Computer Life
ISSN: 1819-4818 | Vol. 15, No. 1, 2026

Query-Aware Cross-Segment KV Compression with Attention-Mass Calibration: An Operator-Level Study

Felix Lin *, Xinyi Lee, Nikhil Cohen, Lana Li

Department of Computer Science and Engineering, Texas A&M University, Texas, USA
Corresponding author: Felix Lin (Email: felix.lin40@tamu.edu)

Abstract: Cross-segment KV compression reduces the history accessible during masked denoising, but the number of summaries retained from each completed segment can distort attention mass. We implement the learned-query CSKV pooling operation of SW-SpeedDLM using Q/K/V extracted from an actual four-head bidirectional masked encoder, then examine query-aware integer budget allocation and an analytic log(W/c) multiplicity correction. Evaluation uses 170 held-out 320-character spans from an Opticks transcription and a Wikipedia extract. With four 64-character segments, uniform pooling at the compressor's training budget (B=16) is hard to beat with prefix-based allocation (NMSE 0.0054 versus 0.0050), whereas at untrained budgets such as B=8 adaptive allocation avoids the collapse of uniform pooling (0.0389 versus 1.8093). With query-ranked slots, calibrated adaptive routing at B=16 achieves 0.0044 versus 0.0051 without calibration; however, a paired interval for its difference against uniform pooling crosses zero. Further independently initialized and variable-budget-trained compressors show that allocation effects depend on learned bank quality. The block-constant attention identity is verified numerically and final-bank storage is exactly 93.75% smaller at B=16 than 256 uncompressed positions. The experiment measures cross-window attention fidelity, not end-to-end diffusion generation or GPU throughput.

Keywords: Masked diffusion language models; KV cache; Attention pooling; Adaptive allocation; Reproducibility; Long context.

1. Introduction

The central scalability difficulty of Transformer inference is not merely that attention accesses many tokens; it is that relevant information has to survive representation, compression, and changing query distributions. The Transformer makes token interactions explicit through query, key, and value projections [1]. Masked token prediction exploits both directions of local context [2]. Diffusion-style language generation adds repeated denoising passes, amplifying the cost of attending to long histories. Structured discrete diffusion introduced absorbing-mask corruption [3], and masked diffusion models showed that repeated recovery of masked tokens can be an effective language-modeling objective [4]. Recent large-scale diffusion systems strengthen the motivation for efficient context handling [5].

Long-range processing presents two linked constraints. A bounded local window can control per-pass working-set size, but information outside that window must be recalled without storing the complete history. Conversely, a summary that is inexpensive to store may misrepresent the probability mass of old segments when queries change. Sparse patterns such as Longformer [6] and BigBird [7] target attention structure directly. Exact attention kernels such as FlashAttention [8] and its successor [9] reduce data movement, but do not by themselves determine which old content should remain in a compact cross-window memory. The present study isolates this latter decision.

Our research question is specific: given four completed windows, a partially masked new window, and a strict total budget B for compressed KV vectors, can an allocation conditioned on the current query reduce error relative to a fixed number of vectors per previous window? A second issue arises automatically when budgets differ: a softmax over compressed vectors treats a segment with eight retained representatives as twice as numerous as one with four, even if both describe the same original number of tokens. The importance of such implicit multiplicity can be tested independently of downstream language generation.

Our contribution comprises a head-consistent implementation of the cross-segment learned-query pooling (CSKV) of SW-SpeedDLM [25], a fixed-budget integer router, and a multiplicity correction with a block-constant exactness result. We evaluate model-specific budget response, routing and ranking ablations, and independently initialized query banks. Gains are strongly budget- and initialization-dependent: attention mass calibration has a clean mathematical rationale, while dynamic slot allocation is not a universal replacement for fixed allocation. Results are limited to single-head attention-output reconstruction in a compact masked encoder.

2. Related work

2.1. Long-context attention and memory policies

A sliding window trades full context for local reuse, but simple recency retention is not a sufficient memory policy in every model. StreamingLLM showed that attention sinks are consequential in autoregressive streaming models [10]. H2O retains frequent heavy-hitter entries along with recent keys [11]; SnapKV predicts useful KV locations from an observation region [12]. PyramidKV redistributes storage across layers in accordance with an observed layer-dependent information funnel [13]. These methods exploit particular forms of importance scoring; they are useful reference points for the question of how one should allocate a limited cache. In contrast, our experiment compresses each completed segment into learned mixtures of its keys and values, rather than deleting individual elements from an autoregressive

running cache. Neuro-Elastic provides a complementary example of adaptive inference engineering in which latency and hardware resource outcomes are measured alongside model behavior [27].

