FIXED STATE, LONG REACH: WHAT A CONSTANT-SIZE CACHE BUYS BLOCK DIFFUSION AT SCALE

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models decode tokens in parallel, but their bidirectional denoiser rules out the naive key–value (KV) cache behind fast autoregressive inference. *Block diffusion* restores caching by decoding block-by-block, and the block caches deployed on it so far are tied to *attention*: $\mathcal{O}(L)$ in memory and, if used as training-free retrofits, only an *approximation* of the model's computation. Both constraints can be overcome: sequence mixers that summarize finalized blocks into a reusable state support block caching, and the corresponding block-causal training objective makes the cache exact. We study this recipe at scale, pretraining three 3B block-diffusion denoisers (attention, mamba, and hybrid) on 300B tokens under one single-frontier objective and decoding all three through a single cached interface. Only the state-space cache is $\mathcal{O}(1)$ in sequence length: its memory and per-step latency stay *constant* at any context length, while an attention cache remains $\mathcal{O}(L)$. At 256k tokens (where attention has grown to 82 GB and 29 ms/step), the Mamba cache runs at $\mathbf{4.3}\times$ lower latency, $\mathbf{11}\times$ less memory, and $\mathbf{2.6}\times$ higher single-stream throughput; and because that footprint is constant it scales with batch as well, reaching $\mathbf{14}\times$ the aggregate throughput, where attention cannot run beyond a single stream. The same linear-state bias lets the Mamba and hybrid backbones keep retrieving out to $\mathbf{8}$–$\mathbf{16}\times$ their training length, whereas attention's retrieval collapses at $\mathbf{2}\times$: the hybrid at no measured quality cost, the pure-SSM model for about one downstream point.

1 INTRODUCTION

Diffusion Language Models (DLMs) are an alternative to the autoregressive (AR) decoding used by most large language models (Brown et al., 2020; Achiam et al., 2023; Chowdhery et al., 2023; Touvron et al., 2023). Masked DLMs such as LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) corrupt a sequence by masking tokens and learn to denoise it, decoding many positions in parallel rather than one token at a time. This promises lower latency and richer per-example supervision (Ni et al., 2025). In practice, however, DLM inference has often been *slower* than AR: the very bidirectional attention that makes denoising possible rules out the naive key–value (KV) cache that AR decoders rely on, so each denoising step recomputes attention over the entire sequence (Wu et al., 2026; Ma et al., 2025).

Block diffusion (Arriola et al., 2025) interpolates between the two regimes: the sequence is partitioned into blocks that are generated autoregressively, while diffusion denoising runs *within* each block. Because finalized blocks form a clean prefix, their representations can be cached and reused, restoring KV-caching and enabling arbitrary-length generation. This has made block diffusion the foundation of recent fast-decoding systems (Wu et al., 2026; 2025; Wang et al., 2025b).

Two limitations remain. First, the block caches deployed so far are built for an attention backbone (Wu et al., 2026; Ma et al., 2025; Liu et al., 2025b; Nguyen-Tri et al., 2025); the resulting KV-cache still grows linearly in context length and the underlying attention still costs quadratic compute, so the long-context regime stays expensive. Second, many of these methods are *training-free* adaptations onto full-attention models, which makes their block cache an *approximation* of the true bidirectional computation (Wu et al., 2026). Neither limitation is intrinsic to block diffusion, which only requires that finalized blocks summarize into a reusable state: any sequence mixer with such a state qualifies, and training with the block-causal objective makes the resulting cache exact rather than approximate (Arriola et al., 2025).

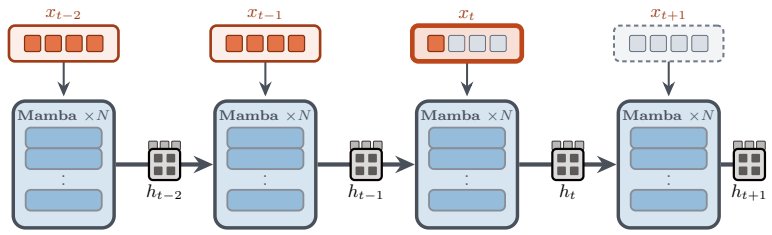
Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping 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.

State-space models (SSMs) such as Mamba (Gu & Dao, 2024; Dao & Gu, 2024) are exactly such mixers. They process a sequence with a linear-time recurrence whose hidden state is a fixed-size summary of the past, independent of how long the past is. We build on the bidirectional-Mamba DLM denoiser of DiffuMamba (Singh et al., 2026) and use its recurrent state as the block cache: when block diffusion finalizes a block, the forward SSM state already encodes the entire prefix in $\mathcal{O}(1)$ memory. Concurrent work (Chaturvedi et al., 2026) has recently shown, at small scale (87M–350M parameters), that a Mamba–attention block-diffusion hybrid admits exactly such a cache once the reverse Mamba scan is confined to the active block. Whether this holds when pretrained at scale, how a *pure* Mamba denoiser fares, and how a constant-size state behaves on long-context *retrieval* rather than throughput alone, have not been studied. We therefore expose the SSM state together with the attention KV-cache and a hybrid of the two through a single decoding interface, and we *pretrain* all three models at 3B parameters under the same block-causal objective, so that cached decoding reproduces training exactly.

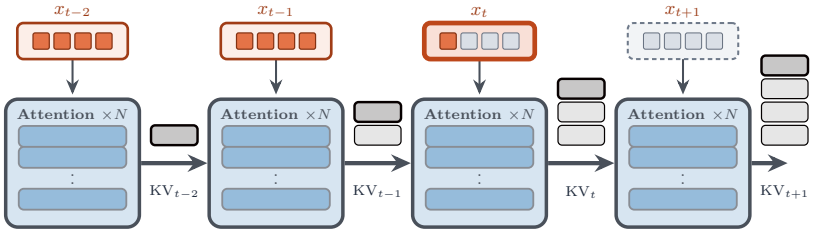
Concretely, we pretrain and analyze three 3B-parameter block-diffusion models on 300B tokens that share data, tokenizer, schedule, and decoding budget and differ only in the denoiser: `Attn` (full attention), `Mamba` (bidirectional Mamba-2), and `Hybrid` (attention interleaved every five Mamba layers). Because the Mamba mixer carries larger projections while the MLP is held fixed, `Mamba` and `Hybrid` technically have 13% and 11% more parameters than `Attn`; Section 5.4 shows that these extra parameters add a fixed $\mathcal{O}(d^2)$ cost per token at every context length, whereas attention's score computation costs $\mathcal{O}(Ld)$ per token and grows with context. Per-token FLOPs are comparable at the training length, and the long-context gap is due to the mixer rather than the parameter budget. Our contributions are:

