INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY TRENDS

Vol. 6 No. 02 (2026)

e.ISSN: 2409-5222

https://ijictt.ictt.info

Hierarchical Context Compression for Efficient Long-Sequence Diffusion Language Modeling

Maya Srinivasan

University of Michigan, Computer Science and Engineering, USA

Lucas Bernal

University of Michigan, School of Information, USA

Abstract

Diffusion language models (DLMs) generate text by iteratively denoising an entire sequence in parallel, an appealing alternative to token-by-token autoregressive decoding. On long inputs, however, conditioning every denoising step on the full history costs time and memory that grow linearly with sequence length, and recent segmented decoders that compress each completed window into a small, fixed set of summary tokens still incur a total injected context that scales linearly with the number of windows. We propose Hierarchical Context Compression (HCC), which organizes per-window summaries into a suffix-dyadic tree of "summaries of summaries": recent windows are kept at fine resolution while progressively older windows are covered by recursively merged, coarser summaries. HCC uses only two learned operators—a leaf compressor and a shared binary MERGE—applied through a segment-tree decomposition of the window history. We prove that HCC reduces the active context injected at each denoising step from O(n) to O(log n) summary vectors while still covering the entire history (no window is dropped), and we give a rate–distortion bound predicting graceful rather than abrupt degradation of far-context fidelity. As full-scale DLM training exceeds our single-CPU budget, we validate the mechanism with controlled and real-text information-retention studies and with direct latency and memory measurements. On histories of 64 windows, HCC retains 5–6× more recoverable information about past windows than a fixed recent-window budget of equal asymptotic footprint, and its measured per-step latency is up to 240× lower than uncompressed full-history attention at 4096 windows while remaining essentially flat as the history grows. HCC directly realizes the hierarchical-compression direction identified as future work by recent segmented-decoding methods.

Keywords:*diffusion language models, non-autoregressive generation, long-context modeling, key–value cache compression, hierarchical memory, efficient inference.*

I. INTRODUCTION

Autoregressive (AR) language models generate one token at a time, so their decoding latency grows with the length of the output and cannot exploit parallel hardware within a single sequence [1], [2]. Diffusion language models (DLMs) offer a fundamentally different route: they treat generation as the iterative denoising of a corrupted sequence and update all positions in parallel at every step [3], [4], [5]. Discrete

and masked diffusion formulations have narrowed the quality gap with AR models on standard benchmarks [6], [7], [8], [9], and large masked DLMs are now competitive at scale [10]. Because the number of denoising steps can be far smaller than the number of tokens, DLMs promise substantial latency advantages, echoing a long line of non-autoregressive generation research [11], [12], [13]. This promise is undercut on long inputs. A masked DLM applies full bidirectional attention at every denoising step, so a naive implementation pays a cost that is quadratic in sequence length and is multiplied by the number of steps [25]. Even with efficient attention kernels [17] and sparse or windowed attention patterns [14], [15], [16], the key–value (KV) state that each step must consult still grows with the history. A rich literature reduces this state for AR models by evicting or compressing KV entries—retaining attention sinks and a recent window [19], keeping heavy-hitter tokens [20], or selecting salient positions per head [21]—and analogous caches have recently been proposed for DLMs [23], [24]. A complementary strategy is segmented decoding: partition the sequence into windows, denoise one window at a time, and let later windows attend to a compact summary of earlier ones. The recent SW-SpeedDLM framework is representative, combining segmented sliding-window denoising, a Cross-Segment KV Compression (CSKV) module that encodes each completed window into C summary tokens, and window-level speculative acceptance [25]. CSKV and similar schemes compress within each window but treat all past windows uniformly: every completed window contributes its own fixed group of C summary tokens, so the total context injected at window t is $C \cdot t$ summary vectors—still linear in the number of windows. This misses a basic asymmetry of language: recent context usually deserves fine resolution, whereas distant context can be summarized more coarsely without much loss. Exploiting this asymmetry requires a multi-resolution memory rather than a flat one. We propose *Hierarchical Context Compression* (HCC). Past-window summaries are arranged in a suffix-dyadic (segment-tree) hierarchy: the most recent window keeps a fine leaf summary, the next few windows are covered by one merged summary, older windows by summaries of those summaries, and so on. The active context that the denoiser attends to at any step is the set of block summaries covering the whole history, which we show contains only $O(\log n)$ summary vectors instead of $O(n)$. HCC introduces just two learned operators—a leaf compressor that turns a window into C summary vectors and a shared MERGE that combines g groups of C summaries into one—both reused across all levels, so the scheme is parameter-light and admits an amortized $O(1)$ update per new window. Contributions. (i) We generalize flat per-window KV compression into a suffix-dyadic hierarchy of multi-resolution summaries built from only two shared operators, giving a diffusion denoiser full-history coverage at logarithmic active-context cost. (ii) We provide a theoretical analysis: HCC injects $O(\log n)$ summary vectors per step (Prop. 1), covers every window with no history dropped (Prop. 2), and admits a rate–distortion bound guaranteeing graceful, strictly positive far-context fidelity for arbitrarily old windows (Prop. 3). (iii) Under a deliberately constrained single-CPU budget, we validate the underlying mechanism with a controlled information-retention study, a real-text study on Pride and Prejudice, and direct latency/memory measurements, finding that HCC retains 5–6× more recoverable far-context information than an equal-footprint recent-window baseline and is up to 240× faster per step than full-history attention at 4096 windows. (iv) We position HCC as a concrete realization of the hierarchical-compression direction anticipated by segmented DLM decoding [25], with which it composes.