Trainable cross-segment memories offer a different interface. AutoCompressors learn continuous summary vectors and condition later segments on them [14]. Compressive Transformers preserve an additional compressed long-range memory [15]. Perceiver IO demonstrates how learned queries can collect information from a much larger input into a smaller latent set [16]. These works help explain why attention pooling can serve as a compact memory operator; our contribution instead concerns how many such representatives to retain from each finished segment, and how the resulting softmax should be normalized.

2.2. Masked denoising and approximate attention

Parallel masking-based refinement is useful for understanding how a partially observed query window can depend on distant completed segments. Mask-Predict demonstrated iterative refinement for conditional generation [17]; discrete score-entropy modeling provides a distinct diffusion formulation [18]. LongBench probes long-range retrieval, summarization and reasoning [19], but our experiments measure an operator-level attention response on different real corpora rather than LongBench scores. CAGR further illustrates the value of operator-level performance experiments across accelerator backends when assessing practical inference efficiency [28].

Speculative decoding accelerates autoregressive sampling through a draft–verify mechanism [20]. We do not optimize such a mechanism here. Our question concerns the memory supplied to any subsequent denoising call. Alternative attention families include linear recurrences [21] and kernel-feature approximations such as Performers [22]. Position strategies such as ALiBi [23] and rotary embeddings [24] address length extrapolation, a separate issue from compressing already completed windows. The methods compared experimentally below all use the same measured dense cross-window attention operator as their reconstruction target.

3. Method

3.1. Target cross-window operator

Let W be the number of positions in a finished segment, P the number of prior segments, d the per-head key width, and Q in R^{W×d} the queries of the current masked window. For segment j, denote by K_j,V_j in R^{W×d} its cached representations. Concatenating the P memories gives K,V in R^{PW×d}. We define the oracle context response Y(Q,K,V)=softmax(QK^T/√d)V. This is a single-head cross-window attention response used for controlled measurement, and is neither a probability distribution over vocabulary tokens nor a generated language-model sample. The same queries and uncompressed K,V are used for every method when calculating the oracle.

We implement the learned-query cross-segment compression operation of SW-SpeedDLM [25]: a trainable bank of C_max queries pools each completed segment's keys and values by A_j=softmax(Q_ψ K_j^T/√d), K̃_j=A_jK_j, and Ṽ_j=A_jV_j. Later-window queries read the concatenated compressed K̃ and Ṽ through ordinary softmax attention. This use preserves the central CSKV operation and its segment-to-segment interface. We adapt the training objective to the present operator-level setting: the frozen masked encoder supplies K,V, while Q_ψ minimizes mean-squared error to the dense attention output. Only the summary-query bank is optimized in this stage.

3.2. Budget-adaptive allocation

The allocation problem differs from choosing the attention mask. We first compute up to C_max candidate pooled vectors for each completed segment, then choose integers c_j satisfying 1≤c_j≤C_max and ∑_{j=1}^P c_j=B. The candidate representation is fixed and the per-segment budget changes with the current masked-window queries. We use the mean query q̄ and a segment relevance statistic s_j=max_r(q̄^T K̃_{jr}/√d). This proxy requires only compressed keys, not the discarded original segment K/V. Its scores are converted to probabilities p_j=exp(s_j/τ)/∑_l exp(s_l/τ).

Starting at one slot per segment, a greedy allocator repeatedly assigns the next of B−P slots to the segment maximizing p_j/(c_j+0.3), subject to c_j<C_max. The denominator implements diminishing returns; the fractional constant is fixed and was not fitted on the test split. The temperature τ and relevance statistic were selected from a small, disclosed validation grid and then frozen. For the budget curve, selection uses the leading c_j learned candidate slots. A separate experiment reorders candidates by query similarity, testing whether within-segment query ranking helps. The integer allocator always preserves the exact B-slot budget, so budget-controlled comparisons do not conflate quality with additional stored vectors.

