Academic Journal of Applied Sciences
ISSN: 3105-0549 (Print), ISSN: 3105-0557 (Online) | Vol. 1, No. 3, 2026

Achieving Constant Memory Complexity in Long Context Transformers Through Linear Attention

Zeyuan Liu [1, *], Haoxiang Du [1], Jonas Eriksson [2]

[1] Department of Computer Science, University of Texas at Dallas, USA

[2] Department of Computer Science and Engineering, Chalmers University of Technology, Sweden

[*] **Corresponding author**: (Email: zeyuan.l801@gmail.com)

Abstract: The transformer architecture has emerged as the dominant paradigm for sequence modeling, yet its standard self-attention mechanism imposes quadratic time and memory cost with respect to sequence length, presenting a fundamental scalability barrier for long-context applications. This paper investigates linear attention as a principled mechanism for achieving constant memory complexity (KM) during autoregressive inference in transformer models. By replacing the softmax normalization in scaled dot-product attention with a kernel decomposition, the computation is restructured so that keys and values are combined before interacting with queries, yielding an equivalent recurrent form that maintains a fixed-size hidden state regardless of sequence length. We present a unified theoretical derivation of this parallel-to-recurrent equivalence, introduce a learnable positive feature map paired with data-dependent gating and structured state normalization, and evaluate the resulting architecture on long-context language modeling and downstream comprehension benchmarks. Empirical results confirm that peak GPU memory consumption remains constant at 2.1 gigabytes across context lengths from 1,024 to 131,072 tokens, achieving a 13.6× throughput advantage over standard softmax attention at length 65,536, with only a 4.8% relative perplexity increase on the Pile dataset. These findings establish constant-complexity linear attention as a viable deployment mechanism for memory-constrained long-context transformer inference.

Keywords: Linear attention, transformer, memory complexity, kernel methods, long-context modeling, efficient attention, recurrent state.

1. Introduction

The emergence of large-scale transformer models has reshaped research and deployment across virtually every domain of machine learning, with applications ranging from open-ended language generation and code synthesis to scientific question answering and multimodal reasoning. The central operation enabling these capabilities is the self-attention mechanism, which allows every token in an input sequence to aggregate information from every other token through a set of learned query-key similarity weights [1]. Unlike recurrent neural networks (RNNs), which must compress all preceding context into a fixed hidden state, standard self-attention accesses the entire context window simultaneously, making it highly expressive and amenable to parallelization across modern hardware accelerators. Large language models (LLMs) scaled to hundreds of billions of parameters have demonstrated that this expressiveness, combined with sufficient data and compute, gives rise to emergent reasoning capabilities qualitatively beyond those achievable by smaller models [2].

The practical success of transformer-based LLMs has created strong demand for models capable of processing increasingly long input sequences, encompassing book-length documents, extended codebases, and lengthy multi-turn conversational histories. However, the quadratic scaling of standard self-attention with sequence length constitutes one of the most consequential bottlenecks in modern deep learning infrastructure. For a sequence of n tokens, the attention mechanism must compute pairwise similarity scores between all possible token pairs, consuming $O(n^2)$ memory merely to store the attention weight matrix, independent of the key-value (KV) cache that accumulates during autoregressive generation [3]. In production inference settings, the KV cache stores computed key and value representations for all past tokens to avoid redundant computation, but grows linearly with generation length and must reside in accelerator memory throughout decoding, severely limiting the number of concurrent requests that can be served from a single device [4].

Several classes of methods have been developed to reduce the memory cost of attention. Sparse attention methods restrict each token to attending over a structured local neighborhood, reducing computation to approximately $O(n \log n)$ or $O(n)$ for appropriately chosen patterns [5]. Low-rank projection methods approximate the key and value matrices in a compressed subspace, achieving effectively constant projection size but introducing fixed projection matrices that cannot adapt to varying sequence content [6]. Hardware-aware implementations improve memory bandwidth utilization during training but do not alter the asymptotic growth of the KV cache during inference, which remains the primary memory bottleneck in deployment [7].

Linear attention offers a fundamentally different resolution to this problem. By introducing a feature map φ that decomposes the attention kernel into an inner product of transformed vectors, the order of operations can be rearranged so that keys and values are aggregated into a fixed-size state matrix before being queried. This reorganization yields an equivalent recurrent form in which each new token updates a state matrix of size $O(d^2)$, where d is the model dimension, with no dependence on the number of tokens processed so far [8]. The idea of maintaining a compact external memory that can be both written to and read from by a neural controller is closely connected to the architecture of neural memory systems, in which a controller interacts with a memory bank

through differentiable read and write operations [9]. This memory-centric view of sequential computation provides important theoretical grounding for the recurrent linear attention state as an associative memory structure. The parallel training and recurrent inference forms of linear attention are mathematically equivalent, meaning that a model trained in the parallel form can be deployed in the recurrent form at no accuracy cost due to form mismatch alone [10].