- **A controlled 3B pretraining study of cacheable block-diffusion denoisers** (Sections 3 and 4): attention, bidirectional Mamba-2, and hybrid backbones, each pretrained on 300B tokens under the same single-frontier block-causal objective. As in BD3LM (Arriola et al., 2025) for attention and, concurrently, Chaturvedi et al. (2026) for Mamba hybrids, the cache is exact because the models are trained with the objective used at decode time; unlike training-free retrofits, cached inference *is* the function the model learned. To our knowledge this is the largest such study, and the only one that includes a *pure* state-space denoiser.
- **Constant memory and long context from a single linear-state backbone** (Sections 5.1 and 5.2). The SSM cache is $\mathcal{O}(1)$ in length: `Mamba` holds $\sim$7.5 GB and 6.8 ms/step at *every* length out to 256k, against attention's 82 GB, reaching $\mathbf{14}\times$ the aggregate throughput at 256k. The same backbone also generalizes far past its 1024-token training length: `Attn` falls behind at just $2\times$ on NIAH and LongBench, whereas `Mamba`/`Hybrid` extrapolate to $\mathbf{8}$–$\mathbf{16}\times$.
- **Long-context evaluation at scale** (Section 5). With everything but the denoiser held fixed, we cover cached-decoding efficiency (latency, memory, and throughput) out to 256k tokens, NIAH and LongBench length extrapolation, an eight-task downstream suite, MAUVE and generative perplexity, and a per-layer FLOPs analysis that attributes the gains to the *architecture* rather than to parameter count.

None of these properties is new to state-space models. Fixed-size state (Gu & Dao, 2024; Dao & Gu, 2024) and length extrapolation (De et al., 2024; Ren et al., 2025) are established properties of *autoregressive* SSMs, with hybrids restoring the recall that pure recurrence lacks (Waleffe et al., 2024; Lieber et al., 2024). Whether they carry over to diffusion denoisers, which decode several tokens per pass, is open. Our results are the block-diffusion counterpart of that AR literature, and the combination matters because block diffusion reveals several tokens per forward pass rather than one, and a linear-state backbone lets it do so on a state that never grows. Singh et al. (2026) showed Mamba denoisers match attention on quality at the training length, and Chaturvedi et al. (2026) that they can be block-cached like attention (Arriola et al., 2025) at small scale; we show that the same backbones, pretrained at 3B, *decode* long contexts with a constant-memory cache and retrieve far past their training length, which extends the earlier training-time case for linear-state diffusion LMs to inference and to long-context retrieval.



(a) `Mamba` (and the Mamba layers of `Hybrid`): a single recurrent state h of fixed shape $\mathbb{R}^{H\times P\times N}$, carried and updated in place ($h_t = \bar{A}_t\, h_{t-1} + \bar{B}_t\, x_t$), $\mathcal{O}(1)$ in sequence length.



(b) `Attn`: the key–value memory, which appends one entry per block ($\mathrm{KV}_{t-2} \to \mathrm{KV}_{t+1}$) and therefore grows as $\mathcal{O}(L)$.

Figure 1: **The unified block cache.** Block diffusion decodes one block at a time, left to right: finalized blocks (x_{t-2}, x_{t-1}) are clean, the frontier block x_t is being denoised (its first token is revealed, the rest are `[MASK]`), and future blocks (x_{t+1}) stay fully masked. Once a block is finalized, every layer writes its state into a cache that the next block reads, so the clean prefix is never recomputed. The cached object depends on the backbone.

2 Background and Related Work

Masked diffusion language models. Discrete diffusion models define a forward process that progressively corrupts a token sequence and learn a parameterized reverse process that denoises (Austin et al., 2021; Campbell et al., 2022). The *masked* (absorbing-state) instantiation has proven the most effective for language: SEDD (Lou et al., 2024), MDLM (Sahoo et al., 2024), and their refinements (Shi et al., 2024; Sahoo et al., 2025) train a bidirectional network to recover masked tokens under a noise schedule, and LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) scale this recipe to billions of parameters. All decode by iteratively unmasking positions in parallel; none admits a KV-cache natively, because every denoising step attends over the full (partially masked) sequence in both directions.

Block diffusion. BD3LM (Arriola et al., 2025) interpolates between AR and diffusion: tokens are grouped into blocks generated left-to-right, with intra-block diffusion. Blocks already produced act as a clean prefix, so their keys and values can be cached between blocks, and sequences of arbitrary length can be generated. Our attention model follows this scheme; we use the name `Attn` rather than "BD3LM" to keep our trained model distinct from the cited training recipe. Subsequent systems build fast decoders on this foundation (Wu et al., 2026; 2025; Wang et al., 2025b).

Caching for diffusion LMs. Because the bidirectional denoiser blocks the standard KV-cache, a line of work restores caching for attention DLMs. Fast-dLLM (Wu et al., 2026) introduces a block-wise *approximate* KV-cache plus confidence-aware parallel decoding, reporting large speedups on LLaDA/Dream by caching the fixed context and refreshing it at block boundaries; dKV-Cache (Ma et al., 2025) proposes a delayed KV-cache compatible with bidirectional attention; and dLLM-Cache (Liu et al., 2025b) and elastic-cache variants (Nguyen-Tri et al., 2025) add adaptive eviction and reuse. More recent work targets the *long-context* regime specifically: Prefilling-dLLM (Xiong et al., 2026) caches the prefix KV in chunks and decodes against the top-K relevant ones, and Focus-dLLM (Long et al., 2026) restricts attention to a confidence-selected context; both report 9–30$\times$ speedups at 8–32k. A separate line extends the *usable* context of attention DLMs by rescaling

Algorithm 1 Cached block-wise generation

Require: prefix, block size G, steps per block S, blocks K, mask id
warm $cache$ on the clean prompt blocks
for $i \leftarrow i_{\text{first}}$ **to** $K-1$ **do**
  $s \leftarrow \textsc{Clone}(cache)$ ▷ entry state, reused across denoising steps
  $m \leftarrow$ # masked tokens in block i
  for $t \leftarrow 1$ **to** S **do**
    $\ell \leftarrow \textsc{ForwardBlockCached}(x_i,\ s,\ \text{pos}_i)$
    reveal the $\lceil m/S \rceil$ highest-confidence masked positions of x_i using ℓ
    if block i fully revealed **then**
      break
    end if
  end for
  $_,\ cache \leftarrow \textsc{ForwardBlockCached}(x_i,\ s,\ \text{pos}_i)$ ▷ fold finalized block into cache
end for
return x

positional encodings: LongLLaDA (Liu et al., 2025a) applies training-free NTK-RoPE (and documents a "local perception" sliding-window bias), while UltraLLaDA (He et al., 2025) *trains* to a 128k window. All of these caches are attention-only and $\mathcal{O}(L)$ in memory, and, being largely training-free retrofits, only *approximate* the full-attention computation they replace. The exception is training-time block-causal schemes, BD3LM (Arriola et al., 2025) for attention and, concurrently with this work, Chaturvedi et al. (2026) for Mamba–attention hybrids, whose caches are exact because the model is trained on the same clean-prefix computation it decodes with; our models belong to this family.