3.3. Multiplicity calibration

An unequal number of compressed keys per segment distorts attention even in a case where compression is otherwise exact. If all W original keys and values inside segment j are identical, c_j repeated representatives should contribute the same attention mass as W originals. Ordinary softmax assigns log-mass log(c_j)+q^T k_j/√d; the true segment contributes log(W)+q^T k_j/√d. We therefore add b_j=log(W/c_j) to every selected compressed key from segment j before applying softmax; this is the segment-level analogue of the proportional attention used for merged tokens in Token Merging [26], derived here for budget-dependent multiplicities together with an exactness result. The calibrated response is Ŷ=softmax(QK̃_B^T/√d+b)Ṽ_B, where b repeats log(W/c_j) for each selected slot in segment j.

Proposition 1 (exactness for block-constant memories): If K_j has W identical rows k_j and V_j has W identical rows v_j for every segment, then the calibrated compressed response equals the dense oracle for any feasible positive allocation (c_1,...,c_P). Indeed, pooling identical inputs produces the same k_j,v_j in each summary slot. Summing the c_j unnormalized exp(q^Tk_j/√d+log(W/c_j)) terms gives W exp(q^Tk_j/√d) for every j, reproducing both numerator and denominator of dense attention. The result is independent of the number of representatives retained per segment. For nonconstant windows the correction removes this combinatorial bias but does not guarantee optimal reconstruction.

Asymptotically the downstream attention operator reads B historical KV pairs instead of PW pairs. At fixed W and P, the query–history multiplication decreases from O(WPWd) to

O(WBd); the reusable summary bank costs O(PWC_max d) once per completed context set, plus temporary P C_max candidates for routing. In a repeated denoising procedure these one-time costs may be amortized, but our CPU implementation is not an optimized GPU kernel. Numerical storage calculations below refer to the final retained KV bank and explicitly exclude ephemeral candidate staging.

3.4. Proxy-distortion greedy variant

A more expensive control computes a surrogate response from the entire P C_max compressed bank, then greedily assigns B slots to reduce mismatch on ten sampled query positions. It only uses already-compressed candidates, never uncompressed raw KV during allocation. This diagnostic is computationally expensive and is not guaranteed to track full-attention error: self-consistency within a compressed representation is a different optimization target from fidelity to original KV. We include its observed failure mode rather than assuming proxy minimization is beneficial.

4. Experimental protocol

4.1. Real textual sources and partitioning

The experiment uses two non-synthetic, publicly distributed English sources: a public-domain plain-text transcription of Isaac Newton's Opticks (as distributed with the Go source tree) and a shortened English Wikipedia XML dump (as distributed with the Gensim test data). Parsing yields 560,782 and 4,112,392 normalized ASCII characters, respectively. The corpora are not PG-19 or LongBench and are not represented as those benchmarks. Each source is split contiguously by character into 70% training, 15% validation, and 15% test portions. Example sampling stays entirely inside the split boundaries. We take a 320-character slice, divide it into four clean prior windows and one partially masked query window of W=64 positions, and retain source identity for domain-stratified analysis.

The reconstruction set comprises 380 train, 110 validation, and 170 held-out test examples (equal counts per source). The training stage samples character segments from the training portions; evaluation windows sample random offsets from the disjoint test portions. Characters are represented as ASCII code points 0 through 127, with 128 reserved for the mask token. A 35% independent random masking probability is applied to the query window, while earlier windows are clean and considered completed. This models the partial observation of a denoising step without executing an entire diffusion schedule.

4.2. Masked encoder and learned compressor

The base is a two-layer bidirectional Transformer with exactly four attention heads, hidden size 48 and per-head width d=12. The self-attention tensor uses shape [batch, 4, 64, 12]; we extract Q/K/V from head zero of the second self-attention layer with the identical decomposition, thereby measuring a genuine trained head rather than an arbitrary projection slice. A maximum 64-character local window is trained for 420 AdamW steps at batch size 48 and learning rate 0.002 on masked-character cross-entropy. After freezing the encoder, a C_max=12 learned-query bank is fitted for 350 Adam steps at rate 0.05 to reconstruct the dense cross-window attention response; the objective is the sum of four-per-window reconstruction loss and 0.65 times twelve-per-window loss. The original corpus split and fixed seed are retained across all comparisons.

