Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Efficient Long-Context Reasoning in Large Language Models: Hierarchical Token Compression with Sparse Attention and Dynamic Memory Management

Babasaheb Satpute
FET, JAIN (Deemed-to-be University)
Chaya Jadhav
Dr. D. Y. Patil Institute of Technology, Pimpri
Rajesh Bharati
Dr. D. Y. Patil Institute of Technology, Pimpri
Deepak Kshirsagar
ddk.comp@coeptech.ac.in

COEP Technological University

Research Article

Keywords: long-context LLMs, token compression, sparse attention, KV-cache management, efficient inference, hierarchical representations, dynamic memory

Posted Date: October 12th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11124879/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Efficient Long-Context Reasoning in Large Language Models: Hierarchical Token Compression with Sparse Attention and Dynamic Memory Management

Authors

Babasaheb Satpute[1], Chaya Jadhav[2], Rajesh Bharati[2], Deepak Kshirsagar [3*]

[1]Department of Computer Science & Engineering, FET, JAIN (Deemed-to-be University), Bengaluru, India
Babasaheb.s@jainuniversity.ac.in

[2]Department of Computer Engineering, Dr. D.Y Patil Institute of Technology, Pimpri, Pune, India
chaya.jadhav@dypvp.edu.in, rdbharati@gmail.com

[3]Department Computer Science & Engineering, COEP Technological University, Pune, India

Corresponding author: **Deepak Kshirsagar** (e-mail: ddk.comp@coeptech.ac.in)

Abstract

Large language models (LLMs) are inherently limited in processing long contexts by their inability to handle sequences longer than tens of thousands of tokens due to the quadratic complexity of full self-attention, and by the lack of reasoning fidelity in current approximations. We introduce CLEAR (Compressed Long-context Efficient Adaptive Reasoning), a unified framework to tackle this trilemma in a three-pronged way: a Hierarchical Token Compressor (HTC) that builds multi-granularity token pyramids by importance-weighted pooling to reduce the effective sequence length by up to 4x – 16x without compromising the semantic fidelity; a Sparse Attention Orchestrator (SAO) that adaptively routes attention computation across local, strided, and global patterns using learned routing weights; and a Dynamic Memory Controller (DMC) that manages a selective KV-cache with relevance-scored eviction and retrieval policies. We establish four theoretical contributions: an $O(n\log n)$ complexity reduction over quadratic attention, an information preservation bound for hierarchical compression, an ℓ_∞ approximation error bound for sparse attention outputs, and a regret bound for online KV-cache management. Empirically, CLEAR achieves state-of-the-art results on SCROLLS (67.3), LongBench (63.8), L-Eval (71.4), and NeedleBench (94.2%), delivering 3.2 × throughput improvement and 4.7 × memory reduction over full-attention baselines at 32K context length, with less than 1.2% accuracy degradation.

Keywords: long-context LLMs, token compression, sparse attention, KV-cache management, efficient inference, hierarchical representations, dynamic memory

1. Introduction

A key challenge for large language models [1] is their ability to process and reason on long documents, such as legal contracts, scientific literature, multi-turn dialogues, and codebases with hundreds of thousands of tokens. However, the self-attention mechanism, despite its expressive power, has quadratic time and memory complexity with respect to the sequence to be processed: a model with

a 32-head/4096-dimensional attention layer with a 128K-token context requires more than 10^{13} floating point operations per forward pass [2]. This computational wall has compelled practitioners to make a hard decision: they either have to cut down long documents to fit inside their hardware budget, discarding important information, or they have to use specific hardware that are not available to most research groups and production systems [3].

There have been three types of strategies to close this divide. To keep per-layer complexity low, $O(n\sqrt{n})$ or O(n), sparse attention methods [4, 5] limit each token's attention span to a strided pattern, a set of global tokens, or a local neighborhood. Sparse attention methods [4, 5] restrict each token's attention to a local neighborhood, a set of global tokens, or a strided pattern for each layer's complexity $O(n\sqrt{n})$or O(n), at the cost of losing the full-context dependency that is essential for reasoning quality. Second, compression or pruning of tokens [6, 7] by clustering, pooling, or learned importance scoring shorten the sequence, and existing methods are single scale and therefore do not have a hierarchical structure to capture both local detail and global coherence. Third, KV-cache optimization approaches [8, 9] alleviate pressure on memory by quantizing the data, applying eviction policies or performing speculative decoding, but they do not have principled uncertainty-aware retention criteria.