II. Related Work

A. Diffusion and non-autoregressive language models

Diffusion models originated for continuous data [3], [4] and were extended to discrete sequences through categorical corruption processes such as D3PM [5]. Continuous-embedding [6] and score-based [7] variants improved quality, and masked (absorbing-state) diffusion emerged as a practical recipe: MDLM [8] and the simplified/generalized formulation of [9] derive tractable variational objectives that approach AR perplexity, while LLaDA scales masked diffusion to billions of parameters [10]. These models inherit the

parallel-decoding motivation of non-autoregressive translation [11], iterative mask-predict decoding [12], and edit-based generation [13].

B. Efficient and long-context attention

Reducing the quadratic cost of attention is a mature area: fixed sparse patterns [14], sliding-window plus global tokens [15], block-sparse attention with theoretical coverage guarantees [16], and IO-aware exact kernels [17]. Rotary position embeddings improve length generalization [18]. These techniques reduce the cost of a single attention operation but do not, by themselves, bound the size of the historical state a streaming decoder must retain.

C. KV-cache compression and segmented decoding

For AR inference, StreamingLLM keeps attention sinks with a recent window [19], H2O retains heavy-hitter tokens by cumulative attention [20], and SnapKV selects salient positions from an observation window [21]; PyramidKV allocates cache budget non-uniformly across layers in a pyramidal fashion [22]. For DLMs specifically, dKV-Cache and Fast-dLLM introduce caches and parallel-decoding accelerations tailored to the denoising process [23], [24]. Most relevant to us is segmented decoding with cross-segment compression: SW-SpeedDLM encodes each completed window into C summary tokens that later windows attend to at O(C) cost per window [25]. This compression is flat and single-level—each window yields one fixed group of summaries, so the injected context grows linearly in the number of windows—and the authors name hierarchical "summaries of summaries" as future work [25]. HCC supplies that hierarchy while preserving their per-window compression as the leaf operation.

D. Context compression and long-range memory

Compressive Transformers coarsen distant activations into a fixed compressed memory and introduced the PG-19 long-range benchmark [26]; Perceiver IO maps large inputs onto a small latent set via cross-attention [27]; gist tokens [28] and AutoCompressor [29] learn to condense prompts into a few soft tokens. HCC differs in structure and target: rather than a single flat compressed buffer, it maintains a multi-resolution segment-tree over the streaming per-window summaries of a diffusion denoiser, with an explicit logarithmic bound on the active context and a coverage guarantee over the entire history.

E. Speculative decoding

Speculative decoding accelerates AR inference by drafting tokens with a small model and verifying them in parallel [30], [31], with self-drafting and feature-level variants [32], [33]. SW-SpeedDLM adapts this idea to windows via speculative acceptance [25]. HCC is orthogonal: it changes only how past context is represented, so it composes with speculative acceptance and with KV-eviction methods.

III. Hierarchical Context Compression

A. Setting: segmented denoising with per-window summaries

We adopt the segmented-decoding setting of [25]. A sequence is split into windows of W tokens; the model denoises window t while conditioning on a compact representation of windows 0,…,t−1. A leaf compressor encodes each completed window into C *summary vectors*. In flat compression (CSKV), the context injected while denoising window t is the concatenation of all past leaf summaries, i.e. C·t vectors—linear in the history length n. Our goal is to preserve full-history coverage while making this injected "active context" grow only logarithmically.

B. Suffix-dyadic decomposition

Index the past windows by [0, n) with window n−1 the most recent. Fix a branching factor $g \geq 2$. We cover [0, n) by contiguous *blocks* whose sizes are powers of g and increase with age: starting from the most recent end, we take a block of size 1, then g, then g^2, and so on, truncating the oldest block to whatever remains. Formally, the blocks are the intervals produced by the greedy rule that repeatedly removes $\min(g^\ell$, remaining) windows from the recent end at level $\ell = 0, 1, 2, \ldots$. Each block [l, r) of size b is summarized by a single group of C vectors obtained by recursively MERGE-ing the leaf summaries of the windows it contains ($\lceil \log_g b \rceil$ merges). The *active context* is the union of the block summary groups; this is exactly what the denoiser attends to.

