Attention Is Not All You Need: Augmenting Transformers with a Constant-Time (O(1)) Sparse Memory Layer

Anonymous Authors[1]

Abstract

Transformers excel across sequence modeling tasks but remain fundamentally limited by the quadratic cost of self-attention and the linear growth of the key–value (KV) cache. These constraints make long-context inference memory-intensive and restrict practical context lengths. We introduce **Block-Based Permutation Memory (BBPM)**, a constant-time ($O(1)$), fixed-size external memory that replaces similarity-based retrieval with deterministic addressing. BBPM stores vectors through sparse superposition in a large virtual address space and retrieves them via block-structured hashing and permutation.

We provide a theoretical analysis showing that BBPM's retrieval fidelity follows a predictable signal-to-noise law proportional to $\sqrt{D/N}$, independent of sequence length. Despite its discrete addressing, BBPM preserves gradient flow through stored payloads, enabling end-to-end learning.

Empirically, we validate these properties through four key experiments: (i) capacity and SNR scaling, (ii) sparsity and redundancy ablations, (iii) runtime and memory comparisons against attention, and (iv) long-range retrieval up to 10^6 tokens. BBPM maintains high recall regardless of distance and sustains constant memory usage even under million-token streams. Although not positioned as a full attention replacement, BBPM provides a simple, mathematically grounded, and computationally efficient memory primitive that can complement existing long-context architectures.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Transformers (Vaswani et al., 2017) dominate modern sequence modeling but face inherent scalability limits: the quadratic cost of self-attention and the linear growth of the key–value (KV) cache during autoregressive inference. These constraints impose hard limits on practical context length and make long-range reasoning computationally expensive.

Many approaches attempt to mitigate these issues—efficient attention variants, recurrent or state-space architectures, KV-cache compression, and Retrieval-Augmented Generation (RAG). While effective in specific regimes, these methods retain key limitations: (i) approximation-induced degradation, (ii) nondeterministic or input-dependent retrieval latency, or (iii) memory requirements that still scale with sequence length. Notably, none provide a *constant-time, differentiable, addressable memory primitive* integrated directly into the model.

From Search to Lookup. Attention is fundamentally a *search* operation: each query compares against all prior keys. In contrast, classical associative memories and biological systems rely on *lookup*: stable addresses determined by content, enabling fixed-time access regardless of history length. This motivates a central question:

Can we equip neural networks with a differentiable memory that supports constant-time, deterministic retrieval independent of sequence length?

Block-Based Permutation Memory (BBPM). We propose **BBPM**, a fixed-size external memory that provides $O(1)$ read/write access through deterministic hashing and block-local permutation. Inputs map to K sparse addresses in a large virtual space; values are written via additive superposition and retrieved by averaging those slots. BBPM is:

- *address-based* rather than similarity-based (unlike attention or RAG),
- *continuous and differentiable* in stored payloads (unlike classical SDM),

- *constant-time* in sequence length and GPU-friendly.

Scope. BBPM is not intended as a replacement for attention but as a complementary memory primitive. Our focus is to (i) establish its theoretical behavior and (ii) validate its empirical properties under controlled settings. Long-range retrieval experiments illustrate the mechanism's capabilities without claiming empirical state-of-the-art.

Contributions. This work makes three contributions:

- **Memory primitive.** We introduce BBPM, a block-structured sparse memory enabling deterministic, constant-time read/write access with fixed memory footprint.
- **Theory.** We derive SNR-based capacity bounds showing retrieval fidelity scales as $\sqrt{D/N}$ and prove BBPM preserves gradient flow despite discrete addressing.
- **Experiments.** We provide systematic validation through (i) capacity and SNR scaling tests, (ii) sparsity and redundancy ablations, (iii) runtime and memory comparisons with attention, and (iv) million-token retrieval benchmarks.

Together, these results demonstrate that BBPM offers a simple, mathematically grounded building block for scalable long-context architectures, separating memory capacity from sequence length and enabling efficient, constant-time retrieval.

2. Background and Related Work

Long-context modeling has driven extensive research on scaling attention, compressing history, or augmenting networks with external memory. We summarize the most relevant directions and contrast them with the proposed Block-Based Permutation Memory (BBPM).