State-space models and hybrids. SSMs model sequences with a linear-time selective recurrence: S4 (Gu et al., 2022), Mamba (Gu & Dao, 2024), and Mamba-2 (Dao & Gu, 2024) match or beat attention on language while carrying a fixed-size state, and related structured operators (Poli et al., 2023; Fu et al., 2023) share the linear-time property. Hybrid AR models interleave attention with linear recurrence to combine global recall with cheap long-range mixing (Lieber et al., 2024; De et al., 2024; Wang et al., 2025a). DiffuMamba (Singh et al., 2026) brought bidirectional Mamba and Mamba–attention hybrids to *diffusion* denoisers. Chaturvedi et al. (2026) trained block cache diffusion models at a small scale with Mamba–attention hybrids of 87M and 350M parameters.

Relation to concurrent work and positioning. Three properties distinguish the caches studied here from the retrofit line above. They are *architecture-general*: attention, state-space, and hybrid denoisers sit behind one interface. They are *exact rather than approximate*: the models are trained with the same block-causal single-frontier objective used at decode time, so the cached forward pass *is* the function the model learned. And the state-space cache is *constant-memory*, $\mathcal{O}(1)$ in sequence length, whereas every attention cache, ours included, is $\mathcal{O}(L)$. None of these is a claim of priority. Exactness for attention follows from block-causal training as in BD3LM (Arriola et al., 2025), and the block-restricted reverse scan that makes a bidirectional-Mamba layer cacheable was proposed concurrently and independently by Chaturvedi et al. (2026), who train 87M and 350M Mamba–attention hybrids on DCLM with an all-block objective and report decoding throughput on randomly initialized 700M models. What this study adds is scale and evidence: pretraining at 3B parameters and 300B tokens with all three backbones, including a *pure* Mamba denoiser absent from that work; a single-frontier objective; latency, memory, and throughput measured single-stream and batched; and, most importantly, long-context *retrieval* and length extrapolation (NIAH, LongBench) alongside downstream and generation quality. Our claim therefore concerns the behavior of the mechanism at scale: constant-memory block decoding holds after pretraining at 3B, and the linear-state models retrieve over far longer contexts than the attention baseline.

3 Method

3.1 Block-diffusion preliminaries

Let $x = (x_1, \ldots, x_L)$ be a token sequence. A masked diffusion model defines a forward process that, at time $t \in (0, 1]$, independently replaces each token by a special `[MASK]` symbol with probability

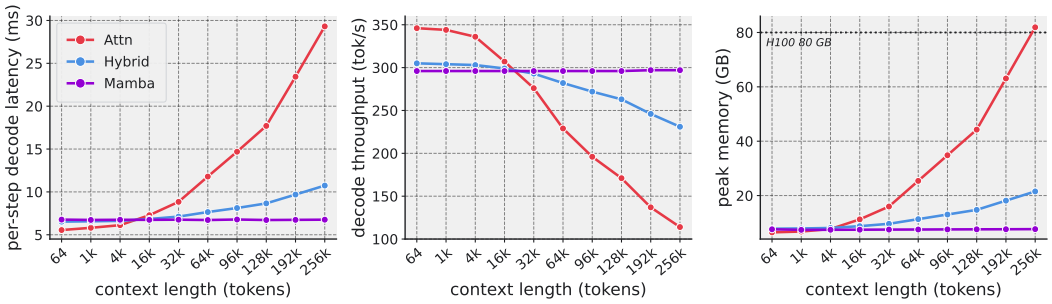


Figure 2: **Cached-decoding efficiency vs. context length (3B, single H100 80 GB, `bf16`, cudagraph, batch 1).** (a) per-step decode latency, (b) end-to-end decode throughput (S=16), (c) peak memory. `Mamba` is flat at every length to 256k (constant SSM state), while `Attn` grows as $\mathcal{O}(L)$: by 256k `Mamba` is $\mathbf{2.6}\times$ faster (297 vs. 114 tok/s) at $\mathbf{11}\times$ less memory (7.5 vs. 82 GB); `Hybrid` interpolates. Figure 4 sweeps batch 1–8.

$1-\alpha_t$, where α_t follows a monotone schedule. The denoiser f_θ is trained to predict the original tokens from the masked sequence, minimizing a re-weighted cross-entropy that forms a negative ELBO on the data log-likelihood (Sahoo et al., 2024; Nie et al., 2025).

Block diffusion factorizes the sequence into $K = L/G$ contiguous blocks of size G and is autoregressive across blocks: block i is generated conditioned on the clean blocks $0, \ldots, i{-}1$, while the tokens *inside* block i are produced by diffusion. We train with a **single-frontier** objective (Figure 1): for each example we sample one frontier block i, keep blocks $< i$ clean, mask a time-dependent random subset of block i, and mask blocks $> i$ *entirely*; the loss is the cross-entropy on the masked positions of block i only (uniform weights, Section A). This exactly matches the information available during cached generation, where block i sees a clean prefix and all-masked future. The frontier structure is enforced differently per backbone. *Attention* layers use a **block-causal** mask: a query in block i attends to all keys in blocks $\leq i$ (bidirectionally within its own block) and to none in blocks $> i$. *Mamba* layers run a forward (left-to-right) recurrence over the whole prefix, and a backward (right-to-left) recurrence whose support is *restricted to the frontier block*, so that no information leaks backward from the masked future (the block-restricted reverse scan that Chaturvedi et al. (2026) term *partial bidirectionality*). The hybrid applies both rules, layer by layer.

3.2 A UNIFIED BLOCK CACHE

At inference the blocks are produced strictly left-to-right, so once block i is finalized its contribution to every future block is fixed. We expose a single per-layer interface that consumes only the G tokens of the current block plus a per-layer cache summarizing blocks $0, \ldots, i{-}1$, and returns the block logits and an updated cache. The cache type is backbone-specific (Table 1). In the attention layers, for the G new tokens, we compute queries, keys, and values, and then concatenate the new keys/values with the cached (K, V) of all finalized blocks. The attended context is the clean prefix plus the current block; the cache grows as $\mathcal{O}(L)$ and the per-step cost is $\mathcal{O}(GL)$.

A Mamba-2 layer maintains two states that together summarize the prefix: a short causal-convolution state (the last $k{-}1$ inputs, where k is the conv width) and the selective-SSM recurrent state of shape $(\text{heads} \times \text{head_dim} \times d_{\text{state}})$. The forward recurrence consumes the cached states, emits the block output, and returns updated states via a chunked scan with initial states; both states are *independent of* L. The backward (bidirectional) recurrence is re-run from a zero state *within the current block only*, reproducing the frontier restriction used in training; it is never carried across blocks. Thus the Mamba cache is $\mathcal{O}(1)$ in length and the per-step cost is $\mathcal{O}(G^2)$, independent of how much text precedes the block. Lastly the hybrid denoiser interleaves the two layer types, so its cache is a heterogeneous list: attention layers store (K, V) tensors and Mamba layers store $(\text{conv}, \text{ssm})$ states. The overall memory is dominated by the (few) attention layers, giving a cost between those of the two pure backbones.

