2146	2206	1702	1698
HumanEval	**3338**	3141	3136	3099	2835	2526	2533
GPQA	**2216**	2118	1890	1816	1888	1293	1437
IFEval	**2683**	2479	2268	2329	2293	1835	2072
MMLU	**2351**	2021	1941	2068	1989	1515	1808
LCB C++	**2026**	1920	1723	1749	1785	1542	1126
AIME 2024	**2199**	2123	1968	1950	1821	1618	1498
AIME 2025	**2213**	2009	2056	1910	1798	1651	1271
Average	**2471**	2307	2176	2178	2122	1769	1733

Table 10: **Per-benchmark α and r for the main comparison.** α is undefined for NATIVE (C=512), whose blocks are twice as long as those of NATIVE (C=256).

	α		r		
Benchmark	FLOWBLOCK (W=2)	OBA	NATIVE (C=512)	FLOWBLOCK (W=2)	OBA
GSM8K	0.09	0.52	1.46	1.04	1.26
MATH-500	0.02	0.37	1.75	1.04	1.26
MBPP	0.03	0.45	2.01	1.02	1.27
HumanEval	0.22	0.73	1.67	1.02	1.24
GPQA	0.10	0.63	1.95	1.03	1.26
IFEval	0.12	0.63	1.55	1.05	1.26
MMLU	0.09	0.77	1.64	1.04	1.25
LCB C++	0.00	0.39	2.22	1.02	1.29
AIME 2024	0.07	0.42	2.00	1.03	1.25
AIME 2025	0.16	0.49	2.17	1.03	1.27
Average	0.09	0.54	1.84	1.03	1.26

Table 11: **Per-benchmark α for the ablations** of Table 2. KEEP ALL is OBA. The fit of Eq. (6) is solved on $[-\frac{1}{2}, 1)$ and so admits $\alpha < 0$, meaning a promoted block needs more denoising passes than one initialized from noise. The negative values all occur under NO SIGNAL, are at most 0.09 in magnitude, and average to $\alpha = 0.02$. They are consistent with a warm-up that saves no passes, within the approximation of the block count in Eq. (4), and possibly with tokens drafted without self-conditioning against an uncommitted predecessor costing slightly more to revise than noise.

Benchmark	KEEP ALL	DROP TIME	KEEP SIGNAL	NO ACCEPT	KEEP TOKENS	NO SIGNAL	DENSE MASK
GSM8K	0.52	0.39	0.30	0.27	0.21	-0.01	0.40
MATH-500	0.37	0.28	0.20	0.21	0.15	-0.02	0.22
MBPP	0.45	0.35	0.18	0.23	0.24	-0.07	0.34
HumanEval	0.73	0.61	0.60	0.58	0.39	0.21	0.52
GPQA	0.63	0.54	0.37	0.31	0.33	-0.09	0.44
IFEval	0.63	0.48	0.33	0.37	0.33	0.02	0.47
MMLU	0.77	0.47	0.40	0.50	0.41	0.02	0.66
LCB C++	0.39	0.30	0.16	0.18	0.17	0.00	0.22
AIME 2024	0.42	0.36	0.26	0.24	0.14	0.01	0.28
AIME 2025	0.49	0.35	0.36	0.28	0.16	0.09	0.25
Average	0.54	0.41	0.32	0.32	0.26	0.02	0.38

Table 12: **Per-benchmark r for the ablations** of Table 2. KEEP ALL is OBA.

Benchmark	KEEP ALL	DROP TIME	KEEP SIGNAL	NO ACCEPT	KEEP TOKENS	NO SIGNAL	DENSE MASK
GSM8K	1.26	1.26	1.26	1.26	1.26	1.27	1.49
MATH-500	1.26	1.26	1.27	1.26	1.26	1.27	1.64
MBPP	1.27	1.27	1.28	1.27	1.25	1.29	1.76
HumanEval	1.24	1.25	1.25	1.25	1.25	1.25	1.50
GPQA	1.26	1.25	1.27	1.27	1.25	1.29	1.79
IFEval	1.26	1.26	1.26	1.26	1.26	1.26	1.51
MMLU	1.25	1.26	1.27	1.25	1.25	1.26	1.53
LCB C++	1.29	1.28	1.29	1.29	1.27	1.28	2.14
AIME 2024	1.25	1.25	1.27	1.26	1.26	1.27	1.71
AIME 2025	1.27	1.28	1.26	1.28	1.26	1.30	1.94
Average	1.26	1.26	1.27	1.27	1.26	1.27	1.70