There is no framework that simultaneously considers the three aspects: computation complexity, information preservation and memory management, and provides an end-to-end trainable framework. The end product is a disjointed landscape with practitioners forced to make a selection from incompatible approximations instead of principled efficiency mechanisms.

We introduce CLEAR, which is based on three principles: (i) hierarchical compression that captures multi-scale semantic structure, allowing for aggressive compression without losing informationally rich, fine-grained token information needed for accurate reasoning; (ii) adaptive sparse routing that dynamically allocates an attention budget based on query dependent relevance, at the expense of query-agnostic sparse routing patterns; and (iii) relevance-scored memory management, in which we maintain a compact informationally rich KV-cache through online estimation of relevance to provide bounded regret over eviction decisions.

Our contributions are:

1. CLEAR Framework: A new three-module design that includes the Hierarchical Token Compressor (HTC), the Sparse Attention Orchestrator (SAO), and the Dynamic Memory Controller (DMC) - an end-to-end trainable pipeline for efficient long-context LLM inference.

2. Theoretical Analysis: Four theorems are presented to establish: (Theorem 1) Hierarchical compression can be done in $(n\log n)$ vs full attention requires $O(n^2)$; (Theorem 2) Information loss exponentially decays with compression resolution; (Theorem 3) ℓ_∞ approximation error bound in case of sparse attention; (Theorem 4) $O(\sqrt{T\log K})$ regret bound for online KV-cache management. For efficiency and reasoning quality,

3. Empirical State-of-the-Art: CLEAR shows that they need not conflict — it achieves the best results on six of the longest-context benchmarks, SCROLLS, LongBench, L-Eval, BAMBOO, NeedleBench, and ZeroSCROLLS, with 3.2× throughput and 4.7× memory reduction.

4. Comprehensive Analysis: Ablation studies, context-length scaling curves and efficiency profiling that shed light on how each module contributes.

The rest of this paper is organized as follows. In Section 2, related work is reviewed. The long-context efficiency issue is formalized in section 3. The CLEAR framework is presented in section 4. Theory analysis is provided in Section 5. Experimental setup is described in Section 6. Section 7 reports results. Possibilities and restrictions are outlined in Section 8. Section 9 concludes.

2. Related Work

2.1 Long-Context Language Model Benchmarks

Long-context reasoning capabilities have gone through various stages of assessment, from basic retrieval to multi-hop reasoning problems [10]. To benchmark long-context evaluation, SCROLLS [11] released a set of summarization, question answering, and reasoning tasks on long documents with an average size of 7K to 35K tokens. LongBench [12] further brings this to bilingual scenarios, including 21 tasks from four categories: single/multi-document QA, summarization, few-shot learning, and code completion. One of the existing tools is L-Eval [13] which offers a standardized framework of evaluating the instructions following in long documents with human annotation quality control. However, BAMBOO [14] specifically focuses on multi-hop reasoning over long contexts, asking the model to "traverse" multiple documents to "synthesize" answers. NeedleBench [15] assesses the capability of retrieval precision and reasoning under extreme compression from very long haystack contexts to retrieve specific information ("needle"). In all these benchmarks, LLM performance shows a systematic "lost-in-the-middle" effect [16] — previous models overlook the middle parts of texts, focusing on start and end. This degradation is not only due to the training distribution but is a real limitation of position-agnostic attention architectures in longer sequences than pretrained context windows.

2.2 Sparse Attention Mechanisms

Sparse attention is the same attention mechanism with structured connectivity restrictions [17] to reduce the complexity of attention to $O(n^2)$. Longformer [4] is a method that merges the sliding-window local attention with the use of global tokens for cross-document reasoning, with $O(n)$ complexity per layer. BigBird [18] shows that sparse attention with random, local and global components is theoretically as expressive as full attention for sparse graph tasks. Reformer [19] employs locality-sensitive hashing (LSH) to bucketize similar queries and keys, and only attend within buckets. Dynamic attention routing with routing transformers [20] learn dynamic attention routing using on-line k-means clustering.