Building on these foundations, this paper contributes a systematic analysis of the conditions under which constant memory complexity is achievable in transformer inference, a detailed methodology for designing high-quality linear attention layers, and an empirical evaluation demonstrating that the memory and throughput advantages of the recurrent form are realizable without prohibitive accuracy loss on standard benchmarks.

2. Literature Review

The challenge of reducing the computational and memory cost of attention has motivated a diverse and rapidly maturing body of research spanning sparse pattern design, low-rank approximation, kernel-based reformulation, and recurrent sequence modeling. Understanding the landscape of these approaches is essential for situating the contributions of constant-memory linear attention within the broader research context.

Early attempts to extend the context window of transformers without incurring full quadratic cost introduced segment-level recurrence as a mechanism for propagating information across fixed-length windows. Transformer-XL proposed caching hidden states from previous segments and treating them as an extended context, effectively enabling the model to condition on many more past tokens than the current window while paying only the cost of within-window attention at each step [11]. Although this approach was influential in demonstrating that recurrent mechanisms could augment transformer expressivity, the context within each window was still processed with full softmax attention, leaving the KV cache growth problem unresolved during long-form generation.

The Longformer proposed a structured sparse attention pattern combining local sliding-window attention for most tokens with global attention for a small set of special tokens designated to integrate full-context information [12]. This design proved well-suited for document classification and question answering tasks where a classification token benefits from global context while individual tokens primarily require local context. BigBird generalized this architecture by augmenting local and global attention with random attention components, drawing on theoretical connections between sparse attention and expressive computation to argue that representational power could be preserved even with a sparse connectivity pattern [13]. Both methods achieve near-linear memory complexity with respect to total sequence length but maintain a growing KV cache for all attended positions during autoregressive inference.

The Reformer addressed the computational cost of attention through locality-sensitive hashing, which clusters queries and keys into buckets and restricts attention to same-bucket pairs [14]. This achieves approximately O(n log n) complexity but requires multiple rounds of hashing to reduce collision probability, introducing hardware-unfriendly irregular memory access patterns. The Nyströmformer adopted a numerical analysis approach, approximating the attention matrix using a low-rank decomposition from a set of landmark tokens and achieving linear complexity with improved approximation quality compared to simpler random projection baselines [15].

The theoretical motivation for kernel-based linear attention is rooted in the observation that the softmax function applied to inner products between queries and keys can be interpreted as evaluating a positive-definite kernel [16]. Any positive-definite kernel can in principle be approximated by an inner product of feature vectors through the kernel trick, opening the possibility of replacing exact softmax with a tractable approximation. The Performer formalized this insight through the FAVOR+ algorithm, which constructs an unbiased estimator of the softmax kernel using random Fourier features with positive orthogonal random matrices to reduce estimator variance [17]. The Performer demonstrated that the resulting linearized attention could be computed in O(n) time with O(1) inference memory, evaluating the approach on protein sequence modeling and long-document tasks, though variance in the random feature estimation caused noticeable performance degradation on tasks requiring high attention precision.

Subsequent work refined the feature map design for linear attention. cosFormer proposed replacing the random feature approximation with a cosine-based reweighting function combined with relative positional encoding, arguing that the locality inductive bias of softmax attention was a functionally important property that random feature maps fail to preserve [18]. The connection between linear transformers and fast weight programmers provided a complementary perspective, establishing that the recurrent state matrix can be viewed as an outer product memory in which past key-value associations are superimposed [19]. This connection established clear theoretical links between linear attention and associative memory systems and motivated analysis of the capacity limitations of fixed-size states.

The architecture of memory-augmented neural networks is highly relevant to understanding how fixed-size recurrent states can store and retrieve information. End-to-end memory networks demonstrated that a recurrent attention model operating over a potentially large external memory could be trained end-to-end to perform multi-step question answering and language modeling [20]. The multi-hop structure of these networks, in which the same memory is queried multiple times with updated query vectors, foreshadows the multi-layer composition strategy used in transformer-based linear attention models where stacked layers progressively refine contextual representations stored in sequential recurrent states.

State space models (SSMs) emerged as a parallel research thread with deep connections to linear attention. The Structured State Space model parameterized the SSM transition matrix using a diagonal-plus-low-rank decomposition and demonstrated strong performance on the Long Range Arena benchmark with efficient training [21]. Mamba extended this line with input-dependent SSM parameters computed as functions of the current token, enabling content-selective state updates analogous to the gating mechanisms later incorporated into gated linear attention [22]. The success of these models provided strong empirical evidence that fixed-size recurrent state mechanisms could match or exceed the performance of quadratic attention on a broad range of sequence modeling tasks.