All computations run on CPU using PyTorch 2.10.0+cpu, float32, two CPU threads, and fixed seed 20261022. The selected relevance statistic is max candidate similarity with temperature tau=4.0, selected using the 110-slice validation set and reused unchanged for test and budget sensitivity. The released code preserves text hashes, model checkpoints, the validation grid and per-trial results.

4.3. Comparators, metrics, and uncertainty

The principal baseline applies the same learned-query CSKV pooling with a uniform c_j=B/P at budgets divisible by four. Adaptive uses the same bank and replaces only the integer allocation. We additionally evaluate a recency-skewed allocation, query-based candidate ranking, mass calibration removed, a zero-history response, and a proxy-distortion variant. Since these policies share weights and test examples, paired differences are meaningful. The main metric is a sample-normalized squared reconstruction error, NMSE_i=‖Ŷ_i−Y_i‖²_F/(‖Y_i−mean(Y_i)‖²_F+10^−8); we report its arithmetic mean. Cosine similarity of flattened attention outputs is a secondary check for the B=16 ablation.

Uncertainty is summarized using paired percentile bootstrap intervals on 170 test slices (4,000 resamples for budget studies and 2,000 for B=16 ablations). Random offsets within each held-out 15% corpus tail can produce overlapping passages; hence the resampled rows are not necessarily independent, and these intervals may underestimate document-level uncertainty. The budgets were explored after initial analysis, no simultaneous-comparison adjustment was applied, and the evidence is considered descriptive. Five independently trained alternative summary banks further expose bank-initialization and training-objective sensitivity.

5. Results and analysis

5.1. Accuracy under matched cache budgets

TABLE I. Normalized cross-window attention reconstruction error (lower is better)

B	Uniform	Adaptive+cal.	Error reduction
8	1.8093	0.0389	+97.9%
12	0.0052	0.0049	+7.3%
16	0.0050	0.0054	-9.0%
24	0.0048	0.0030	+36.9%
32	0.0047	0.0029	+37.4%

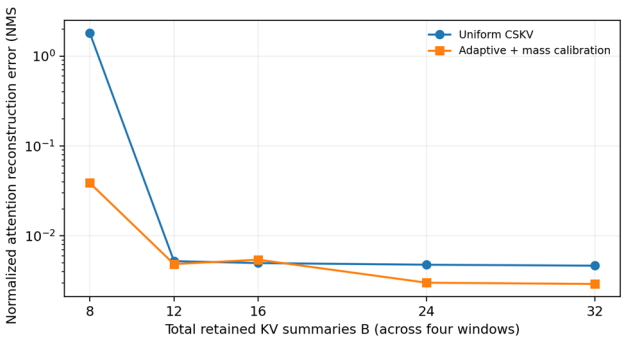


Fig. 1 Budget sensitivity of learned cross-window KV pooling. Each point is the mean of 170 real-text test traces under the same encoder and summary bank; the y-axis is logarithmic. The selection temperature is fixed from validation.

Table I and Fig. 1 report prefix-based slot retention, with one fixed encoder and compressor bank for all methods. At

B=8, uniform two-per-segment pooling gives NMSE 1.8093, whereas the calibrated router gives 0.0389; the paired difference is 1.7705 (descriptive 95% interval [1.4772, 2.1175]). The extremely large gap is partly a prefix-training mismatch: the bank was optimized for four and twelve summaries per segment, not two. At B=12 the difference is much smaller (0.0052 versus 0.0049), and its interval includes zero. This cautions against treating the B=8 contrast as an architecture-independent speed or accuracy improvement.

The preference reverses at B=16 for prefix-only candidate selection: equal four-slot retention attains 0.0050, adaptive prefix retention 0.0054. At B=24 the same adaptive policy scores 0.0030 against 0.0048 for uniform; its paired interval overlaps zero. Results at B=32 show the same direction, also without a uniformly positive paired interval. A learned bank may encode complementary slot identities; changing c_j modifies both summary diversity and softmax normalization. A single winning operating point would overstate the evidence.