3.3 INFERENCE VIA CACHED GENERATION

Generation proceeds block by block (Algorithm 1). Entering block i, we clone the layer cache so its entry state can be reused across the block's denoising steps (each cached forward returns fresh tensors and never mutates its inputs). We then run S denoising steps: at each step a cached forward

Table 1: **The three denoisers differ only in the sequence mixer.** The 3B models share d_{model}, depth, block size G, and differ only in the sequence mixer. The right block gives each backbone's per-step decode cost over a block of G tokens against a prefix of length L: only the SSM cache is $\mathcal{O}(1)$ in L (conv+SSM state vs. an $\mathcal{O}(L)$ KV-cache), the root of `Mamba`'s flat memory and latency; *small constant ($5/28$ layers carry attention).

	Configuration						Per-step cost (block G, prefix L)		
Model	d_{model}	**Layers**	**Mixer**	G	**Train ctx.**	**best val. NLL**	**Compute**	**Cache mem**	**Throughput**
`Attn`	2560	28	28 attn	32	1024	1.104	$\mathcal{O}(GL)$	$\mathcal{O}(L)$	$\mathcal{O}(1/L)$
`Hybrid`	2560	28	5 attn + 23 Mamba	32	1024	1.102	$\mathcal{O}(GL)^*$	$\mathcal{O}(L)^*$	$\mathcal{O}(1/L)^*$
`Mamba`	2560	28	28 bidir. Mamba-2	32	1024	1.134	$\mathcal{O}(G^2)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

pass scores the current block, and we reveal the $\lceil m/S \rceil$ highest-confidence still-masked positions, where m is the number of masked tokens at block entry. Once the block is fully revealed, one final cached pass folds it into the cache that block $i{+}1$ reads. With S the only quality–speed knob, the total number of forward passes is $\mathcal{O}(KS) = \mathcal{O}((L/G)S)$, and for the SSM backbone each pass touches only the G current tokens regardless of L.

3.4 Complexity and asymptotic throughput

Table 1 summarizes the cost of one cached denoising step and the state it carries. Decoding the frontier block of G tokens against a prefix of length L costs $\mathcal{O}(GL)$ for attention – each new query attends over the growing KV-cache – so its per-step latency grows linearly in L and the decode throughput scales as $T = \mathcal{O}(1/L)$. The Mamba model consumes a fixed-size SSM state and touches only the G current tokens, giving an L-independent per-step cost of $\mathcal{O}(G^2)$ and hence constant throughput. The hybrid inherits the attention scaling with a small constant ($5/28$ of its layers carry attention). Section 5.1 measures exactly these three curves.

4 Experimental Setup

Models. We compare three 3B-parameter denoisers that are identical except for the sequence mixer (Table 1): `Attn` (28 attention layers), `Mamba` (28 bidirectional Mamba-2 layers), and `Hybrid` (5 attention layers interleaved with 23 Mamba layers, following Singh et al. (2026)). All models use $d_{\text{model}} = 2560$, block size $G = 32$ and are trained on Nemotron-CC (Su et al., 2024) with a 300B-token budget under the single-frontier block-diffusion objective (Section 3.1) which masks a random subset of a single sampled frontier block per example, keeping earlier blocks clean and later blocks fully masked, so training sees exactly the clean-prefix, all-masked-future context each block meets at cached decode time. We trained with sequence length of 1024 tokens, and the global batch being 4096 sequences ($1024 \times 4096 \approx 4.2$M tokens per step) with a 71 525-step schedule covering $1024 \times 4096 \times 71\,525 \approx 300$B tokens. Optimization is AdamW in `bf16` with a cosine schedule, 5% warmup, weight decay 0.1, peak learning rate 1×10^{-4} (5×10^{-5} for `Mamba`) and gradient-norm clip 1.0 (0.5 for `Mamba`); the complete set of training and model hyperparameters is in Section A.

Hardware and timing. All efficiency measurements use a single NVIDIA H100 80 GB in `bf16` with PyTorch SDPA attention kernels. We time one denoising step by CUDA-graph capture which removes kernel launch overhead, and report the median of 30 replays under CUDA events. End-to-end decode throughput is the ratio of total tokens generated to the total time to decode them, adding up each block's latency at its true cache depth (block i is decoded against a prefix of length $i \cdot G$).

Evaluation protocols. We measured *Efficiency* by sweeping context length L from 64 to 256k tokens at batch sizes 1–8, and report per-step decode latency, peak memory, and end-to-end decode throughput at $S{=}16$ denoising steps per block. For *Long Context Performance* we measure retrieval with a RULER-style (Hsieh et al., 2024) needle-in-a-haystack (NIAH) passkey task, where a short key is hidden at one of five depths inside a long distractor context and must be recovered (20 trials per (length, depth), L from 512 to 16k). We further evaluated the models on LongBench (Bai et al., 2024), a 16-task English-and-code long-context suite (middle-truncated to L, 20 examples per task, L from 2k to 16k). For the attention backbones we additionally report an NTK-RoPE variant (Peng et al., 2024; Liu et al., 2025a): a training-free rescaling of RoPE's rotation frequencies that folds the longer positions seen at inference back toward the range covered during training (the identity for

Table 2: **NIAH retrieval vs. context length: attention fails at $2\times$ the training length, the linear-state models extrapolate to 8–$16\times$.** Needle-in-a-haystack passkey retrieval accuracy (%) vs. context length L (3B; trained at L=1024, so $L\geq$2k is extrapolation). "+NTK" adds NTK-RoPE to the attention-bearing models (identity for $L \leq$ 4k); best per length shaded (all models are at ceiling for $L \leq$ 1k, left unshaded). `Attn` drops to 12% at $2\times$ and 0% by $8\times$, while `Mamba` (and `Hybrid` with NTK) still retrieves needles at $16\times$.

L (= ×train)	512	1k (1×)	2k (2×)	4k (4×)	8k (8×)	16k (16×)
`Attn`	100	100	12.3	1.7	0	0
`Attn`+NTK	100	100	12.3	1.7	0	0
`Hybrid`	98.7	100	53.4	37.9	23.0	2.3
`Hybrid`+NTK	98.7	100	53.4	37.9	26.1	14.3
`Mamba`	99.0	99.0	75.7	32.3	23.6	22.3

$L \leq$ 4k, where the rows coincide). For *Downstream Performance* we score eight common-sense / reasoning tasks with the block-diffusion likelihood harness, and for evaluating *Generation Quality* we report generative perplexity (scored by GPT-2 large) alongside MAUVE (Pillutla et al., 2021), which quantifies the distributional gap between generated and reference text.