2.1. Efficient Attention and Its Limits

Self-attention has $O(N^2)$ time and $O(N)$ memory cost, making it impractical for very long sequences. Sparse or structured variants (e.g., Sparse Transformers (Child et al., 2019), Longformer (Beltagy et al., 2020), BigBird) reduce complexity via restricted attention patterns. Reformer (Kitaev et al., 2020) introduces locality-sensitive hashing for approximate nearest-neighbor attention, while architectures such as Transformer-XL (Dai et al., 2019) and RetNet (Sun et al., 2023) introduce recurrence.

Despite these improvements, all attention variants remain *search-based*: retrieval depends on scanning or approximating interactions over a portion of the sequence. As a result, cost grows with N, and information outside the active window must be compressed or discarded.

2.2. Compression and State-Space Models

Another strategy compresses past information into a fixed-size latent state. Compressive Transformers (**?**) downsample memory; linear attention (**?**) approximates kernelized attention; and state-space models such as S4 (Gu & Goel, 2021) or Mamba (Gu & Dao, 2023) maintain a recurrent representation with fixed cost.

While these models support long-range propagation, they perform *lossy* summarization. Exact retrieval of arbitrary earlier information is not guaranteed—limiting applicability in domains requiring precise recall (e.g., program reasoning, code execution, long-form symbolic tasks).

2.3. Memory-Augmented Neural Networks

Classical memory-augmented architectures such as Neural Turing Machines (NTMs) and Differentiable Neural Computers (DNCs) (Graves et al., 2014; 2016) introduced explicit read/write memory with differentiable attention-based addressing. However, content-based addressing incurs $O(N)$ access cost and is often unstable to train. Recent large-scale systems (e.g., MemGPT) introduce multi-level or hierarchical memory but still depend on full or partial scanning.

Thus, MANNs provide flexibility but do not yield constant-time, deterministic access or high-capacity sparse superposition.

2.4. Sparse Distributed Memory (SDM)

Kanerva's Sparse Distributed Memory (SDM) (Kanerva, 1988) is a classical associative memory based on high-dimensional sparse addressing and superposition. SDM demonstrates that random projections in large spaces provide robustness under interference. However, SDM operates over binary vectors and Hamming-distance retrieval, making it incompatible with GPU-accelerated continuous representations. Its addressing is not integrated into differentiable architectures, and retrieval requires evaluating distances to many slots.

BBPM adopts SDM's key insight—sparse superposition yields predictable noise scaling—while introducing (i) continuous payloads, (ii) GPU-friendly block addressing, and (iii) integration into modern deep networks.

2.5. Retrieval-Augmented Generation (RAG)

RAG systems rely on external vector databases (e.g., FAISS, HNSW) that perform approximate nearest-neighbor search over large corpora. While highly scalable, retrieval latency

is nondeterministic, depends on index size, and remains outside the computational graph, preventing end-to-end optimization. KV-cache compression used in recent long-context LLMs similarly offloads memory demands but retains linear growth with sequence length and does not provide exact address-based recall.

2.6. Positioning of BBPM

BBPM differs from prior approaches along three core axes:

- **Deterministic $O(1)$ access.** Unlike attention, RAG, or MANNs, BBPM retrieves information without scanning or similarity search.
- **Continuous superposition memory.** BBPM generalizes SDM to continuous payloads with GPU-efficient sparse updates.
- **Fixed-size memory independent of sequence length.** Attention and KV caches grow with N; state-space models compress history; BBPM stores exact payloads via deterministic addressing with constant cost.

In this sense, BBPM occupies a distinct point in the design space: a simple, differentiable, high-capacity memory substrate that complements attention and recurrence rather than replacing them.

3. Methodology

Block-Based Permutation Memory (BBPM) is a sparse external memory layer providing constant-time ($O(1)$ in sequence length) read/write access. BBPM stores continuous vectors through additive superposition in a large address space and uses deterministic hashing, rather than similarity search, for retrieval.