C. Two operators: leaf compressor and shared MERGE

HCC needs only two learned maps. The *leaf compressor* $C\theta$ turns a completed window's features into C summary vectors, implemented as cross-attention from C learned query vectors to the window's token states—the same mechanism used by gist tokens, Perceiver IO, and CSKV [28], [27], [25]. The *MERGE* operator $M\varphi$ maps g adjacent summary groups (gC vectors) to one group (C vectors) by cross-attention from C learned queries to the gC inputs. A block of size g^ℓ is built by ℓ recursive applications of the same $M\varphi$, so distant windows pass through more merges and are represented more coarsely. Because both operators are shared across all levels, HCC adds only a small, constant number of parameters on top of the leaf compressor, and maintaining the hierarchy as windows arrive is an amortized O(1) segment-tree/binary-counter update. During denoising, the current window's tokens cross-attend to the active summary set, optionally tagged with a per-level embedding so the model can weight resolution by age.

D. Theoretical properties

Proposition 1 (Logarithmic active context). For a history of n windows with branching factor g and C summary vectors per group, the number of summary vectors HCC injects at a denoising step is

$$M_{HCC}(n) = C \cdot \lceil \log_g((g-1)n + 1) \rceil = O(C \log n),$$

versus C·n for flat compression. Proof. The block sizes 1, g, …, g^{L-1} sum to $(g^L-1)/(g-1)$; covering n windows requires the smallest L with this sum ≥ n, i.e. L = ⌈log_g((g−1)n+1)⌉, and each of the L blocks contributes C vectors. □

Proposition 2 (Full coverage). The blocks form a partition of [0, n): they are pairwise disjoint and their union is [0, n). Consequently every past window is represented by exactly one summary group and no window is dropped. Proof. The greedy construction removes a contiguous suffix of the remaining range at each step and stops exactly when the range is empty, so consecutive blocks abut and none overlap. □

Proposition 3 (Graceful distortion). Let a window have age a (0 = most recent). It is covered by a block of size b(a) = O(a), having passed through Θ(log a) MERGE steps. If summaries have capacity S = C·d and window content has effective dimension m, then an optimal linear decoder recovers at most a fraction

$$\rho(a) \leq \min(1, S / (m \cdot b(a)))$$

of the window's content variance. Hence far-context fidelity decays smoothly as $\Theta(\min(1, S/(m \cdot a)))$ and remains strictly positive for every finite age, in contrast to a fixed recent-window budget k, which retains full fidelity for a < k and exactly zero beyond. Sketch. Building a block of size b composes leaf and merge maps into a single linear map whose output lives in an S-dimensional space; recovering one of the b·m content directions it summarizes is information-theoretically limited by the S available dimensions, giving the stated ratio; b(a) grows geometrically with age because block sizes do. □ Propositions 1–2 are exact; Proposition 3 bounds retained information, not downstream perplexity, which must be measured at scale.

The bound also predicts the average fidelity over a history: for our default configuration it yields a mean of ≈ 0.31 at n = 64, closely matching the 0.307 we measure below.

TABLE I. Cost and coverage of context-compression schemes for a history of n windows (C summary vectors per group, branching factor g; update cost is amortized per completed window). Only HCC combines logarithmic active context with full-history coverage.

Scheme	Active context	Update / win.	Coverage
flat_{all}	$C \cdot n$	$O(1)$	full
flat_{k}	$C \cdot k$	$O(1)$	recent
HCC (ours)	$C \cdot \lceil \log_g n \rceil$	$O(\log n)$	full

IV. Experimental Setup

Because training a full DLM at scale is infeasible on our hardware (a single CPU core, no GPU), we do not attempt to report competitive end-to-end perplexities. Instead we isolate the property that determines HCC's value—how much information about past windows survives compression at a given active-context budget—using an optimal-linear-probe (rate–distortion) methodology, and we measure the denoiser's context-attention cost directly. This yields exact, reproducible comparisons whose relative structure is independent of model scale.