5 Results

5.1 Efficiency: the state-space cache is constant in length

Figure 2 and Table 8 report per-step latency, peak memory, and decode throughput against context length at batch 1. The Mamba block cache is $\mathcal{O}(1)$ across the whole 64–256k-token range: `Mamba` holds a *flat* 6.8 ms per step, 7.5 GB of memory, and ∼296 tok/s, because its state does not grow with the prefix. `Attn`'s $\mathcal{O}(L)$ KV-cache shows in every panel: from 64 to 256k tokens its latency rises $5.5 \rightarrow 29$ ms ($5.3\times$), its decoding throughput falls $346 \rightarrow 114$ tok/s ($3\times$), and its memory grows $6.5 \rightarrow 82$ GB, close to the device limit. At 256k, `Mamba` is therefore ahead of `Attn` on all three measures: $\mathbf{4.3}\times$ lower latency (6.8 vs. 29 ms), $\mathbf{11}\times$ less memory (7.5 vs. 82 GB), and $\mathbf{2.6}\times$ higher throughput (297 vs. 114 tok/s). `Hybrid` lies between the two: with only 5 of its 28 layers attentional it follows the attention curves at $5/28$ of the slope, so at 256k it retains attention for global recall while still improving on `Attn` by $\mathbf{2.7}\times$ in latency (10.7 vs. 29 ms), $\mathbf{3.8}\times$ in memory (21.5 vs. 82 GB), and $\mathbf{2.0}\times$ in throughput (231 vs. 114 tok/s).

The constant footprint matters most under batching (Figure 4): because `Mamba`'s per-stream memory does not grow, many sequences fit on one accelerator, and at 256k it sustains 1593 tok/s at batch 8 (a flat ∼8 GB), whereas `Attn`'s $\mathcal{O}(B \cdot L)$ KV-cache is already out of memory at batch 2 and limited to 114 tok/s, a $\mathbf{14}\times$ difference in aggregate throughput (Appendix C). The advantage is a long-context effect: up to 16k `Attn` is 4–17% ahead of `Mamba` in throughput on its cheap short-context blocks, and the backbones separate only at tens to hundreds of thousands of tokens, where `Mamba` and `Hybrid` lead on latency, memory, and throughput.

5.2 Long-context generalization beyond the training length

The recurrence that makes the SSM cache cheap carries no positions, which also makes it robust to length. The mechanism is the block-causal mask: regardless of block size, a query still attends over the *entire* clean prefix, so past L=1024 the attention layers meet absolute RoPE positions and prefix lengths never seen in training (an out-of-distribution regime), whereas the recurrent state carries no positions and simply folds a longer prefix into the same fixed-size memory. Table 2 shows this on NIAH passkey retrieval. Within the 1024 training length all three models retrieve near-perfectly; at $2\times$ (L=2k) `Attn` drops to 12% while `Hybrid` retains 53% and `Mamba` 76%, and at $16\times$ (L=16k) `Attn` is at 0% while `Mamba` still recovers 22% of needles. `Mamba` is best at the two ends of the range ($2\times$ and $16\times$) and `Hybrid` in the middle ($4\times$, and $8\times$ with NTK-RoPE).

The same pattern holds on *realistic* tasks (Table 3). `Mamba` is the strongest backbone and degrades little with length, staying at 10.2–11.2 from 2k to 16k and ahead of `Hybrid` ($10.8 \rightarrow 9.1$) at every length. `Attn` falls by a third by 4k ($7.1 \rightarrow 4.7$) and roughly halves by 16k ($\rightarrow 3.6$), so the gap widens with context: at 16k `Mamba` scores $2.8\times$ and `Hybrid` $2.5\times$ the `Attn` average (10.2 and 9.1 vs. 3.6), as expected of a recurrence that carries no positions and needs no length extrapolation.

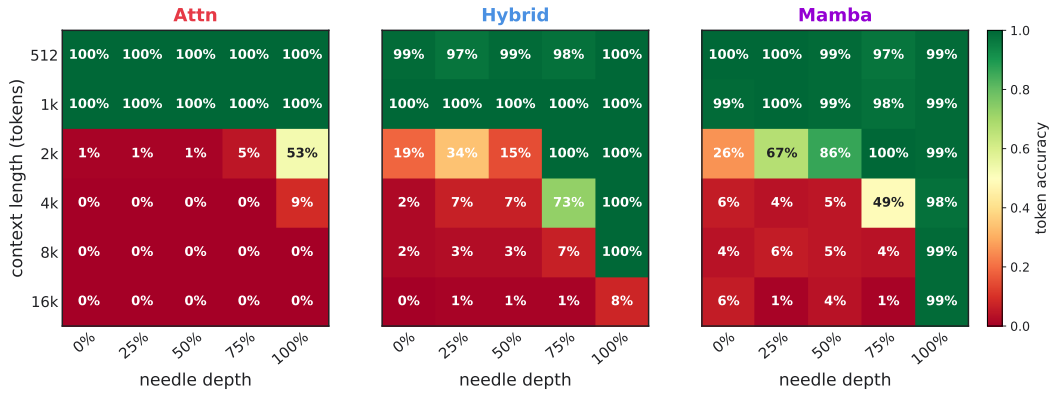


Figure 3: **NIAH retrieval by needle depth × context length (3B).** Token accuracy at five depths (0% = oldest, 100% = most recent) across lengths. `Attn` falls almost uniformly past the 1024-token training length; `Mamba`/`Hybrid` degrade *gracefully* and retain near-perfect retrieval of *recent* needles (rightmost column: `Mamba` 99% at 16× training length; per-depth numbers in Section B). This matches the "local perception" / sliding-window behavior reported for diffusion LMs by LongLLaDA (Liu et al., 2025a), which appears here as a property of the linear-state backbone rather than of an approximate cache.

Table 3: **LongBench vs. context length: the linear-state models stay flat, attention degrades.** LongBench macro-average (%) over 16 English+code tasks vs. length L (3B, cached greedy decoding, middle-truncation, 20 examples/task; every row $L \geq 2$k is extrapolation from the 1024 training length). "+NTK" = NTK-RoPE on the attention-bearing models; best per length shaded. `Mamba` holds 10.2–11.2 throughout; `Attn` drops by a third by 4k and reaches 3.6 at 16k.

L	Attn	Attn+NTK	Hybrid	Hybrid+NTK	Mamba
2k	7.07	7.07	10.83	10.83	11.20
4k	4.65	4.65	10.39	10.39	10.45
8k	3.96	4.66	9.30	9.83	10.46
16k	3.64	4.75	9.13	8.10	10.18

Table 4: **Downstream accuracy: `Hybrid` ties `Attn`, `Mamba` trails by ∼1 point.** Accuracy on eight common-sense/reasoning tasks (3B) via the block-diffusion likelihood harness (block-decomposed MC-ELBO, 128 samples, CFG 0.5); **Macro** is the mean, best shaded. `Attn` and `Hybrid` tie (0.434/0.432) and `Mamba` trails by ∼1 point.