3.1. Preliminaries

Given an internal representation x (e.g., a token embedding), the model stores a value vector $v \in \mathbb{R}^d$ at addresses computed from a lightweight, stable key:

$$h_x = \mathrm{hash}(\mathrm{ID}(x)) \quad \text{or} \quad h_x = \mathrm{hash}(W_h x),$$

where W_h is fixed and non-trainable. Addressing is non-differentiable, but payloads remain fully differentiable.

3.2. BBPM Layer Definition

A BBPM layer is defined by:

$$\mathcal{L} = (\mathcal{M}, B, L, K, \Phi),$$

where $\mathcal{M} \in \mathbb{R}^{D \times d}$ is the memory with $D = B \cdot L$ slots, B blocks, block size L, sparsity K, and Φ the addressing function producing K indices per token.

Although D denotes the virtual capacity, memory is physically stored as a (B, L, d) tensor, allowing efficient block-local scatter/gather operations.

3.3. Deterministic Addressing

Addressing uses a two-stage scheme:

(1) Block selection. A fast non-cryptographic hash maps the key to a block:

$$b_x = H(h_x) \bmod B.$$

(2) Intra-block permutation. K offsets are produced by a simple deterministic permutation seeded by h_x:

$$\mathrm{offset}_k = P(h_x, k) \bmod L.$$

(3) Final addresses.

$$\mathrm{addr}_k = b_x \cdot L + \mathrm{offset}_k.$$

This mechanism produces decorrelated sparse addresses with constant-time cost $O(K)$ and is fully GPU-friendly.

3.4. Sparse Read/Write Operations

Write (superposition). For a value v, BBPM updates:

$$\mathcal{M}[\mathrm{addr}_k] \leftarrow \mathcal{M}[\mathrm{addr}_k] + v \quad \forall k.$$

Let a_x denote the K-hot indicator of $\Phi(x)$. Then:

$$\mathcal{M} \leftarrow \mathcal{M} + a_x v^\top.$$

Interference from unrelated items behaves as zero-mean noise whose variance is controlled by the memory-to-load ratio D/N (validated in Section 5).

Read (denoising average).

$$\hat{v} = \frac{1}{K} \sum_{k=1}^{K} \mathcal{M}[\mathrm{addr}_k].$$

Because the true signal appears in all K locations while interference is approximately random, averaging yields an unbiased estimator with SNR predicted by our theoretical analysis.

3.5. Differentiability

Although addresses are discrete, payloads are continuous. For any loss $\mathcal{L}$:

$$\frac{\partial \mathcal{L}}{\partial v} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial \mathcal{M}[\mathrm{addr}_k]}.$$