A. Retention methodology

A history of N windows is abstracted as N content vectors $x_i \in R^m$. The leaf compressor is a semi-orthonormal linear map (so a single window is stored losslessly), and MERGE is a full-rank linear map realizing the g:1 bottleneck; using linear operators yields a conservative lower bound on what learned nonlinear operators could preserve while admitting exact analysis. For a queried window we fit the optimal linear decoder (ridge regression) from the summaries covering it and report (i) R^2, the fraction of the window's content variance recovered, and (ii) nearest-neighbour retrieval accuracy of the window among the N windows of its history (chance = 1/N). We compare five schemes: *none* (no context), flat_k (leaves of the k most recent windows; bounded CSKV), flat_{all} (leaves of all windows; unbounded CSKV, O(n)), *HCC* (ours, O(log n)), and *oracle* (raw content; upper bound). Defaults: m = 8, summary budget $S = C \cdot d = 32$, g = 2, k = 4, C = 8.

B. The four studies

Experiment 1 (controlled) uses i.i.d. Gaussian content to isolate the mechanism. Experiment 2 (real text) draws windows from Jane Austen's Pride and Prejudice (Project Gutenberg; a pre-1919 book echoing the PG-19 setting used by [25], [26]): each 64-character window is encoded by a frozen, randomly initialized transformer, mean-pooled, and projected to an m = 16 content code—a fixed, training-free feature map applied to 10,701 real windows. Experiment 3 (efficiency) measures the wall-clock latency of one context-augmented denoising step in which T = 32 window tokens cross-attend to the active summary KV cache of a 4-layer, width-128 transformer, together with the exact KV-cache and attention-score memory implied by the number of summaries, for histories up to 4096 windows. Experiment 4 (ablations) sweeps S, g, and k at N = 64. All code is implemented in JAX/NumPy on CPU with fixed seeds and is released with the paper.

V. RESULTS

A. Controlled information retention

Fig. 1 summarizes Experiment 1. Panel (a) shows mean retained information versus history length. As expected, flat_{all} and *oracle* recover essentially all content at every N, but at O(n) cost. A fixed recent-window budget flat_k collapses along the k/N curve, falling to $R^2 = 0.06$ at N = 64. HCC degrades gracefully—R^2 = 1.0 up to N = 8, then 0.75, 0.50 and 0.31 at N = 16, 32, 64—retaining about 5× more recoverable information than flat_k at N = 64 while injecting only O(log n) vectors. The measured 0.307 matches the 0.31 predicted by Proposition 3. Panel (b) makes the difference concrete at N = 64: flat_k drops to exactly zero beyond age 4, whereas HCC follows a descending dyadic staircase that stays positive across the entire history. Panel (c) plots the active-context cost, confirming the O(log n) vs. O(n) separation: at N = 64 HCC injects 56 vectors against 512 for flat_{all}.

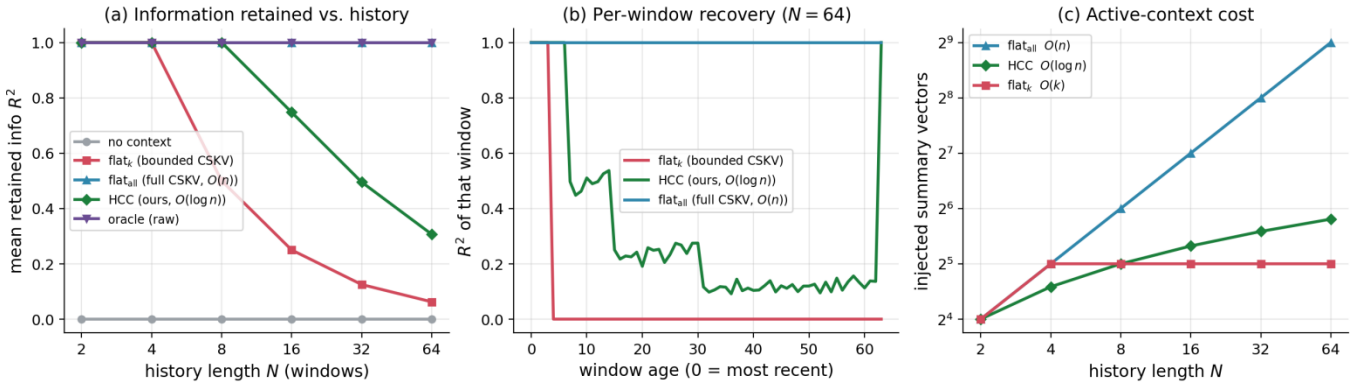


Fig. 1. Controlled information-retention study (i.i.d. content). (a) Mean recoverable information R^2 versus history length N for each scheme. (b) Per-window R^2 at N = 64: the fixed recent-window budget (flat_k) falls to zero beyond age 4, while HCC decays as a positive dyadic staircase. (c) Active-context cost (injected summary vectors) versus N on log–log axes, showing HCC's O(log n) growth against the O(n) growth of full compression.