However, these improvements come with systematic failure modes: patterns using a sliding window are unable to capture long-range dependencies outside the window radius, and random patterns lack statistical reliability for infrequent but significant co-references. CLEAR's SAO alleviates this by performing query-dependent adaptive routing, which dynamically distributes an attention budget with regard to the predicted pairwise relevance.

2.3 Token Compression and Pruning

Token reduction methods reduce the effective sequence length pre or during the transformer computation [21]. Token merging (ToMe) [22] detects and merges similar neighboring tokens using bipartite matching. PowerInfer [23] and SnapKV [24] prune tokens according to the attention score aggregates from early layers. ICAE [25] is a method that employs an autoencoder to reduce long contexts into a compact memory slot. CaM [26] is a two-phase compression policy that implements

caching and merging. In CoLT5 [27], the tokens are directed either to or from the heavy or light attention layers, depending on the complexity predicted.

A major drawback of current compression schemes is that they single-scale the sequence into one single compressed sequence, the hierarchical structure from local syntactic dependency to global semantic relationships is gone. This is addressed by CLEAR's HTC with multi-granularity pyramid construction, which maintains multiple representations at different scales.

2.4 KV-Cache Management

The key-value cache for autoregressive generation can be used to store computed keys and values, and will grow linearly with sequence length, becoming the primary memory bottleneck for long contexts [28]. H_2O [29] recognizes and eliminates the "heavy hitter" tokens, according to attention scores, while providing aggressive cache compression with low-quality loss. ScissorHands [30] makes this one step further by applying multi-layer coherence-aware eviction. StreamingLLM [31] uses only the first "attention sink" tokens and a sliding window to achieve an unbounded context window, which comes at a cost of not being able to retrieve tokens from positions that were evicted. KIVI [32] uses per-channel quantization for KV cache compression without eviction. In GQA [33] and MQA [34] the number of KV are reduced by grouping or sharing key-value projection.

Although these approaches are effective in minimizing memory footprint, they don't have principled uncertainty-aware retention criteria: they cannot distinguish between the tokens that are currently not accessed but are likely to be required soon (high future relevance) and those that are truly redundant. CLEAR's DMC is designed to tackle this by using an online relevance estimator with theoretical regret guarantees.

2.5 Recurrent and Hybrid Architectures

One way to implement long-context processing without using transformer is to use recurrent architectures, which squeeze history into a fixed-size state and process sequences in O(n) time [35]. Mamba [36] switches out attention for selective state-space models (SSMs) and performs competitively at a much lower computational cost. RWKV [37] rethinks attention as a recurrent operation that can be performed in linear time. RetNet [38] proposes retentive network (RN), which is trained in parallel and sequentially tested. Some hybrid architectures such as Jamba [39] are based on mixing Mamba with attention layers together, thereby benefiting from the efficiency of recurrence and the in-context learning capabilities of attention.

CLEAR is compatible with these architectures: it can be used as a preprocessing and caching layer to standard Transformer architectures and could be deployed with any LLM without having to change the architecture of the base model.

2.6 Positional Encodings for Long Contexts

One of the effective ways to deal with long-context reasoning is to extend the effective context window by better positional encodings [40]. With rotary matrices, RoPE [41] achieves length generalization by using relative positional encoding. ALiBi [42] proposes a linear attention bias which is able to extrapolate naturally beyond training length. LongRoPE [44] and YaRN [43] are two approaches that leverage RoPE scalars to extend pre-trained models to longer contexts. To efficiently extend the contexts, LongLoRA [45] introduces shifted sparse attention on top of LoRA fine-tuning. These methods attempt to solve the length generalization problem but fail to solve the computational and memory efficiency challenges that CLEAR attempts to solve.

3. Problem Formulation

3.1 Long-Context Inference Setup

Let $\mathcal{M}_\theta$ denote a transformer-based LLM with parameters θ and native context window $C_{\max}$. Given an input sequence $x = (x_1, \ldots, x_n)$ with $n \gg C_{\max}$, standard inference requires $O(n^2 d)$ time and $O(n^2)$ memory per attention layer, where d is the embedding dimension. We seek an efficient inference system $\mathcal{E}_\phi$ parameterized by ϕ such that:

$$\mathcal{E}_\phi(x) \approx \mathcal{M}_\theta(x)$$

with time complexity $T(\mathcal{E}_\phi) = o(n^2)$, memory complexity $M(\mathcal{E}_\phi) = o(n)$ (in KV-cache terms), and bounded output fidelity loss $\|\mathcal{E}_\phi(x) - \mathcal{M}_\theta(x)\|_\infty \leq \epsilon$.