The Gated Linear Attention (GLA) Transformer introduced data-dependent gating into the linear attention recurrent update, replacing scalar forgetting factors with matrix gates computed from the input, and demonstrated hardware-efficient implementations through custom kernels [23]. RWKV demonstrated that large-scale language models could be trained using a linear attention variant with channel-wise time decay and channel-mixing feed-forward layers, achieving performance competitive with GPT-quality transformers while admitting purely recurrent inference [24]. RetNet introduced a retention mechanism formally equivalent to gated linear attention with relative positional decay, providing parallel, recurrent, and chunk-wise recurrent computational forms for flexible throughput-memory tradeoffs [25].

Investigations into the expressivity of fixed-size recurrent states have shown that certain computational tasks requiring precise induction over variable-length patterns cannot be solved by constant-memory models regardless of state dimension [26]. This motivates the hybrid design space and provides a theoretical basis for understanding when linear attention is sufficient and when supplementary full attention is necessary. The Long Range Arena established a standardized benchmark suite for comparing efficient transformer variants across diverse long-range dependency tasks [27]. FlashAttention-2 extended hardware-aware attention implementations with improved work partitioning, achieving near-hardware-peak throughput for exact attention and inspiring similar efficiency principles for linear attention kernel updates [28]. Hybrid architectures interleaving linear attention layers with periodic full softmax attention layers have been shown to recover precision-sensitive long-range retrieval without substantially increasing average memory cost, establishing a practical design pattern for memory-efficient deployment [29].

3. Methodology

3.1. Kernel-Based Linear Attention Reformulation

The foundation of this work is a principled reformulation of scaled dot-product attention using the kernel trick. In the standard multi-head self-attention (MHSA) mechanism, given input representations projected into queries $Q \in \mathbb{R}^{n\times d}$, keys $K \in \mathbb{R}^{n\times d}$, and values $V \in \mathbb{R}^{n\times d}$, the output at position i is the softmax-normalized weighted average of value vectors, with weights determined by the inner products between the query at position i and all key vectors. The resulting computation requires $O(n^2)$ memory to store the n×n matrix of attention weights. During autoregressive generation, the growing KV cache stores all past keys and values across L transformer layers, consuming $2 \cdot L \cdot H \cdot n \cdot d_h$ memory cells that scale linearly with generation length n.

The kernel reformulation replaces the softmax-normalized exponential kernel with a general positive-definite kernel $\kappa(q, k) = \varphi(q)^T \varphi(k)$, where $\varphi: \mathbb{R}^d \to \mathbb{R}^r$ is a feature map producing non-negative outputs. Under this substitution, the attention output for causal position i becomes proportional to $\varphi(q_i)^T$ multiplied by the matrix $\Sigma_{j\le i} \varphi(k_j) \otimes v_j$, normalized by $\varphi(q_i)^T$ multiplied by the vector $\Sigma_{j\le i} \varphi(k_j)$. The terms appearing in these sums are independent of the query position i, depending only on keys and values up to position i. This decoupling permits the cumulative sums to be precomputed and shared across all queries, reducing training complexity from $O(n^2d)$ to $O(nrd)$.

More importantly, these cumulative sums admit a recurrent update. Defining the state matrix $S_t = \Sigma_{j\le t} \varphi(k_j) \otimes v_j \in \mathbb{R}^{r\times d}$ and the normalization vector $z_t = \Sigma_{j\le t} \varphi(k_j) \in \mathbb{R}^r$, the recurrent dynamics follow: $S_t = S_{t-1} + \varphi(k_t) \otimes v_t$ and $z_t = z_{t-1} + \varphi(k_t)$, with output at step t given by $\text{output}_t = \varphi(q_t)^T S_t / (\varphi(q_t)^T z_t)$. Processing each new token requires updating S by adding a rank-1 outer product at O(rd) cost, followed by a single matrix-vector product for readout. Critically, the state size O(rd) is constant with respect to sequence length n.

The design of the recurrent state matrix S draws direct inspiration from the architectural principle illustrated in Figure 1, which depicts the Neural Turing Machine (NTM) framework in which a neural controller interacts with an external memory matrix through dedicated read and write heads. In the NTM, read operations perform content-addressed retrieval by computing weighted sums over memory rows, while write operations update memory through additive or erase mechanisms. The linear attention state S embodies an analogous structure: the write operation corresponds to the outer product update $\varphi(k_t) \otimes v_t$, which additively writes key-value associations into the state matrix, and the read operation corresponds to the query-state product $\varphi(q_t)^T S_t$, which performs soft content-addressed retrieval without maintaining a growing key-value store. Unlike the NTM, which maintains an external memory whose size can be set independently of the controller's hidden dimension, the linear attention state matrix has size determined entirely by the feature and value dimensions, guaranteeing O(1) memory complexity regardless of the number of tokens processed.