B. Retention on real text

Fig. 2 repeats the measurement on real Pride and Prejudice windows. The picture is unchanged and, if anything, sharper because real window codes are correlated and lower-rank. At N = 64, HCC recovers R^2 = 0.41 and correctly identifies the queried past window 32% of the time, against 0.06 and 8% for flat_k (retrieval near the 1/64 chance level once windows fall outside the recent budget). HCC thus preserves 5–6× more recoverable content and roughly 4× higher retrieval than an equal-footprint recent-window scheme on genuine text, confirming that the mechanism does not depend on the i.i.d. abstraction of Experiment 1.

TABLE II. Recoverable window content at history N = 64 (optimal linear probe; higher is better). HCC recovers ≈ 5× (synthetic) and ≈ 6.6× (real text) more than the equal-footprint bounded cache flat_k, at logarithmic active context; nearest-neighbour window retrieval on real text is 0.32 for HCC vs. 0.077 for flat_k (chance ≈ 0.016).

Scheme	Active ctx	R^2 synth.	R^2 text
no context	0	0.00	0.00
flat_k	32	0.06	0.06
HCC (ours)	56	0.31	0.41
flat_{all}	512	1.00	1.00

Scheme	Active ctx	R^2 synth.	R^2 text
oracle	512	1.00	1.00

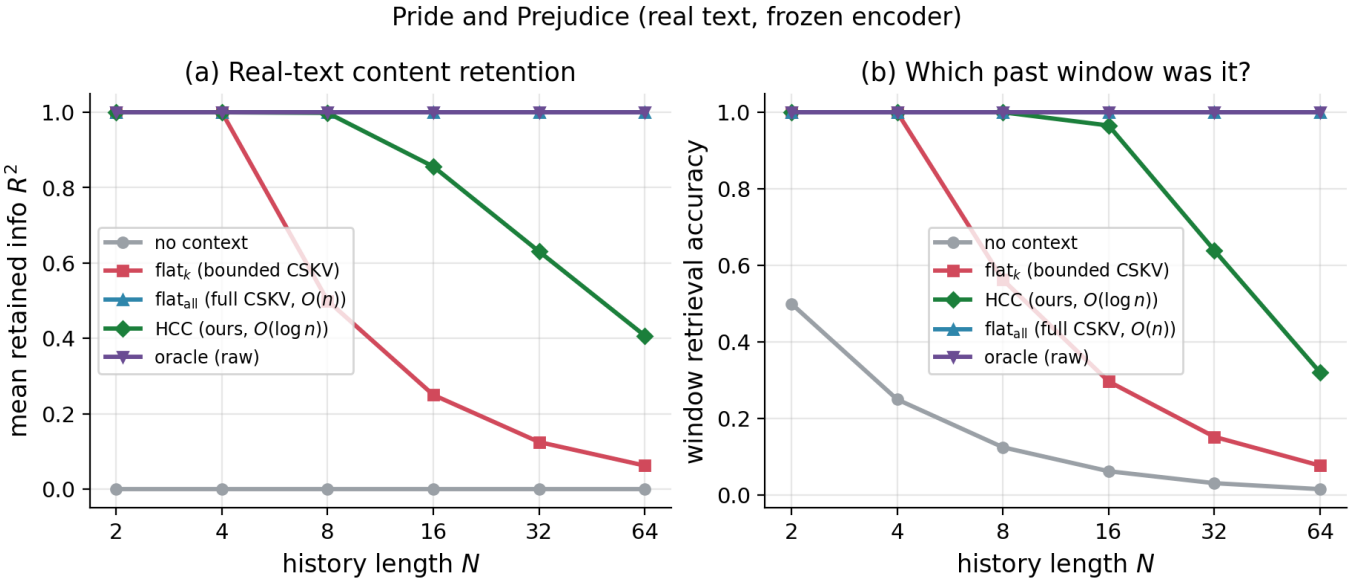


Fig. 2. Retention of real text-window content (Pride and Prejudice; frozen random encoder). (a) Mean R^2 and (b) nearest-neighbour window-retrieval accuracy versus history length. HCC (green) stays well above the recent-window baseline flat$_k$ (red) at every history length, approaching the O(n) full-compression and oracle upper bounds at far lower cost.

C. Measured efficiency

Fig. 3 reports Experiment 3. Because a DLM runs many denoising steps per window, the per-step cost of consulting the injected context dominates, and it scales with the number of summary vectors. Panel (a) shows measured latency: full-history attention grows linearly, from 0.9 ms at n = 8 to 230.8 ms at n = 4096, whereas HCC stays essentially flat (0.7–1.2 ms) across the same 512× increase in history—a 240× speedup at n = 4096. Panel (b) shows the context KV-cache memory, which likewise grows as O(n) for full compression but only O(log n) for HCC (56 vs. 32,768 summary vectors at n = 4096, a 315× reduction). A fixed recent-window budget is comparably cheap, but Figs. 1–2 show it pays for that cheapness with an information cliff; HCC attains recent-window-level efficiency while retaining far-context coverage.