Model	PIQA	ARC-c	HSwag	OBQA	BoolQ	SciQ	CSQA	SIQA	Macro
`Attn`	0.655	0.226	0.347	0.210	0.616	0.829	0.188	0.400	**0.434**
`Hybrid`	0.663	0.224	0.339	0.214	0.621	0.815	0.187	0.392	0.432
`Mamba`	0.658	0.230	0.334	0.194	0.581	0.778	0.198	0.392	0.421

5.3 Quality at the training length

The efficiency and long-context gains come with little change in quality. The three models sit within ∼0.03 nats on validation NLL, with `Hybrid` best (Table 1), and on an eight-task downstream suite scored by the block-diffusion likelihood harness (Table 4) `Attn` and `Hybrid` are level (macro 0.434 vs. 0.432) while `Mamba` trails by about a point. The hybrid therefore reaches its 8–16× length generalization and much lower decode memory with no measured loss in quality, and the pure-SSM `Mamba` is ∼1 downstream point lower in exchange for constant-memory, length-independent inference.

Denoising budget. The per-block denoising budget S trades quality for speed (Section D). For all three backbones, raising S from 4 to 32 *monotonically* lowers generative perplexity (15.3 → 6.4 for `Attn`, 17.1 → 6.5 for `Hybrid`, 18.5 → 6.5 for `Mamba`) and broadly improves MAUVE, at 5–6× lower throughput, so the operating point can be chosen per deployment (MAUVE is low and noisy for these small *base* models, so Gen-PPL is the more reliable measure).

Table 5: **Parameters and forward FLOPs/token at short and long context.** Exact parameters and analytic forward FLOPs/token for the 3B models (Section 5.4). `Mamba` and `Hybrid` carry 11–13% *more* parameters than `Attn`, but these live in L-independent $\mathcal{O}(d^2)$ projections that add no long-context cost: per-token FLOPs are comparable at short context, while only the SSM cost stays flat with length (shaded), so at L=64k `Mamba` spends $0.26\times$ the attention FLOPs while `Attn`'s $\mathcal{O}(Ld)$ term dominates.

Model	Params	FLOPs/tok (L=1024)	FLOPs/tok (L=64k)
`Attn`	3.033 B	5.712 G	24.209 G
`Hybrid`	3.360 B (+11%)	6.139 G	9.442 G
`Mamba`	3.431 B (+13%)	6.232 G	6.232 G

5.4 Parameter count versus architecture

`Mamba` and `Hybrid` have more parameters than `Attn`, so one could ask whether the extra parameters, rather than the linear-time mixing, explain the results. Table 5 shows that they do not. `Mamba` and `Hybrid` do carry 13% and 11% *more* parameters than `Attn`, but those parameters live in $\mathcal{O}(d^2)$ projections that are independent of L and add no long-context cost: their forward FLOPs/token stay flat with context (`Mamba` at 6.2 G for *every* length), whereas attention's parameter-free $\mathcal{O}(Ld)$ score term grows to 18.8 G, taking its per-token total to 24.2 G. The three are comparable at the 1024 training length, then diverge sharply: at L=64k `Mamba` and `Hybrid` spend just $0.26\times$ and $0.39\times$ the attention FLOPs. With quality already at parity (Section 5.3), the long-context advantage comes from the linear-time mixer rather than from the parameter budget.

We calculate forward-pass FLOPs per token *analytically* from the architecture (two FLOPs per multiply–accumulate), summing every layer's mixer and gated FFN plus the untied output head over the vocabulary; kernels are not profiled. Per token, an attention layer costs $8d^2$ for the Q, K, V, O projections plus $4Ld$ for the score and value-aggregation matmuls, the *only* L-dependent term. A bidirectional Mamba-2 layer costs

$$2\big[\, 2d(2d_{\text{in}}{+}2d_{\text{state}}{+}n_h) + 2d_{\text{in}}d + 2d_{\text{in}}d_{\text{state}} + 2d_{\text{conv}}(d_{\text{in}}{+}2d_{\text{state}}) \,\big]$$

with $d_{\text{in}}{=}d$ and n_h Mamba heads (in/out projections, the SSD scan, and the depthwise convolution, doubled for the two directions), all independent of L; the output head adds $2dV$. Unlike the smaller-scale DiffuMamba hybrids (Singh et al., 2026), the 3B Mamba/hybrid variants keep the *same* MLP as attention, so their extra mixer parameters are not offset. Table 5 evaluates this forward count for the 3B models (d=2560, 28 layers, hybrid = 5 attention + 23 Mamba, gated FFN with hidden size 7680, d_{state}=64, expand 1, d_{conv}=4, and an untied vocabulary of V=126 464) at L=1024 and L=64k; parameter counts are read exactly from the trained checkpoints. The formula also accounts for every entry of Table 5. At L=1024 the attention score term summed over `Attn`'s 28 layers is only $28 \cdot 4Ld \approx 0.29$ G, about 5% of its 5.7 G total, which is why the three backbones cost nearly the same at the training length. At L=64k the same term grows to 18.8 G and dominates. `Hybrid` sits in between: its 5 attention layers add exactly $5 \cdot 4(L{-}1024)d \approx 3.3$ G going from 1024 to 64k tokens, taking it from 6.1 to 9.4 G, while `Mamba`'s cost does not move at all (6.2 G at every length). The long-context compute gap is therefore set by the number of attention layers rather than by the parameter count: with 5 of 28 layers attending, `Hybrid` spends $0.39\times$ the attention FLOPs at 64k, against `Mamba`'s $0.26\times$ with none.

6 Conclusion

We pretrained attention, bidirectional-Mamba, and hybrid block-diffusion denoisers at 3B parameters and 300B tokens under one block-causal frontier objective, and decoded all three through a single, *exact* cached interface whose state-space variant is $\mathcal{O}(1)$ in sequence length. Concurrent small-scale work (Chaturvedi et al., 2026) showed that such a cache is possible; our results show how it behaves at scale. At 256k tokens `Mamba` decodes at a flat 6.8 ms/step and 7.5 GB where `Attn` needs 29 ms/step and 82 GB, and because that per-stream footprint never grows it scales with batch to $14\times$ the aggregate throughput. The same linear-state bias makes decoding length-robust: `Mamba` and `Hybrid` retrieve needles and answer LongBench queries at 8–16$\times$ their training length while `Attn`'s retrieval fails at $2\times$, and the hybrid matches `Attn` on likelihood and downstream accuracy, so these gains come at no measured quality cost.