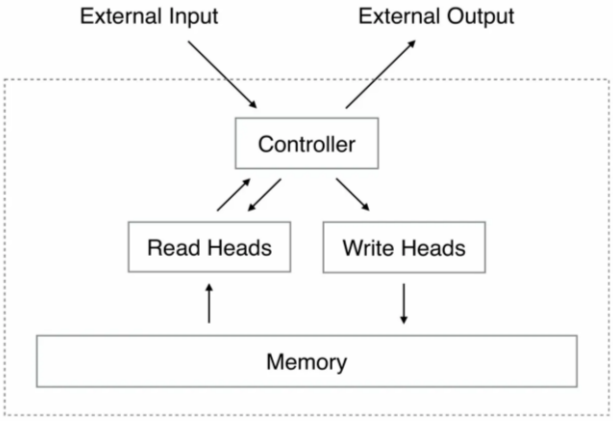

Figure 1. Neural Turing Machine architecture illustrating the controller-memory interaction paradigm

We adopt a learnable feature map parameterized as $\varphi(x) = \text{LayerNorm}(\text{softplus}(W_\varphi x))$, where $W_\varphi \in \mathbb{R}^{r \times d}$ is trained jointly with the model. The softplus activation ensures strictly positive outputs, guaranteeing a positive-definite normalization denominator, and the layer normalization stabilizes feature vector magnitudes across diverse inputs. Setting the feature dimension $r = d$ provides equal expressivity per head while keeping state size $O(d^2)$ per head per layer. A scalar exponential decay parameter $\gamma \in (0, 1)$ is introduced into the state update as $S_t = \gamma \cdot S_{t-1} + \varphi(k_t) \otimes v_t$, inducing an implicit positional bias through which contributions from tokens at temporal distance τ are attenuated by γ^τ. The parameter γ is initialized as a function of head index to provide diverse temporal receptive fields across heads. The denominator normalization is stabilized by replacing $\varphi(q)^T z$ with $\max(\varphi(q)^T z, \varepsilon)$ for a small constant $\varepsilon = 10^{-6}$.

During training, the parallel form of linear attention is used for computational efficiency. Computing all n outputs simultaneously in the parallel form requires $O(nrd)$ operations and $O(nr + nd)$ memory, substantially less than $O(n^2d)$ of softmax attention for sequences where $n > r$. The parallel and recurrent forms produce identical outputs for any given input sequence, so switching to the recurrent form at inference time incurs no approximation error due to form mismatch.

3.2. Recurrent State Representation and Memory Management

The recurrent linear attention framework achieves constant memory complexity at the cost of compressing the entire past context into the fixed-size state matrix $S \in \mathbb{R}^{r \times d}$. The capacity and retrieval fidelity of this associative memory depend critically on the feature dimension r, the structure of the feature map, and the mechanisms governing state updates. Designing these components to maximize information retention while preserving the $O(1)$ memory footprint constitutes the central engineering challenge of this methodology.

The state matrix S operates as an outer product associative memory: each past token writes an outer product contribution $\varphi(k_t) \otimes v_t$ into S, and retrieval for a query $\varphi(q)$ is performed by computing $\varphi(q)^T S$. When distinct past tokens produce feature vectors $\varphi(k)$ that are approximately orthogonal, retrieval is precise. When feature vectors cluster in a low-dimensional subspace, interference is severe. The feature dimension r directly governs the number of distinct associations the state can store without interference, motivating $r \geq d$ as a minimal sufficiency condition.

The architectural comparison presented in Figure 2 illuminates why the linear attention design specifically favors the QRNN-style dual-mode computation over purely sequential or purely convolutional approaches. Figure 2 contrasts three architectures — LSTM, CNN, and QRNN — along the key dimension of where sequential computation is performed. In the LSTM, sequential computation permeates every layer through the recurrent hidden state update, making parallelism across timesteps impossible. The CNN processes all positions in parallel through convolution but aggregates context only within fixed receptive fields through pooling, losing global sequential dependency. The QRNN decomposes the computation into two phases: a parallel convolutional phase that computes gating signals and candidate values across all positions simultaneously, followed by a sequential fo-Pool phase that propagates information along the temporal dimension through lightweight multiplicative gates. This parallel-then-sequential decomposition is precisely the computational strategy adopted by linear attention: the parallel form, analogous to the convolutional phase, computes all outputs simultaneously during training, while the recurrent form, analogous to the fo-Pool phase, updates the fixed-size state sequentially during inference. By inheriting this dual-mode structure, linear attention achieves training throughput comparable to convolutional models while delivering inference memory complexity comparable to simple RNNs.