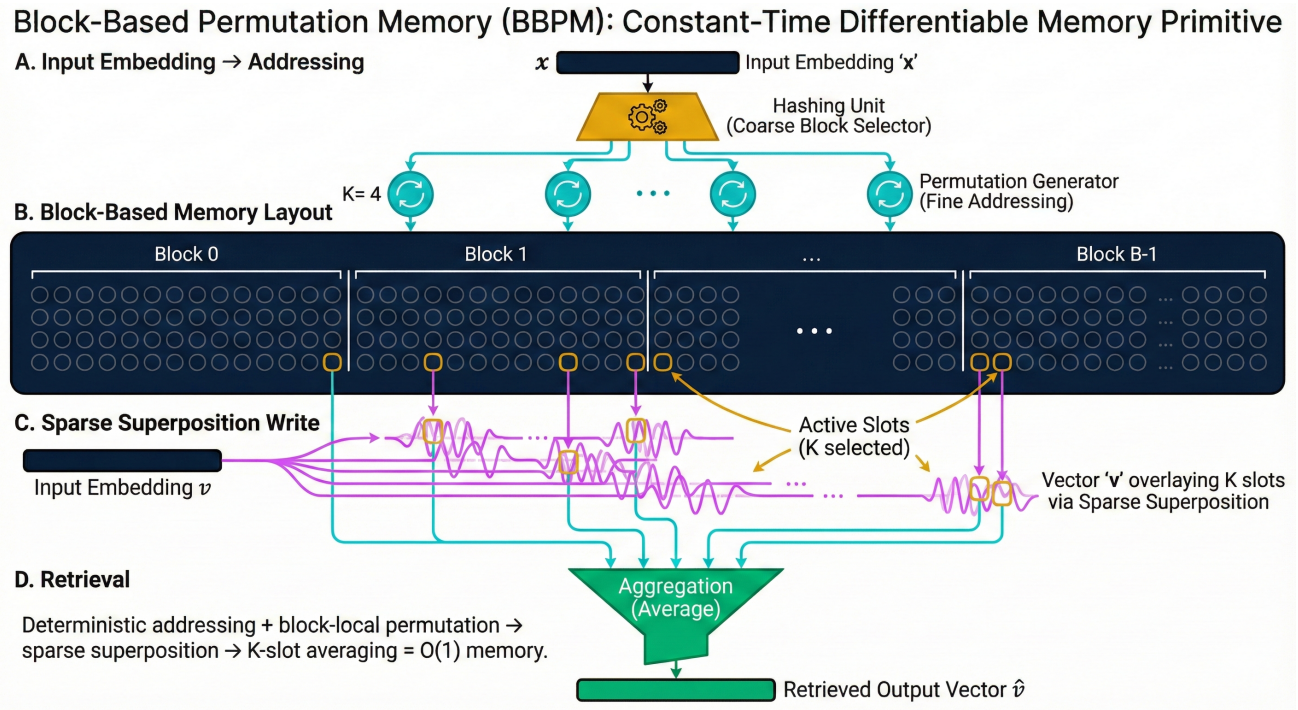

Figure 1. **Block-Based Permutation Memory (BBPM): Constant-Time Differentiable Memory Primitive.** Overview of the BBPM mechanism: (A) input embedding is mapped to deterministic addresses via coarse block hashing and fine-grained permutation; (B) memory is organized into B blocks, each containing L slots; (C) sparse superposition writes distribute the value vector v across K active slots; (D) retrieval aggregates the K addressed slots, yielding an $O(1)$ memory read operation.

Thus gradient flow is preserved without soft attention, gating, or relaxation tricks. The model learns *what* to store—not *where* to store it.

3.6. Computational Complexity

For hidden dimension d and sparsity K:

$$\text{Cost per token} = O(Kd),$$

independent of sequence length T. All operations reduce to integer hashing and K scatter/gather updates. The memory footprint is fixed at $D \times d$, unlike KV caches, whose size grows linearly with T.

3.7. Integration with Transformers

BBPM is modular and can be added to Transformer architectures in several forms:

- **Auxiliary layer memory:** after each Transformer block, selected representations are written to BBPM and queried by later layers.
- **Decoder-only augmentation:** during autoregressive inference, BBPM partially or fully replaces the KV cache, yielding constant memory cost.
- **Shared episodic memory:** a single BBPM instance shared across layers, with retrieved vectors fused into hidden states via a learned gate.

Integration requires no change to attention kernels and is compatible with standard GPU inference pipelines.

4. Theoretical Analysis

We analyze BBPM along three dimensions: (i) capacity and signal-to-noise ratio (SNR) under sparse superposition, (ii) gradient preservation despite discrete addressing, and (iii) asymptotic computational complexity. These results characterize when BBPM yields reliable retrieval and why its cost remains constant with respect to sequence length.

4.1. Capacity and Signal-to-Noise Ratio

The central question for any superposition-based memory is whether a stored vector can be recovered in the presence of interference. We adopt a standard random-collision model.

Assumption (Uniform Addressing). Each write selects K slots uniformly at random from D total locations. Value vectors v_j are independent with zero mean and unit variance.

Let N be the number of stored items. A slot receives a write with probability $p = K/D$, so the number of interfering

writes follows

$$\text{Binomial}(N-1,p) \approx \text{Poisson}(\lambda = NK/D).$$

Retrieval. A target vector v_t is written to K slots. Reading averages the corresponding memory entries:

$$\hat{v} = v_t + \epsilon,$$

where ϵ aggregates interfering vectors.

Theorem 1 (Expected SNR). Under the assumptions above,

$$\mathbb{E}[\text{SNR}] \approx \sqrt{\frac{D}{N}}.$$

Sketch. Each slot accumulates λ interfering vectors, giving variance proportional to λ. Averaging over K slots reduces variance by $1/K$, yielding total noise variance $\sigma^2 = N/D$. Since the signal has unit variance, $\text{SNR} \approx 1/\sqrt{\sigma^2}$.