AI USE STATEMENT

In this work, we used generative AI tools for polishing and refining the writing of the manuscript to prevent any inadvertent grammatical mistakes. We have not used generative AI tools to generate synthetic data sets, to formulate mathematical claims or to prove them, to assist with translation, or to clean or reformat datasets, and the remaining required-disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work: all text was revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The training objective and the cached decoding procedure are specified in Section 3.1 and Algorithm 1; the complete training and architecture configuration of all three 3B models is in Section A; the FLOP accounting is derived in closed form in Section 5.4; and the full per-depth NIAH results and batch-size efficiency sweep are in Sections B and C. All models are trained on the public Nemotron-CC corpus and evaluated on public benchmarks (NIAH, LongBench, and standard zero-shot suites) under the settings stated in Section 4.

REFERENCES

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

Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pp. 50726–50753, 2025.

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

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

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

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Pranshu Chaturvedi, Parth Shroff, Tarun Suresh, Hangoo Kang, and Kaiyue Wen. Training hybrid block diffusion language models with partial bidirectionality. *arXiv preprint arXiv:2607.02805*, 2026.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of machine learning research*, 24(240):1–113, 2023.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ztn8FCR1td.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Re. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=COZDy0WYGg.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Guangxin He, Shen Nie, Fengqi Zhu, Yuankang Zhao, Tianyi Bai, Ran Yan, Jie Fu, Chongxuan Li, and Binhang Yuan. Ultrallada: Scaling the context length to 128k for diffusion large language models. *arXiv preprint arXiv:2510.10481*, 2025.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024. COLM 2024.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Xiaoran Liu, Yuerong Song, Zhigeng Liu, Zengfeng Huang, Qipeng Guo, Ziwei He, and Xipeng Qiu. Longllada: Unlocking long context capabilities in diffusion llms. *arXiv preprint arXiv:2506.14429*, 2025a.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025b.

Lingkun Long, Yushi Huang, Shihao Bai, Ruihao Gong, Jun Zhang, Ao Zhou, and Jianlei Yang. Focus-dllm: Accelerating long-context diffusion llm inference via confidence-guided context focusing. *arXiv preprint arXiv:2602.02159*, 2026.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=CNicRIVIPA.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *Advances in Neural Information Processing Systems*, 38:149009–149033, 2025.

Quan Nguyen-Tri, Mukul Ranjan, and Zhiqiang Shen. Attention is all you need for kv cache in diffusion llms. *arXiv preprint arXiv:2510.14973*, 2025.

Jinjie Ni, Qian Liu, Longxu Dou, Chao Du, Zili Wang, Hang Yan, Tianyu Pang, and Michael Qizhe Shieh. Diffusion language models are super data learners. *arXiv preprint arXiv:2511.03276*, 2025.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2025.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. In *Advances in Neural Information Processing Systems*, 2021.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pp. 28043–28078. PMLR, 2023.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *International Conference on Learning Representations*, 2025.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T Chiu, and Volodymyr Kuleshov. The diffusion duality. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9P9Y8FOSOk.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 103131–103167. Curran Associates, Inc., 2024. doi: 10.52202/079017-3277.

Vaibhav Singh, Oleksiy Ostapenko, Pierre-André Noël, Eugene Belilovsky, and Torsten Scholak. Diffumamba: High-throughput diffusion lms with mamba backbone. In *International Conference on Machine Learning*, 2026.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-CC: Transforming Common Crawl into a refined long-horizon pretraining dataset. *arXiv preprint arXiv:2412.02595*, 2024. ACL 2025.

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

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024.

Dustin Wang, Rui-Jie Zhu, Steven Abreu, Yong Shan, Taylor Kergan, Yuqi Pan, Yuhong Chou, Zheng Li, Ge Zhang, Wenhao Huang, et al. A systematic analysis of hybrid linear attention. *arXiv preprint arXiv:2507.06457*, 2025a.

Xu Wang, Chenkai Xu, Yijie Jin, Jiachun Jin, Hao Zhang, and Zhijie Deng. Diffusion llms can do faster-than-ar inference via discrete diffusion forcing. *arXiv preprint arXiv:2508.09192*, 2025b.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dllm v2: Efficient block-diffusion LLM. *arXiv preprint arXiv:2509.26328*, 2025.

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

Jing Xiong, Qi Han, Shansan Gong, Yunta Hsieh, Chengyue Wu, Chaofan Tao, Chenyang Zhao, and Ngai Wong. Prefilling-dllm: Predictive prefilling for long-context inference in diffusion language models. *arXiv preprint arXiv:2606.10537*, 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.

Table 6: Full training and model hyperparameters, shared across `Attn`, `Hybrid`, and `Mamba` unless a per-model value is given (the sequence mixer is the only architectural difference; Table 1).

Hyperparameter	**Value**
Architecture	
Model dimension d_{model}	2560
Layers	28
Attention heads (= KV heads)	20
Head dimension	128
MLP hidden size	7680
Block size G	32
RoPE θ	5×10^5
Vocabulary (incl. mask)	126 464
Mixer (`Attn` / `Hybrid` / `Mamba`)	28 attn / 5 attn + 23 Mamba-2 / 28 Mamba-2
Data	
Corpus	Nemotron-CC (Su et al., 2024)
Tokenizer	LLaDA tokenizer (Nie et al., 2025)
Sequence packing	packed, no padding
Objective and optimization	
Training objective	single-frontier block diffusion
Loss weighting	uniform
Optimizer	AdamW (β_1=0.9, β_2=0.999, ϵ=10^{-8})
Peak learning rate	1×10^{-4} (5×10^{-5} for `Mamba`)
LR schedule	cosine decay
Warmup	5% of steps
Weight decay	0.1
Gradient-norm clip	1.0 (0.5 for `Mamba`)
Precision	`bf16`
Batching and token budget	
Sequence length (training context)	1024
Per-device batch	16
Global batch	4096 sequences
Tokens per step	$1024 \times 4096 \approx 4.2$M
Training steps	71 525
Total tokens	≈300B
Systems and checkpointing	
Parallelism	FSDP
Attention kernel	PyTorch SDPA
Gradient checkpointing	off
`torch.compile`	off
Evaluation interval	every 100 steps (≤8192 samples)
Checkpoint interval	every 50 steps
Model selection	best validation NLL

A Training and model hyperparameters

Table 6 lists the full training and architecture configuration. The three denoisers are identical except for the sequence mixer (Table 1); every other setting is shared, with the two per-model values noted in the table.

B Per-depth needle-in-a-haystack

Table 7 reports the retrieval accuracy behind Figure 3 by needle depth (0 = far from the end-of-sequence query, 1.0 = adjacent) at the two lengths just past training length. All models show the recency/sliding-window effect, but only the linear-state backbones still retrieve away from the most recent positions at $4\times$.

Table 7: **Per-depth NIAH retrieval past the training length.** Retrieval (%) at L=2k (2×) and L=4k (4×) (3B; depth 0 = oldest, 1.0 = adjacent to the query). Every backbone peaks on *recent* needles, but only `Mamba`/`Hybrid` keep retrieving older ones past training length; `Attn` is near-zero at all but the last depth.