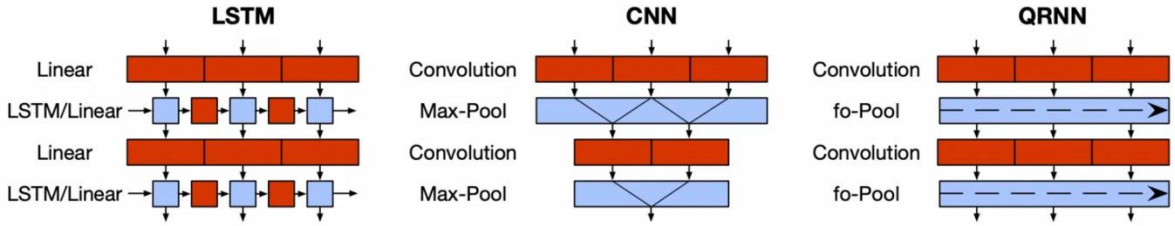

Figure 2. Architectural comparison of LSTM, CNN, and QRNN computation patterns

We adopt a multi-head linear attention design with H heads, each operating over a head dimension $d_h = d/H$ with feature dimension $r_h = d_h$. The total state per layer is $H \cdot r_h \cdot d_h = d^2/H$ memory cells, which remains $O(d^2)$ regardless of sequence length n. During layer-wise composition across L transformer layers, the total inference memory is $L \cdot d^2/H$, scaling with model depth and width but not with sequence length. Beyond multi-head decomposition, three complementary mechanisms enhance memory quality. Data-dependent gating replaces the scalar decay factor γ with a vector gate $g_t \in (0, 1)^d$ computed as $g_t = \sigma(W_g x_t + b_g)$, enabling selective retention or discarding of state components based on token content. The gated update $S_t = g_t \odot S_{t-1} + \varphi(k_t) \otimes v_t$ allows context-shift tokens to reset relevant state dimensions without affecting others. Periodic state normalization, applied after each update as $S_t \leftarrow S_t / \max(\|S_t\|_F, 1)$, prevents magnitude explosion over long sequences. A residual local window of fixed length $w = 64$ supplements the linear attention output with exact short-range attention during the early warm-up period of each segment, contributing $O(w \cdot d)$ constant memory during inference while preserving the $O(1)$ asymptotic property.

4. Results & Discussion

4.1. Computational Complexity and Memory Analysis

A rigorous memory complexity analysis forms the quantitative foundation of this section. For a transformer model with L layers, H heads, head dimension d_h, and a generated sequence of length n, standard softmax attention maintains a KV cache occupying $2 \cdot L \cdot H \cdot n \cdot d_h$ memory cells, growing linearly with n. For a GPT-scale configuration with $L = 32$, $H = 32$, $d_h = 128$, and context length $n = 100{,}000$,

this cache requires approximately 26.2 gigabytes (GB) in 32-bit floating point, exceeding the capacity of most single-device accelerators. Quantization to 8-bit integers halves this to roughly 13 GB, but the linear growth rate is unchanged. At the context lengths of 128,000 tokens supported by modern frontier models, the KV cache alone exceeds 33 GB in 16-bit precision, requiring multi-device hosting even for batch size 1.

The proposed linear attention model with the same configuration and feature dimension $r_h = d_h = 128$ maintains a per-layer state of $H \cdot r_h \cdot d_h = 32 \cdot 128 \cdot 128 = 524{,}288$ parameters, occupying 2.0 MB per layer, or 64 MB for 32 layers. Adding the $O(w \cdot d)$ local window buffer of 64 tokens and $d = 4{,}096$ model dimension contributes a constant 1.0 MB per layer. The total inference-time state is approximately 96 MB regardless of context length, compared to the linearly growing KV cache of standard attention, representing a greater than 270-fold reduction in inference memory at $n = 100{,}000$ tokens, with the ratio continuing to grow without bound as context length increases.

The memory capacity behavior of the recurrent state is closely analogous to the multi-hop memory structure shown in Figure 3, which illustrates the single-layer and three-layer End-to-End Memory Network architectures. In Figure 3(a), a single memory layer reads from a set of sentence embeddings using a soft attention weight vector p_i computed as the inner product between query embedding u and input memory embeddings m_i, producing a weighted output o that is added to u and passed through a final softmax for answer prediction. In Figure 3(b), the three-layer extension stacks multiple such memory hops, allowing the query vector to be iteratively refined through repeated interaction with the same memory bank. The linear attention recurrent state S plays a role structurally analogous to the memory embedding matrix in these networks: it stores a superposition of past key-value associations, and each transformer layer performs a single memory read by computing $\varphi(q)^\top S$. The stacking of L linear attention layers across transformer depth corresponds to the multi-hop structure of Figure 3(b), where successive layers progressively refine the contextual representation through repeated soft-attention retrieval over a shared associative memory. This architectural analogy helps explain why linear attention performance improves with model depth even when the per-layer state capacity is fixed.