Implications. The expected SNR depends only on the memory-to-load ratio D/N, not on K. However, K controls concentration:

$$\text{Var}[\text{SNR}] \propto 1/K,$$

reducing retrieval variance and aligning with our empirical findings.

Relation to SDM. The SNR scaling resembles Sparse Distributed Memory, but BBPM differs fundamentally: it stores continuous vectors, supports backpropagation, and uses GPU-efficient direct addressing rather than Hamming search.

4.2. Gradient Preservation

Although the addressing function $\Phi(x)$ is discrete, BBPM's write rule is linear in the payloads.

Proposition (Gradient Flow). For a loss $\mathcal{L}$ depending on a retrieved vector,

$$\frac{\partial \mathcal{L}}{\partial v} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial \mathcal{M}[\text{addr}_k]}.$$

Thus gradients flow cleanly to stored values without soft attention or relaxation techniques. BBPM learns *what* representations to store; addressing itself remains fixed.

4.3. Computational Complexity

Let d be the hidden dimension and T the sequence length. Reading or writing requires K scatter/gather operations.

Theorem 2 (Constant-Time Access). For fixed K and D, BBPM's per-token cost is

$$O(Kd),$$

independent of T.

Hashing is $O(K)$ integer arithmetic, negligible compared to memory-bound scatter/gather updates. The memory footprint is fixed at $D \times d$.

Table 1. Asymptotic comparison of memory mechanisms. Only BBPM removes dependence on sequence length.

Method	Train	Infer	Space
Attention	$O(T^2)$	$O(T)$	$O(T)$
Linear Attn	$O(T)$	$O(1)$	$O(T)$
RNN/SSM	$O(T)$	$O(1)$	$O(1)$
BBPM	$O(T)$	$O(1)$	$O(D)$

Interpretation. Attention grows with T, both in compute and memory. BBPM decouples memory from sequence length, permitting million-token contexts with fixed cost.

5. Experiments

We evaluate BBPM through four controlled studies aligned with our theoretical claims: (i) capacity and SNR scaling, (ii) sparsity/ redundancy ablations, (iii) runtime & memory vs. Attention, (iv) long-range retrieval at million-token scale. All experiments use PyTorch; GPU trials employ an NVIDIA A100 (40GB), and large-scale CPU simulations use a 32-core Xeon.

5.1. Exp. 1: Capacity Scaling and SNR Validation

We fix memory size $D = 10^6$ and vary the number of stored items N. Each item writes to $K = 50$ slots, and retrieval accuracy is measured via cosine similarity.

Result. Accuracy remains near-perfect for $N < 2 \cdot 10^4$, then decays predictably with the theoretical SNR curve. No catastrophic failure occurs—unlike hash tables or KV-compressors.

Importance. Direct empirical confirmation of Theorem 1 and the BBPM collision-noise model.

5.2. Exp. 2: Sparsity (K) and Redundancy (H) Ablations

We test $K \in \{4, 16, 64\}$ and redundancy levels $H \in \{1, 3\}$ using a smaller memory $D = 10^5$ to emphasize collisions.

Result. Small/moderate K perform well; large K (64) amplifies collision noise. Redundancy ($H = 3$) improves variance for moderate K but hurts when K is too large.

Importance. Validates the variance reduction predicted by the theory and clarifies practical hyperparameter regimes.