5.2. Component and routing ablations

TABLE II. Sixteen-slot ablation, single frozen model

Configuration	Mean NMSE
No context	1.0062
Uniform CSKV	0.0050
Recency (1,2,5,8)	0.4826
Uniform + query ranking	0.0053
Adaptive + ranking, no cal.	0.0051
Adaptive + ranking + cal.	0.0044
Adaptive prefix + cal.	0.0054
Proxy-greedy + cal.	0.1607

Table II compares different policies at exactly B=16. The zero-context response yields NMSE 1.0062; uniform learned-query CSKV gives 0.0050, confirming that the reconstruction target has a meaningful contextual signal. A fixed recency-biased [1,2,5,8] allocation is much worse (0.4826), while uniform query re-ranking gives 0.0053. A proxy greedy scheme based on the full compressed candidate bank also performs poorly (0.1607), demonstrating that candidate-only self-consistency may mislead allocation.

For adaptive routing with candidate re-ranking, omitting the log multiplicity correction gives NMSE 0.0051. Adding it reduces NMSE to 0.0044, a 14.3% reduction. The paired no-correction-minus-calibrated interval is [0.0001, 0.0014], positive on this split. Calibrated routing is also below uniform in pooled mean (0.0050), but their paired interval includes zero. The effect differs by corpus: routing is slightly worse for Newton and better for Wikipedia. This distinction matters: mathematical mass calibration is validated more directly than a universal allocation advantage. As with the cross-segment CSKV interface in [25], each policy uses the same learned attention pooling and changes only the memory representation fed to the next query window.

The candidate-only greedy allocator yields NMSE 0.1607, far above the 0.0050 uniform reference. This is a substantive negative result. The surrogate response formed from all 48 candidates is not a calibrated teacher for individual slot additions: the greedy path may overfit its surrogate despite access to the same memory bank. It is inappropriate to describe this proxy as an oracle allocator. The lightweight relevance policy avoids this expensive surrogate-selection loop.

5.3. Storage and measured CPU operator overhead

For one selected attention head, four 64-position prior windows require 256 historical key–value entries. At d=12 and float32 this amounts to 24,576 bytes for both K and V. A retained 16-slot representation uses 1,536 bytes (1.5 KiB), or 93.75% fewer stored entries and bytes, excluding the candidate staging pool. The compact cross-attention score product has 64×16 instead of 64×256 query–memory pairs, a 16× reduction in that matrix-product count. At B=8 the corresponding final storage reduction is 96.875%. These are exact tensor-size and operator-count calculations, not empirical peak process-memory claims.

The actual CPU operator benchmark tests batches of 32 with eight timing repeats. The median dense attention operation is 0.570 ms, the Python-loop pooled dispatcher 4.046 ms, and bank production 0.570 ms. These measured CPU timings include distinct implementation overheads, and should not be divided into a claimed decoder speedup. A fused GPU implementation is necessary to test practical acceleration.

6. Budget-dependent behavior

6.1. What the softmax correction removes

The mass calibration has a direct numerical interpretation. Consider two original segments of W=64 positions each. If every key is the same and every query has the same key score, dense attention gives one-half of its total probability mass to either segment. Suppose the compressor retains one slot from the first segment and eight from the second. Without calibration, softmax sees nine identical logits and assigns probabilities 1/9 and 8/9 to the two segments. With the correction, the first stored slot is assigned weight exp(log 64)=64, while each of the eight stored slots in the second is assigned exp(log 8)=8; both segments now carry total unnormalized weight 64 and the intended equal split is recovered. Thus, the correction is not an arbitrary learned adjustment. It encodes the number of original positions represented by each summary slot.

The maximum absolute discrepancy between calibrated and full cross-window responses in this float32 unit test is 1.88e-05; the uncalibrated maximum is 2.215. The test uses 40 synthetic block-constant memories and [1,1,2,12] slots. It validates only the special-case proposition, independently from the real-text evaluation.

6.2. Identifying the regimes of advantage

Two distinct issues explain the uneven response over B. At B=8 a uniform two-summary prefix has received no direct reconstruction supervision, whereas the available adaptive counts include some segments with more than two trained candidate positions. At B=16 the uniform four-summary prefix matches the training loss exactly, making this setting naturally favorable to equal allocation. Above 16, prefix diversity and calibration interact with the distribution of query scores. Since the training loss includes a full twelve-slot term, simply increasing available candidate count is not the same as learning a nested representation for every intermediate budget.