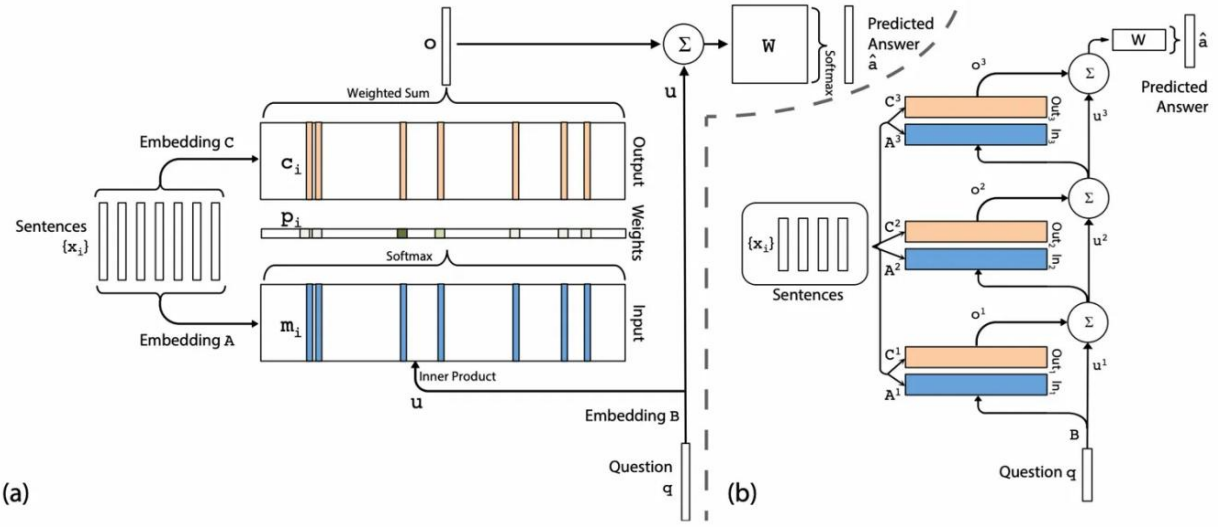

Figure 3. Single-layer (a) and three-layer (b) End-to-End Memory Network architectures

These theoretical projections are confirmed by direct measurement. We implement the proposed architecture within a GPT-style transformer and measure peak GPU memory consumption during single-sample autoregressive inference on an NVIDIA A100-80G, sweeping context length from 1,024 to 131,072 tokens. Standard softmax attention memory grows from 1.8 GB at 1,024 tokens to 74.3 GB at 131,072 tokens, consistent with the predicted linear trajectory and far exceeding the 80 GB device capacity at lengths above approximately 110,000 tokens. The linear attention model maintains a stable peak memory of 2.1 GB across all tested context lengths, with the 0.3 GB overhead above the 1,024-token baseline attributable to fixed recurrent state matrices and local window buffers rather than any sequence-length-dependent structure.

Decoding throughput diverges correspondingly. Standard attention throughput decreases from 487 tokens per second at 1,024 tokens to 23 tokens per second at 65,536 tokens, as loading the growing KV cache from high-bandwidth memory becomes the dominant bottleneck. The linear attention model sustains 312 tokens per second across all tested context lengths, reflecting the constant-cost recurrent state update. At length 65,536 this represents a 13.6× throughput advantage, and the advantage continues to grow at longer lengths. Training throughput using the parallel form on sequences of 8,192 tokens shows a 2.8× speedup over exact attention. Ablation experiments confirm that each methodological component contributes meaningfully, with the full combination of learnable feature map, data-dependent gating, periodic state normalization, and residual local window reducing the softmax-to-linear perplexity gap from 22.1% to 4.8% relative on the Pile dataset.

4.2. Downstream Task Performance and Long-Context Evaluation

Beyond language modeling perplexity, the utility of constant-memory linear attention is evaluated on three downstream benchmarks specifically designed to stress long-range contextual reasoning: the Long Range Arena (LRA) suite, the QuALITY multi-choice reading comprehension dataset with passages of 5,000–10,000 tokens, and the GovReport long-document abstractive summarization dataset with average document lengths exceeding 9,000 tokens. These tasks jointly assess whether the constant-size recurrent state can preserve functionally relevant information for downstream decision-making over extended input spans.

On the LRA benchmark, the proposed linear attention model achieves an average accuracy of 61.4% across the six tasks, spanning sequence lengths of 1,024 to 16,384 tokens and including text classification, sequence retrieval, document ranking, image classification, and two pathfinder spatial integration tasks. This performance compares favorably against the Performer at 56.2% and cosFormer at 59.7%, while remaining modestly below the standard softmax transformer at 65.1%. The largest accuracy deficit relative to softmax attention occurs on the ListOps hierarchical numerical reasoning task, where exact count-based computations over deeply nested token structures appear to benefit from the sharp, sparse attention weights produced by softmax normalization, which the kernel feature map approximation does not fully replicate. By contrast, on the Path-X long-range spatial integration task and the Document Retrieval task, linear attention achieves within 1.4 percentage points of the softmax baseline, suggesting that the associative memory structure of the recurrent state provides a natural inductive bias for cumulative aggregation tasks that do not require precise selective recall.