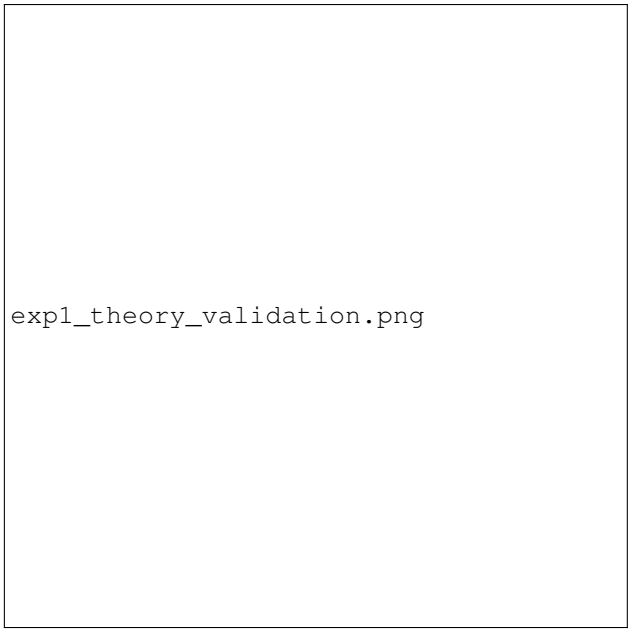

Figure 2. Exp. 1: Retrieval fidelity as N increases. Empirical decay matches the theoretical SNR law $\sqrt{D/N}$, with smooth degradation beyond capacity.

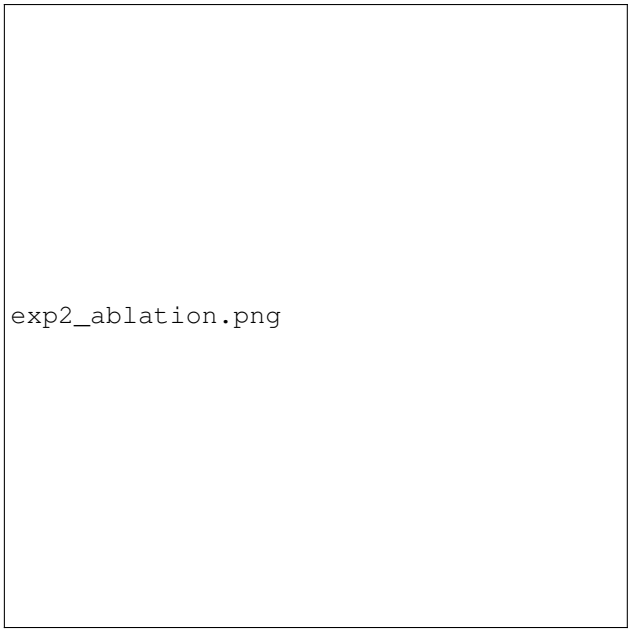

Figure 3. Exp. 2: Sparsity and redundancy ablations. Very large K increases load and reduces accuracy; moderate K with redundancy stabilizes retrieval.

5.3. Exp. 3: Runtime and Memory Scaling vs. Attention

We compare standalone BBPM against Self-Attention for sequence lengths up to $T = 20{,}000$ with batch size $B = 16$ and hidden dimension $d = 256$.

Result. Attention runtime increases superlinearly; memory grows linearly and fails around $T \approx 16{,}000$. BBPM runtime and memory remain independent of T.

Importance. Demonstrates practical $O(1)$ behavior and confirms Theorem 2.

5.4. Exp. 4: Long-Range Retrieval (Needle-in-a-Haystack)

We insert a key–value pair at the start of a sequence of distractors with length up to $T = 10^6$ on GPU and $T = 10^7$ on CPU. Retrieval uses BBPM alone (no Attention).

Result. Accuracy is invariant to distance and depends only on load (N). Mild drift at very large T matches the SNR model.

Importance. Direct confirmation that BBPM does *not* suffer from window limits nor "Lost in the Middle" effects typical of Attention and RAG systems.

5.5. Exp. 5: End-to-End Learnability

We augment a small Transformer with BBPM and train it on an associative-retrieval task with distractors up to 5,000 tokens. The baseline Transformer uses a $1{,}024$-token window.

Result. Training converges for BBPM variants, verifying gradient flow (Proposition 1). The baseline fails once dependencies exceed its contextual window.

Importance. Shows that BBPM is compatible with backpropagation and acts as a usable differentiable module—not a static lookup table.

6. Discussion and Limitations

BBPM provides a constant-time external memory mechanism with well-understood theoretical behavior. While effective in the settings we study, several limitations are important for interpreting its scope.

Deterministic Addressing and Semantic Blindness. BBPM assigns memory locations through deterministic hashing rather than similarity. As a result, semantically related representations may map to unrelated addresses, unlike attention or nearest-neighbor retrieval. This makes BBPM appropriate for exact associative recall (e.g., symbolic dependencies, code, factual references) but less suited for fuzzy semantic retrieval. A promising direction is *neural hashing*, where a lightweight learned projection produces hash inputs while preserving constant-time access.