The two sources do not have identical effects. At B=8 the uniform and adaptive means are 1.2087 and 0.0081 on Newton, and 2.4100 and 0.0696 on Wikipedia. At B=12, however, Newton favors uniform while Wikipedia favors

adaptive. The test has 85 spans from each source, but random-offset sampling may yield shared passages, so source-stratified means should not be treated as independent-document replication.

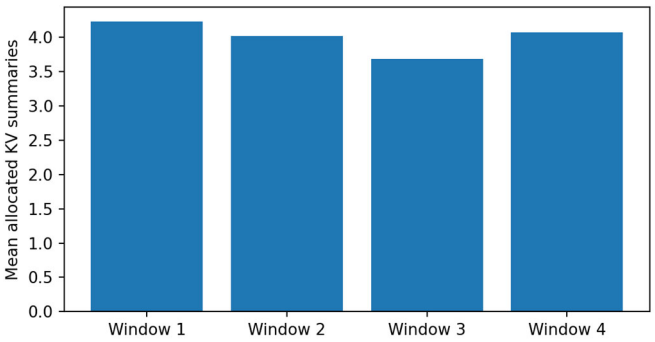


Fig. 2 Mean distribution of the sixteen retained slots across four previous windows under the validation-selected relevance allocator. Slot counts vary per query, but total retained slots remain sixteen for every example.

For B=16 the validation-selected, candidate-reordering adaptive policy allocates on average 4.23, 4.02, 3.68 and 4.07 slots to the four prior segments; 84.1% of test spans differ from equal allocation. The average can therefore look nearly uniform although individual query windows receive different distributions. The small size of the encoder and the dependence on candidate-bank training limit extrapolation to industrial-scale caches.

6.3. Independent compressor and budget-training runs

To test whether the conclusions depend on one query-bank initialization, we fit five additional learned-query compressors using two new fixed-loss seeds and three variable-budget-loss seeds. Each run uses the same frozen encoder, 350 optimizer steps, and an independently initialized 12-vector query bank. The variable-budget loss randomly selects an equal per-segment count from {2, 3, 4, 6, 8, 12} at each step, together with the twelve-summary auxiliary loss; routing mode and temperature remain frozen from the original validation grid. These tests investigate bank-training sensitivity, not model-scale or corpus-scale transfer.

TABLE III. Independent query-bank restarts at b=16 (prefix slots)

Compressor run	Uniform NMSE	Adaptive NMSE
fixed seed2	0.0041	0.0039
fixed seed3	0.0062	0.0033
variable budget seed1	0.0079	0.0049
variable budget seed2	0.0039	0.0042
variable budget seed3	0.0053	0.0047

Table III shows two independently seeded fixed-loss banks and three variable-budget banks under the identical prefix-based 16-slot evaluator. Some banks favor adaptive, whereas others favor uniform. This does not negate the block-constant mass identity; it means that query-bank optimization and routing cannot be evaluated as separable design decisions. All five fitted bank checkpoints and the full budget-by-run matrix are included in the reproducibility package. Their variation is a stronger warning against extrapolation than any single favorable pooled mean.

6.4. Accounting for compute and implementation artifacts

A fair deployment comparison must distinguish three costs. The first is the stored final K/V memory: B vectors, a strict and readily verifiable budget. The second is the staging representation of up to P C_max learned candidates; at P=4 and C_max=12, this is 48 pairs temporarily available during routing. The third is the work of scoring and selecting an allocation. The simple mean-query relevance score uses only compressed keys, so it does not reconstruct the discarded raw K/V. The proxy-distortion greedy variant runs many tiny attention comparisons against the 48-candidate surrogate and incurs more host overhead. On CPU the Python dispatcher can be slower despite processing fewer attention entries, as the measured 32-example microbenchmark confirms.

For downstream use, caching a single chosen allocation across successive denoising iterations could amortize routing and staging. That step is not evaluated in the current experiment: the test observes one masked query window and computes one exact-reference cross-attention response. The distinction also controls interpretation of memory reduction. The value 93.75% refers to the retained cross-window K/V entries at B=16 relative to PW=256 entries. It is not a statement that total model memory, activations, attention workspace, or peak resident set size falls by the same percentage. Similar caution applies to the 16× reduction in pairwise query–history products, which is a score-matrix operation count rather than an observed end-to-end throughput ratio.