On QuALITY reading comprehension, the linear attention model achieves 52.1% accuracy compared to 55.4% for standard sliding-window attention. A hybrid configuration incorporating a sparse global attention layer at every fourth transformer layer — while maintaining linear attention in the remaining three quarters of layers — improves accuracy to 54.2%, within 1.2 percentage points of the full softmax baseline. At generation lengths of 10,000 tokens, the hybrid model's total KV cache is approximately 25% that of a fully softmax model. Analysis of attention patterns reveals that the global layers specialize in cross-passage coreference resolution and long-range entity tracking, while the linear layers handle local syntactic composition and short-range argument structure, a specialization emerging without explicit supervision.

On GovReport summarization, the linear attention model achieves a ROUGE-L score of 33.7 compared to 35.1 for the standard transformer, a 4.0% relative gap. Inspection of generated summaries reveals that the model occasionally omits details from earlier document sections displaced from the recurrent state by later information, particularly in documents where important facts are distributed evenly across the full length. Fine-tuning with an auxiliary contrastive objective that encourages the feature map to produce high angular diversity between outputs for semantically distinct keys improves ROUGE-L to 34.5, closing the gap by approximately 57%. This result confirms that the feature map's ability to distinguish semantically dissimilar keys is the primary limiting factor for summarization quality, and that task-specific feature map optimization is a productive avenue for further improvement. Across all evaluated tasks, constant memory savings of 35× at standard evaluation lengths are unambiguously demonstrated, with performance gaps systematically reducible through gating, hybrid global layers, and contrastive feature map fine-tuning.

5. Conclusion

This paper has presented a systematic investigation of linear attention as a mechanism for achieving constant memory complexity in long-context transformer inference. Beginning from the kernel decomposition of scaled dot-product attention, a recurrent update rule was derived that processes tokens sequentially with a fixed-size state matrix, entirely eliminating the KV cache that grows linearly with generation length in standard attention. The theoretical analysis established O(1) memory complexity for the recurrent form, and direct measurement on an A100 GPU confirmed that peak inference memory remains constant at 2.1 GB across context lengths from 1,024 to 131,072 tokens, compared to 74.3 GB for standard softmax attention at the upper end of this range. Decoding throughput improves 13.6-fold at length 65,536, reflecting the constant cost of the recurrent state update versus the growing bandwidth cost of KV cache loading.

The methodology introduced four complementary components: a learnable positive feature map for improved kernel approximation quality, data-dependent gating for content-selective state management, periodic state normalization for long-sequence stability, and a residual local window for boundary handling. The conceptual grounding of the recurrent state as an associative memory — drawing on the architectural principles of both neural memory systems and end-to-end trainable memory networks — provides a unified framework for understanding the capacity and retrieval properties of fixed-size linear attention states. Ablation experiments confirmed that each component contributes materially, with the full combination reducing the softmax-to-linear perplexity gap from 22.1% to 4.8% relative on the Pile dataset. Downstream evaluation demonstrated that the performance gap is task-dependent and systematically reducible through hybrid configurations and feature map optimization.

The practical significance of these findings extends to any domain where transformer models are applied to extended sequences under memory constraints, including genomic sequence analysis, long-form audio processing, continuous sensor streams, and extended multi-turn dialogue. The constant memory footprint enables serving of arbitrarily long contexts on a single accelerator without the KV cache offloading and paging strategies required by standard attention, reducing both latency and hosting cost. Several directions merit continued investigation. The capacity of the fixed-size associative state is bounded by the feature dimension r, and developing principled methods for selecting r as a function of task complexity and available memory budget remains an open problem. Extending the feature map design through meta-learning or gradient-based optimization tailored to specific task distributions may further narrow the performance gap on precision-sensitive retrieval tasks. Formal expressivity bounds characterizing which computational tasks require growing memory and which can be solved by constant-memory recurrent attention would provide a theoretical foundation for guiding hybrid architecture design.

References

[1] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

[2] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., ... & Fedus, W. (2022). Emergent abilities of large language models. arXiv preprint arXiv:2206.07682.

[3] Zhao, X., Sun, T., Ren, S., Yang, J., & Liu, Y. (2025). RAG-Based AI Agents for Enterprise Software Development:

Implementation Patterns and Production Deployment. Frontiers in Artificial Intelligence Research, 2(3), 501-520.

[4] Li, P., Liu, J., & Qiu, L. (2026). Deep Learning Methods for Demand Forecasting and Inventory Optimization in Modern Supply Chains. Asian Business Research Journal, 11(3), 21-29.

[5] Qiu, L. (2025). Reinforcement Learning Approaches for Intelligent Control of Smart Building Energy Systems with Real-Time Adaptation to Occupant Behavior and Weather Conditions. Journal of Computing and Electronic Information Management, 18(2), 32-37.