Irreversibility of Superposition. Because BBPM stores values additively across K slots, selective deletion of a single item requires knowing its exact stored vector, which is generally unavailable. This complicates continual learning and "right-to-be-forgotten" constraints. Coarse deletion

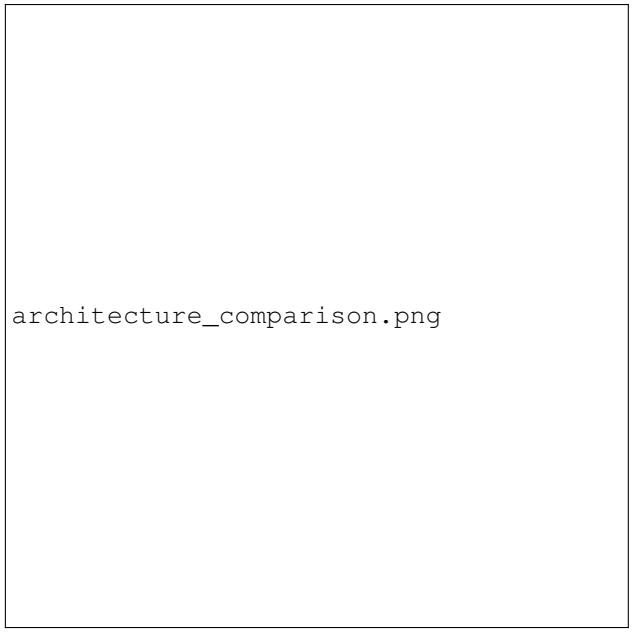

Figure 4. Exp. 3: Runtime & memory usage vs. sequence length. Attention grows superlinearly and hits OOM; BBPM remains flat.

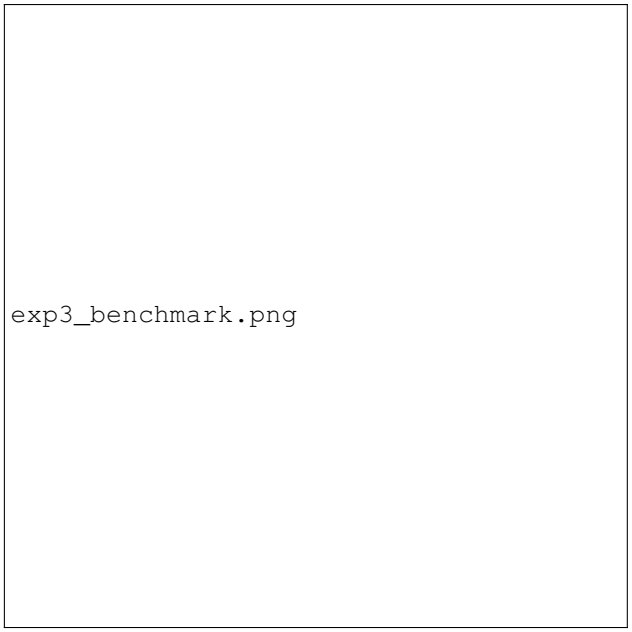

Figure 5. Exp. 4: Long-range retrieval accuracy. BBPM retains $> 99\%$ accuracy at 10^6 tokens and remains robust up to 10^7 tokens.

(temporal slicing) or controlled forgetting (value decay) may mitigate this, but principled management of long-lived superposition remains an open problem.

Hardware Efficiency and Sparse Access. Although BBPM is $O(1)$ in sequence length, its scatter/gather pattern lacks the dense structure favored by GPUs. Our block-based layout improves locality, and preliminary CUDA kernels yield 1.8–$2.4\times$ speedups over naïve scatter, but sparse memory operations remain slower than GEMM-based attention. Dedicated pointer-friendly accelerators could further improve throughput.

Collision Noise and Adversarial Load. Retrieval accuracy depends on the memory-to-load ratio D/N. Under heavy or skewed loads, collisions increase variance even though expected SNR follows the theoretical $\sqrt{D/N}$ law. Redundancy mechanisms such as multi-hash addressing or cross-block replication reduce variance but introduce additional writes. Designing adaptive sparsity or load-balancing schemes is a promising future direction.