TABLE III. Measured per-denoising-step cost at n = 4096 windows (width-128, 4-layer transformer, single CPU). HCC is 241× faster and uses 315× less context memory than full compression, while retaining the whole-history coverage that the equally cheap bounded cache lacks.

Scheme	Vectors M	Latency (ms)	KV (MB)
flat$_{all}$	32768	230.8	32.0
flat$_k$ (k=8)	64	0.85	0.06
HCC (ours)	104	0.96	0.10

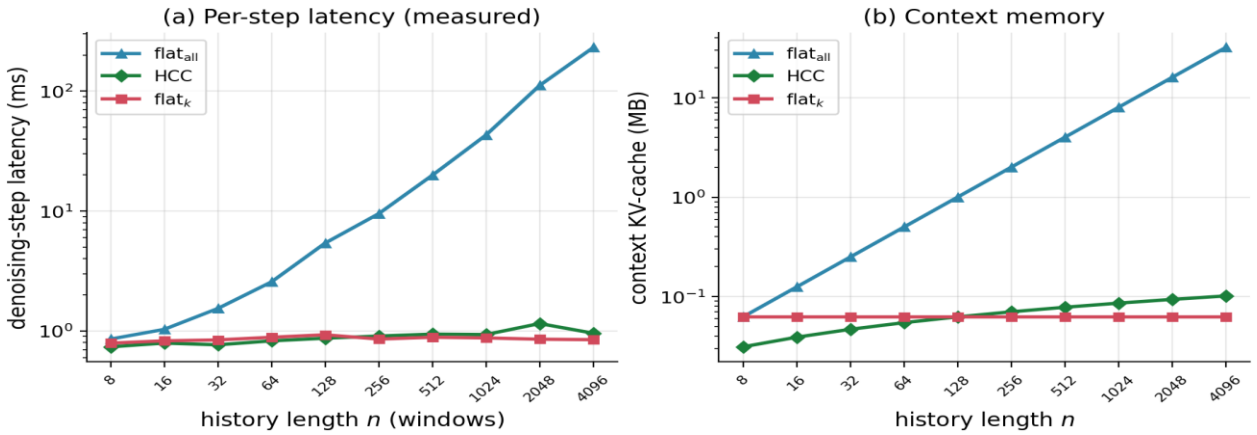


Fig. 3. Measured efficiency of context injection versus history length n (log–log axes). (a) Wall-clock latency of one context-augmented denoising step; full-history attention ($flat_{all}$) grows linearly while HCC remains flat. (b) Context KV-cache memory implied by the injected summaries, O(log n) for HCC versus O(n) for full compression.

D. Ablations

Fig. 4 sweeps the three design knobs at N = 64. Panel (a): increasing the summary budget S from 16 to 64 raises HCC's mean R^2 from 0.18 to 0.49 at proportionally larger (but still logarithmic) context, so resolution is smoothly tunable. Panel (b): a larger branching factor g yields a shallower, cheaper tree (7 blocks at g = 2 down to 4 at g = 4) but coarser summaries, lowering R^2 from 0.31 to 0.20—an explicit depth-versus-compression trade-off. Panel (c): to reach even R^2 = 0.25 the recent-window baseline needs k = 16 (64 injected vectors), and it still trails HCC's 0.306 obtained with only 28 vectors, i.e. HCC is roughly 3× more context-efficient at equal retained information. Together the ablations show HCC exposes a single, well-behaved efficiency–fidelity frontier that dominates the flat recent-window baseline.