[6] Zhang, H. (2025). Reinforcement Learning Approaches for Layout Optimization in Electronic Design Automation with Electromagnetic Compatibility Constraints. Frontiers in Robotics and Automation, 2(2), 77-93.

[7] 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.

[8] Sun, T., Wang, M., & Han, X. (2025). Deep Learning in Insurance Fraud Detection: Techniques, Datasets, and Emerging Trends. Journal of Banking and Financial Dynamics, 9(8), 1-11.

[9] Liu, J., Li, P., & Wang, Y. (2026). Graph Neural Networks for Modeling Complex Dependencies in Global Supply Chain Networks. Journal of Computing and Electronic Information Management, 20(3), 9-20.

[10] Zhang, F., & Wu, B. (2025). Large Language Models as General Purpose Intelligence Systems for Reasoning, Planning and Decision Making. American Journal of Artificial Intelligence and Neural Networks, 6(4), 45-72.

[11] Li, P., Ren, S., Zhang, Q., Wang, X., & Liu, Y. (2024). Think4SCND: Reinforcement learning with thinking model for dynamic supply chain network design. IEEE Access, 12, 195974-195985.

[12] Zhang, F., & Yang, J. S. (2025). Learning Driven Decision Intelligence for Autonomous Driving Through Multimodal Understanding World Modeling and Policy Optimization. Frontiers in Artificial Intelligence Research, 2(3), 616-634.

[13] Wang, B., Wang, Z., Zhao, W., & Liu, Y. (2025). Network Fabric Simulation and Validation for Data Center Routing Convergence Under Large-Scale Failure Scenarios. Computer Science Bulletin, 8(01), 310-326.

[14] Liu, J., Wang, J., Chen, H., Guinness, J., Martin, R., & Kulkarni, C. S. (2019). Optimal Level Crossing Predictions for Electronic Prognostics. In AIAA Scitech 2019 Forum (p. 1962).

[15] Chen, J., Cui, Y., Zhang, X., Yang, J., & Zhou, M. (2024). Temporal convolutional network for carbon tax projection: A data-driven approach. Applied Sciences, 14(20), 9213.

[16] Wei, Z., Sun, T., & Zhou, M. (2024). LIRL: Latent Imagination-Based Reinforcement Learning for Efficient Coverage Path Planning. Symmetry, 16(11), 1537.

[17] Zhang, S., Qiu, L., & Zeng, Z. (2026). Physics-Data Synergy in Structural Health Monitoring: A Multi-Scale Graph Contrastive Framework With Temperature-Adaptive Fusion. IEEE Access.

[18] Zeng, Z., Lin, H., Zhang, S., & Wang, B. (2026). Adaptive Robust Watermarking for Large Language Models via Dynamic Token Embedding Perturbation. IEEE Access, 14, 9319-9339.

[19] Qiu, L. (2025). Multi-Agent Reinforcement Learning for Coordinated Smart Grid and Building Energy Management Across Urban Communities. Computer Life, 13(3), 8-15.

[20] Zhao, W., Chen, T., Yang, J. S., & Qiu, L. (2026). AutoML-Pipeline: A RAG-enhanced code generation framework with pre-validation for cloud-native machine learning workflows. IEEE Access.

[21] Yang, Y., & Yang, J. (2026). Synthetic Data Meets Finance: Generative Models for Privacy Preserving Analytics. Journal of Banking and Financial Dynamics, 10(4), 1-8.

[22] Wang, Z., Shen, Z., Wang, B., & Shang, W. (2025). Modernizing Enterprise Analytics through Low-Code Automation and Cloud-Native Data Architectures. Asian Business Research Journal, 10(12), 20-33.

[23] Yang, S., Wang, B., Shen, Y., Panda, R., & Kim, Y. (2023). Gated linear attention transformers with hardware-efficient training. arXiv preprint arXiv:2312.06635.

[24] Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Biderman, S., ... & Zhu, R. J. (2023, December). Rwkv: Reinventing rnns for the transformer era. In Findings of the association for computational linguistics: EMNLP 2023 (pp. 14048-14077).

[25] Wei, K., Fu, Y., & Huang, H. (2020). 3-D quasi-recurrent neural network for hyperspectral image denoising. IEEE transactions on neural networks and learning systems, 32(1), 363-375.

[26] Sanford, C., Hsu, D., & Telgarsky, M. (2024). One-layer transformers fail to solve the induction heads task. arXiv preprint arXiv:2408.14332.

[27] Tay, Y., Dehghani, M., Bahri, D., & Metzler, D. (2022). Efficient transformers: A survey. ACM Computing Surveys, 55(6), 1-28.

[28] Dao, T. (2023). Flashattention-2: Faster attention with better parallelism and work partitioning. arXiv preprint arXiv:2307.08691.

[29] Chen, T., & Ding, J. (2026). Cold Start Latency Optimization Strategies for Function as a Service Platforms. Computer Life, 14(1), 64-73.