7. Discussion

The experiments establish a specific, reproducible claim. The CSKV learned-query pooling operator is implemented on authentic trained attention-head K/V; an explicit log(W/c_j) term corrects segment multiplicity in a provable special case and improves the query-ranking ablation. Adaptive routing often lowers mean reconstruction error, but it can also worsen it, and paired confidence intervals against a uniform policy commonly cross zero. Additional compressor seeds produce mixed outcomes. The contribution is a mechanism and an empirical characterization, not evidence of broadly superior long-form generation.

The evaluation has deliberate boundaries. Our masked model is a compact ASCII-level encoder trained in a short CPU run; its learned attention head is not an independently evaluated diffusion sampler. The dense cross-window response is constructed from Q/K/V projections learned on isolated 64-character windows, so it is an operator-level probe, not a history-attending pathway trained end-to-end inside the encoder. The oracle is the exact output of this head's cross-window attention given past states, and there is no downstream masked-token accuracy, perplexity, multi-step sampling, LongBench task score, or long-form generation assessment. It would therefore be inappropriate to transfer our relative NMSE reductions to 8B model memory or decoding speed claims. Further, the two real textual corpora come from particular documents rather than benchmark test suites with many independent books.

Three limitations are important. First, mean-query relevance may neglect rare token-specific dependencies. Second, the C_max bank temporarily uses 48 summary pairs even when the final retained budget is 8–32, so persistent cache shrinkage does not imply peak allocation shrinkage.

Third, the compressor loss explicitly trains certain nested prefixes, and sensitivity to its optimization is borne out by additional independently initialized and variable-budget-trained query banks. Because each test corpus represents one underlying document stream, simple slice-level bootstrap cannot establish corpus-general or cross-seed significance.

The data and ablations also suggest implementable next steps. One route is a differentiable allocation prior jointly trained with the compressor so that summary diversity is preserved under variable c_j. Another is a small set of query probes, rather than one mean, with calibration still applied to segment mass. Finally, routing could be performed on GPU in batches, amortized over the many denoising steps of a window. The benchmark included here supplies a reproducible operator target against which these extensions can be tested before investing in full-model deployment.

8. Conclusion

We implemented a head-consistent version of the learned-query cross-segment KV pooling of SW-SpeedDLM, introduced fixed-sum relevance routing with multiplicity-calibrated logits, and characterized performance across tight memory budgets. The segment-mass correction recovers exact dense attention for block-constant memories and improves a measured real-text ablation. Across budget points and independently trained summary banks, dynamic allocation is not uniformly beneficial. These results clarify which approximation mechanisms merit direct integration into an iterative masked diffusion decoder; generation quality, multi-step scheduling and accelerator throughput remain tasks for a separate end-to-end evaluation.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998-6008. https://doi.org/10.48550/arXiv.1706.03762

[2] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT) (pp. 4171-4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

[3] Austin, J., Johnson, D. D., Ho, J., Tarlow, D., & van den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. Advances in Neural Information Processing Systems, 34, 17981-17993. https://doi.org/10.48550/arXiv.2107.03006

[4] Sahoo, S. S., Arora, S., Goyal, A., & Arora, S. (2024). Simple and effective masked diffusion language models. Advances in Neural Information Processing Systems, 37. https://doi.org/10.52202/079017-4135

[5] Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., Zhou, J., Lin, Y., Wen, J.-R., & Li, C. (2025). Large language diffusion models. Advances in Neural Information Processing Systems, 38. https://doi.org/10.52202/085713-1689

[6] Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. arXiv, arXiv:2004.05150. https://doi.org/10.48550/arXiv.2004.05150

[7] Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontañón, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2020). Big Bird: Transformers for longer sequences. Advances in Neural Information Processing Systems, 33, 17283-17297. https://doi.org/10.48550/arXiv.2007.14062

[8] Chen, J., Zhang, X., & Han, X. (2023). Explainable identification of tax-related financial reporting anomalies in U.S. public companies: Cross-fitted tax-consistency residuals and Owen-value Shapley propagation. World Journal of Economics and Business Research, 1(1), 33-45.