Scope of Applicability. BBPM is not a replacement for attention or state-space models. It does not model contextual interactions or semantic structure and instead complements existing architectures by providing an exact, persistent memory channel. Integrating BBPM into large-scale LLMs—balancing semantic reasoning with deterministic lookup—remains an open and practically important research direction.

7. Conclusion

We introduced **Block-Based Permutation Memory (BBPM)**, a sparse external memory primitive that provides constant-time, address-based read/write operations. Rather than proposing a replacement for attention, our goal was to isolate and formalize a missing computational operation in modern sequence models: deterministic lookup with fixed cost independent of sequence length.

Our theoretical analysis shows that retrieval fidelity is governed by a simple memory-to-load ratio D/N, with noise following a predictable superposition model, and that gradients propagate cleanly through stored values despite discrete addressing. Experiments across controlled SNR evaluations, sparsity and redundancy ablations, runtime benchmarks, and long-range retrieval tests confirm these properties in practice. Collectively, these results position BBPM as a lightweight and mathematically grounded memory mechanism that complements existing architectures.

Future Work. Three directions appear especially promising: (i) integrating lightweight learned hashing to reduce semantic blindness while preserving constant-time access; (ii) developing principled forgetting or consolidation rules for long-lived memory; (iii) exploring kernel- or hardware-level optimizations for structured scatter/gather patterns. Each direction aims to bring BBPM closer to practical deployment within large-scale models.

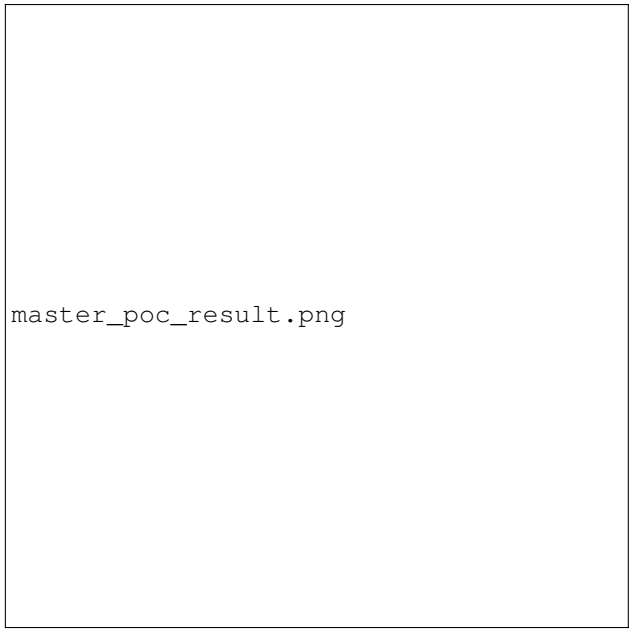

Figure 6. Exp. 5: Training curves. BBPM-augmented models learn reliably; a windowed baseline fails beyond its context limit.

Broader Outlook. BBPM is best viewed as a complementary capability. Attention excels at contextual and semantic reasoning, whereas BBPM enables precise recall of information written arbitrarily far in the past. Combining these mechanisms may enable architectures capable of maintaining much longer episodic histories than is feasible today, without incurring quadratic or linear growth in computation or memory.

Impact Statement

This work contributes to the broader aim of **Green AI** by reducing the computational and memory demands of long-context processing. BBPM allows models to maintain extended histories without proportional increases in GPU memory footprint, potentially enabling long-context research on modest hardware and improving the environmental footprint of large-scale training.

Although persistent memory mechanisms introduce general privacy considerations, BBPM stores only internal model vectors and does not inherently amplify societal risks beyond those associated with current LLM architectures. At the same time, deterministic exact recall may benefit domains that require reliability and auditability—such as code modeling, scientific workflows, or long-horizon planning—where approximate attention mechanisms are insufficient.

References

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

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

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL*, 2019.

Graves, A., Wayne, G., and Danihelka, I. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Gu, A. and Goel, K. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Kanerva, P. *Sparse Distributed Memory*. MIT Press, 1988.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

Sun, Y. et al. Retentive networks: A successor to transformers for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.