L	Model	depth 0.0	0.25	0.5	0.75	1.0
2k	`Attn`	1	1	1	5	53
	`Hybrid`	19	34	15	100	100
	`Mamba`	26	67	86	100	99
4k	`Attn`	0	0	0	0	9
	`Hybrid`	2	7	7	73	100
	`Mamba`	6	4	5	49	98

C FULL BATCH-SIZE EFFICIENCY SWEEP

Figure 2 plots batch 1 (exact numbers in Table 8); Figure 4 gives the full sweep over batch sizes $1/2/4/8$. `Mamba` throughput *scales with batch and stays flat in length* (1593 tok/s at batch 8 for *every* length to 256k, $\sim$5.4× the batch-1 rate, at a constant $\sim$8 GB), while `Attn`'s $\mathcal{O}(B \cdot L)$ KV-cache exhausts the H100 (`OOM`) at progressively shorter lengths as batch grows (last feasible: 256k@B1, 128k@B2, 64k@B4, 32k@B8). At 256k, `Mamba` at batch 8 reaches 1593 tok/s while `Attn` cannot run even batch 2; its best feasible aggregate is batch 1 at 114 tok/s, the $\sim$14× gap quoted in Section 5.1.

Table 8: **Batch-1 efficiency vs. context length.** Per-step latency (ms), peak memory (GB), and end-to-end decode throughput (tok/s, S=16) vs. context length (3B, H100 80 GB, cudagraph), the exact values behind Figure 2. `Mamba` is constant ($\sim$6.8 ms, $\sim$7.5 GB) out to 256k while `Attn` grows to 29 ms and 82 GB; best per length shaded.

	Latency (ms) ↓			Peak mem (GB) ↓			Gen. tput (tok/s) ↑		
L	**Attn**	**Hybrid**	**Mamba**	**Attn**	**Hybrid**	**Mamba**	**Attn**	**Hybrid**	**Mamba**
64	5.55	6.53	6.77	6.46	7.80	7.59	346	305	296
4k	6.12	6.62	6.75	7.69	8.08	7.41	336	303	296
16k	7.28	6.84	6.74	11.24	8.74	7.44	307	299	296
64k	11.79	7.65	6.72	25.40	11.32	7.51	229	282	296
128k	17.70	8.66	6.72	44.26	14.74	7.58	171	263	296
256k	29.31	10.74	6.76	81.91	21.52	7.65	114	231	297

D QUALITY–THROUGHPUT TRADE-OFF

Full per-budget MAUVE and generative-perplexity results behind Section 5.3.

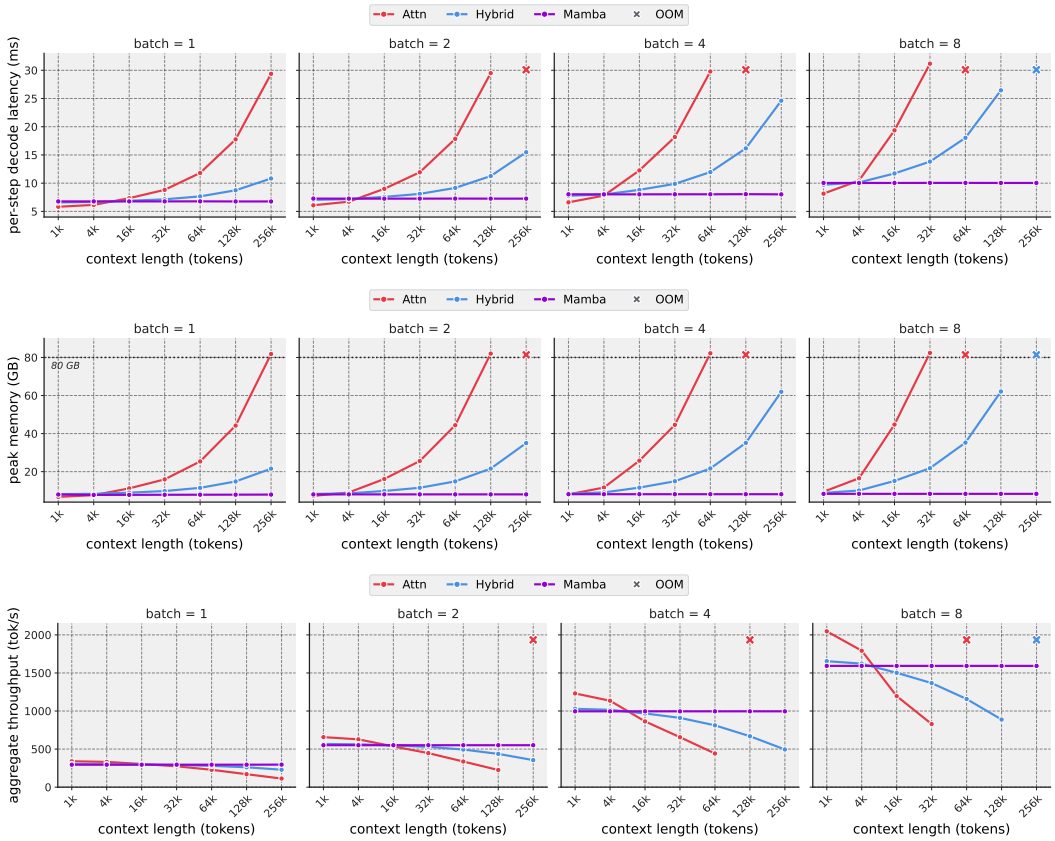


Figure 4: **Efficiency vs. context length across batch sizes (3B, single H100 80 GB, `bf16`, cudagraph).** Top to bottom: per-step decode latency, peak memory, aggregate decode throughput (S=16); within each row the four panels are batch $1/2/4/8$ (left to right). `Mamba` stays $\sim$8 GB at every (batch, length); `Attn`'s last-feasible length shrinks with batch (256k@B1, 128k@B2, 64k@B4, 32k@B8), marked `OOM` ($\times$) beyond.

Table 9: **The denoising budget S trades quality for speed.** Per-block denoising budget S (3B, cached generation, 64 $\rightarrow$ 128-token continuations, 512 samples): larger S monotonically lowers Gen-PPL and broadly raises MAUVE, at 5–6$\times$ lower throughput (best per column shaded). MAUVE (k-means over GPT-2-large features) is noisy for small base models, so Gen-PPL is the cleaner signal.

	MAUVE ↑			Gen-PPL ↓		
S	`Attn`	`Hybrid`	`Mamba`	`Attn`	`Hybrid`	`Mamba`
4	0.055	0.050	0.043	15.26	17.14	18.50
8	0.141	0.148	0.099	12.10	12.85	13.05
16	0.236	0.289	0.287	9.22	9.06	9.50
32	0.180	0.210	0.190	6.44	6.48	6.52