[9] Dao, T. (2024). FlashAttention-2: Faster attention with better parallelism and work partitioning. In Proceedings of the International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2307.08691

[10] Xiao, G., Tian, Y., Chen, B., Han, S., & Lewis, M. (2024). Efficient streaming language models with attention sinks. In Proceedings of the International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2309.17453

[11] Yue, X., & Zhu, R. (2024). An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study. World Journal of Management Science, 2(1), 63-72.

[12] Li, Y., Huang, Y., Yang, Y., Zhang, J., Luo, M., Wang, H., & Zhang, Y. (2024). SnapKV: LLM knows what you are looking for before generation. Advances in Neural Information Processing Systems, 37. https://doi.org/10.52202/079017-0722

[13] Cai, Z., Zhang, Y., Gao, L., & Liu, T. (2025). PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In Proceedings of the Conference on Language Modeling (COLM). https://doi.org/10.48550/arXiv.2406.02069

[14] Chevalier, A., Wettig, A., Ajith, A., & Chen, D. (2023). Adapting language models to compress contexts. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 3829-3846). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.232

[15] Rae, J. W., Potapenko, A., Jayakumar, S. M., & Lillicrap, T. P. (2020). Compressive transformers for long-range sequence modelling. In Proceedings of the International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.1911.05507

[16] Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., Hénaff, O., Botvinick, M. M., Zisserman, A., Vinyals, O., & Carreira, J. (2022). Perceiver IO: A general architecture for structured inputs & outputs. In Proceedings of the International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2107.14795

[17] Ghazvininejad, M., Levy, O., Liu, Y., & Zettlemoyer, L. (2019). Mask-Predict: Parallel decoding of conditional masked language models. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 6112-6121). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1633

[18] Lou, A., Meng, C., & Ermon, S. (2024). Discrete diffusion modeling by estimating the ratios of the data distribution. In Proceedings of the 41st International Conference on Machine Learning (ICML), PMLR 235 (pp. 32819-32848). https://doi.org/10.48550/arXiv.2310.16834

[19] Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., & Li, J. (2024). LongBench: A bilingual, multitask benchmark for long context understanding. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL) (pp.

3119-3137). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.172

[20] Leviathan, Y., Kalman, M., & Matias, Y. (2023). Fast inference from transformers via speculative decoding. In Proceedings of the 40th International Conference on Machine Learning (ICML), PMLR 202 (pp. 19274-19286). https://doi.org/10.48550/arXiv.2211.17192

[21] Chen, J., Han, X., Zhang, X., & Yang, Y. (2023). Exact concept-level explanations for early detection of financial misstatements. Eurasia Journal of Science and Technology, 5(2), 34-48.

[22] Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., & Weller, A. (2021). Rethinking attention with performers. In Proceedings of the International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2009.14794

[23] Press, O., Smith, N. A., & Lewis, M. (2022). Train short, test long: Attention with linear biases enables input length extrapolation. In Proceedings of the International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2108.12409

[24] Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., & Liu, Y. (2024). RoFormer: Enhanced transformer with rotary position embedding. Neurocomputing, 568, Article 127063. https://doi.org/10.1016/j.neucom.2023.127063

[25] Teng, D., Rhee, M., Qin, Y., Zi, B., & Liu, W. (2026). SW-SpeedDLM: Sliding window speculative decoding for diffusion language models under long context constraints. Mathematics, 14(12), Article 2137. https://doi.org/10.3390/math14122137

[26] Bolya, D., Fu, C.-Y., Dai, X., Zhang, P., Feichtenhofer, C., & Hoffman, J. (2023). Token merging: Your ViT but faster. In Proceedings of the International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2210.09461

[27] Shang, W., Teng, D., Chen, T., Zhang, F., & Ding, J. (2026). Neuro-Elastic: A unified framework for hardware-aware adaptive quantization and dynamic sparsity in real-time edge intent prediction. IEEE Access, 14, 1-16. https://doi.org/10.1109/ACCESS.2026.3715276

[28] Shen, Z., Zhao, W., Wang, B., Wang, Z., & Shang, W. (2026). CAGR: A cross-accelerator graph optimization framework for efficient recommender system inference. IEEE Access, 14, 1-15. https://doi.org/10.1109/ACCESS.2026.3671798