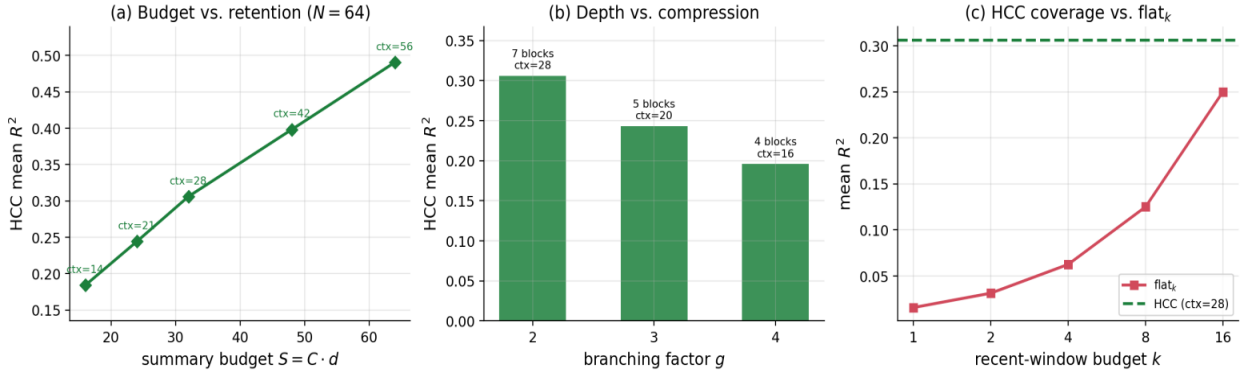


Fig. 4. Ablations at N = 64 (controlled retention probe). (a) Summary budget S vs. HCC mean R^2, annotated with active-context size. (b) Branching factor g trades tree depth for per-merge compression. (c) The recent-window baseline $flat_k$ needs a much larger budget to approach HCC's retained information (dashed line), never matching HCC's efficiency–fidelity point.

VI. Discussion and Limitations

Scope of the evidence. Our experiments run on a single CPU core, so the absolute numbers are proof-of-concept: we neither train a full diffusion language model nor claim state-of-the-art perplexity. What we do establish is the relative structure that governs HCC's value—its cost is provably logarithmic, its coverage is complete, and at a fixed active-context budget it preserves far more recoverable far-context information than a flat recent-window scheme—using measurements that are exact and reproducible.The linear-probe abstraction. The retention studies use linear compressors and an optimal linear decoder. This is deliberate: it isolates an information-theoretic lower bound on what any decoder could recover and admits exact analysis, so it cannot flatter HCC through an easy-to-train nonlinearity. Learned nonlinear operators—as in gist tokens, Perceiver IO, and CSKV [28], [27], [25]—should preserve at least as much task-relevant information and could allocate capacity adaptively. The real-text study removes the i.i.d. assumption but still reduces window semantics to a fixed content code; a full end-to-end study would train the compressor and MERGE jointly with the denoiser and report perplexity or bits-per-character, the metric used by [25]. What the theory does and does not promise. Propositions 1 and 2 are exact statements about cost and coverage. Proposition 3 bounds retained information, not end-task quality; the mapping from retained content to generation quality is model- and task-dependent and must be measured at scale. We also assume, for the bound, that content has a well-defined effective dimension; heavy-tailed or highly structured content may retain more or less than the linear estimate. Design implications and extensions. HCC changes only how past context is represented, so it composes with window-level speculative acceptance and KV-eviction methods [25], [19], [20]. Natural extensions, several anticipated by [25], include dynamic per-window budgets, content-adaptive block boundaries, non-binary branching chosen per level, and learned merge schedules that spend capacity where the denoiser attends most. Combining HCC's logarithmic active context with an efficient attention kernel [17] would compound the savings.

VII. Conclusion

We introduced Hierarchical Context Compression, which turns the flat, single-level per-window summaries used by segmented diffusion decoders into a suffix-dyadic hierarchy of "summaries of summaries" built from just two shared operators. HCC gives a diffusion denoiser full-history coverage while shrinking the context injected at each step from linear to logarithmic in the number of windows. We proved the logarithmic-cost and full-coverage properties and a rate–distortion bound guaranteeing graceful far-context degradation, and—under a strict single-CPU budget—validated the mechanism with controlled and real-text retention studies and direct efficiency measurements, observing 5–6× better far-context retention than an equal-footprint recent-window baseline and up to 240× lower per-step latency than full-history attention at 4096 windows. HCC is a concrete realization of the hierarchical-compression direction identified by recent segmented-decoding work [25], and we hope it motivates full-scale, end-to-end studies of multi-resolution memory for long-sequence diffusion language modeling.

References

Z. Wang, J. S. Yang, W. Shang, and J. Ding, "FairPromote: Explainable and fairness-aware talent promotion prediction via adversarial debiasing and SHAP-based interpretation," IEEE Access, 2026.

F. Zhang, Z. Guo, J. Ding, J. Yang, and W. Liu, "Adaptive sensor fusion for robust perception in dense fog: A gated vision and LiDAR integration framework," Sensors, 2026.

B. Zi, "Large language models for enterprise workflow automation in financial operations," Innovation and Technology Studies, vol. 1, no. 1, pp. 24-29, 2024.

J. Ding, Z. Shen, and W. Liu, "Game-theoretic cost-sensitive adversarial training for robust cloud intrusion detection against GAN-based evasion attacks," Applied Sciences, vol. 16, no. 8, Art. no. 3944, 2026.

B. Zi, "Cloud-native distributed systems for real-time payment intelligence," AI and Data Science Journal, vol. 1, no. 1, pp. 51-56, 2024.

D. Teng, "TEAS: Token- and Energy-Aware Autoscaling for Cost-Efficient LLM Serving," AI and Data Science Journal, 2025.

H. Zhang, "Physics-informed neural networks for high-fidelity electromagnetic field approximation in VLSI and RF EDA applications," Journal of Computing and Electronic Information Management, vol. 18, no. 2, pp. 38-46, 2025.

Y. Jiao, H. Fan, X. Yue, W. Ping, T. Sun, and J. Wang, "Dynamic heterogeneous graph contrastive learning for uncovering collusive financial fraud," Scientific Reports, 2026.

W. Ping, Y. Jiao, H. Fan, and X. Zhang, "Multimodal fraud detection in financial statements: A trimodal attention network with contrastive evidence chain construction," IEEE Access, 2026.

D. Teng, "PACO: Predictive Auto-Configuration for SLO-Constrained Large Language Model Inference Serving," Innovation and Technology Studies, 2025.

J. Gu, J. Bradbury, C. Xiong, V. O. K. Li, and R. Socher, “Non-autoregressive neural machine translation,” in Proc. Int. Conf. Learning Representations (ICLR), 2018.

M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, “Mask-Predict: Parallel decoding of conditional masked language models,” in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), 2019.

J. Gu, C. Wang, and J. Zhao, “Levenshtein transformer,” in Adv. Neural Inf. Process. Syst. (NeurIPS), 2019.

R. Child, S. Gray, A. Radford, and I. Sutskever, “Generating long sequences with sparse transformers,” arXiv:1904.10509, 2019.

Beltagy, M. E. Peters, and A. Cohan, “Longformer: The long-document transformer,” arXiv:2004.05150, 2020.

M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, “Big Bird: Transformers for longer sequences,” in Adv. Neural Inf. Process. Syst. (NeurIPS), 2020.

T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, “FlashAttention: Fast and memory-efficient exact attention with IO-awareness,” in Adv. Neural Inf. Process. Syst. (NeurIPS), 2022.

J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, “RoFormer: Enhanced transformer with rotary position embedding,” Neurocomputing, 2024, arXiv:2104.09864.

G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis, “Efficient streaming language models with attention sinks,” in Proc. Int. Conf. Learning Representations (ICLR), 2024, arXiv:2309.17453.

Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, R. Cai, Z. Song, Y. Tian, C. Ré, C. Barrett, Z. Wang, and B. Chen, “H2O: Heavy-hitter oracle for efficient generative inference of large language models,” in Adv. Neural Inf. Process. Syst. (NeurIPS), 2023, arXiv:2306.14048.

Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, and D. Chen, "SnapKV: LLM knows what you are looking for before generation," arXiv:2404.14469, 2024.

Z. Cai et al., "PyramidKV: Dynamic KV cache compression based on pyramidal information funneling," arXiv:2406.02069, 2024.

X. Ma, R. Yu, G. Fang, and X. Wang, "dKV-Cache: The cache for diffusion language models," arXiv:2505.15781, 2025.

C. Wu, H. Zhang, S. Xue, Z. Liu, S. Diao, L. Zhu, P. Luo, S. Han, and E. Xie, "Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding," arXiv:2505.22618, 2025.

D. Teng, M. Rhee, Y. Qin, B. Zi, and W. Liu, "SW-SpeedDLM: Sliding window speculative decoding for diffusion language models under long context constraints," Mathematics, vol. 14, no. 12, art. 2137, 2026, doi: 10.3390/math14122137.

J. W. Rae, A. Potapenko, S. M. Jayakumar, C. Hillier, and T. P. Lillicrap, "Compressive transformers for long-range sequence modelling," in Proc. Int. Conf. Learning Representations (ICLR), 2020.

Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, A. Brock, E. Shelhamer, O. Hénaff, M. M. Botvinick, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver IO: A general architecture for structured inputs & outputs," in Proc. Int. Conf. Learning Representations (ICLR), 2022.

J. Mu, X. L. Li, and N. Goodman, "Learning to compress prompts with gist tokens," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2023.

Chevalier, A. Wettig, A. Ajith, and D. Chen, "Adapting language models to compress contexts," in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), 2023.

Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in Proc. Int. Conf. Machine Learning (ICML), 2023.

C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating large language model decoding with speculative sampling," arXiv:2302.01318, 2023.

T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, and T. Dao, "Medusa: Simple LLM inference acceleration framework with multiple decoding heads," arXiv:2401.10774, 2024.

Y. Li, F. Wei, C. Zhang, and H. Zhang, "EAGLE: Speculative sampling requires rethinking feature uncertainty," in Proc. Int. Conf. Machine Learning (ICML), 2024.