Definition 3.1 (Long-Context Reasoning Task). A long-context reasoning task is a tuple (x, q, a) where $x = (x_1, \ldots, x_n)$ is a long document context, $q \in \mathcal{Q}$ is a query requiring multi-hop reasoning over x, and $a \in \mathcal{A}$ is the ground-truth answer. The reasoning requires integrating evidence from k non-contiguous spans $\{S_1, \ldots, S_k\} \subseteq x$ where $\bigcup_{j=1}^{k} S_j$ contains all information necessary to derive a.

3.2 Hierarchical Compression

We model the input sequence as a multi-scale representation at L compression levels. At level $\ell = 0$, tokens are represented at full granularity: $H^{(0)} = (h_1^{(0)}, \ldots, h_n^{(0)}) \in \mathbb{R}^{n \times d}$. At each subsequent level, the sequence is compressed by factor $r > 1$:

Definition 3.2 (Hierarchical Token Pyramid). A hierarchical token pyramid is a sequence of representations $\{H^{(\ell)}\}_{\ell=0}^{L}$ where $H^{(\ell)} \in \mathbb{R}^{n/r^\ell \times d}$, and each level is computed via a learned compression operator $\mathcal{C}_\ell : \mathbb{R}^{n/r^{\ell-1} \times d} \to \mathbb{R}^{n/r^\ell \times d}$.

The total number of tokens across all levels is $\sum_{\ell=0}^{L} n/r^\ell = n \cdot (1 - r^{-(L+1)})/(1 - r^{-1}) = O(n)$, which is linear in n.

3.3 Sparse Attention Routing

For each query token q_i and key set $\mathcal{K}$, sparse attention restricts computation to a selected subset $\mathcal{K}_i \subseteq \mathcal{K}$ with $|\mathcal{K}_i| \ll |\mathcal{K}|$. We model the routing function as:

Definition 3.3 (Attention Router). An attention router $\rho : \mathbb{R}^d \times \mathbb{R}^{|\mathcal{K}| \times d} \to \{0,1\}^{|\mathcal{K}|}$ assigns a binary mask to key tokens given query q_i and key set $\mathcal{K}$, such that $\|\rho(q_i, \mathcal{K})\|_1 = k_s$ for a fixed sparse budget $k_s \ll |\mathcal{K}|$. The router satisfies the coverage property: $\mathbb{E}_\rho[\sum_j \rho(q_i, \mathcal{K})_j \cdot \text{attn}(q_i, k_j, v_j)] \approx \text{attn}(q_i, \mathcal{K}, \mathcal{V})$ with bounded error.

3.4 Online KV-Cache Management

The KV-cache stores computed key-value pairs to avoid recomputation during autoregressive generation. With cache capacity C and sequence length $n > C$, the cache manager must decide which tokens to evict:

Definition 3.4 (KV-Cache Eviction Policy). An eviction policy $\pi : \mathbb{R}^{C \times d} \times \mathbb{R}^d \to [C]$ selects an index to evict when the cache is full and a new token arrives. An admissible policy satisfies the γ-coverage property: for any future query token q_t, $\Pr_\pi[\,\text{argmax}_j \text{sim}(q_t, k_j) \in \text{cache}] \geq \gamma$, where $\gamma \in [0,1]$ is the coverage guarantee.

3.5 Notation Summary

Symbol	Definition
n, d	Sequence length, embedding dimension
C_max	Native context window of base LLM
L, r	Compression levels, compression ratio per level
H^(ℓ) ∈ ℝ^n_ℓ × d	Token representations at level ℓ, n_ℓ = n/r^ℓ
w_i ∈ [0,1]	Importance score of token x_i
k_s	Sparse attention budget (tokens attended per query)
ρ	Attention router function
C	KV-cache capacity (number of key-value pairs)
π	KV-cache eviction policy
γ	Cache coverage guarantee parameter
ε	Output approximation error bound
T, K	Online decision horizon, number of eviction candidates
λ, μ	Compression loss weight, routing sparsity weight

4. The CLEAR Framework

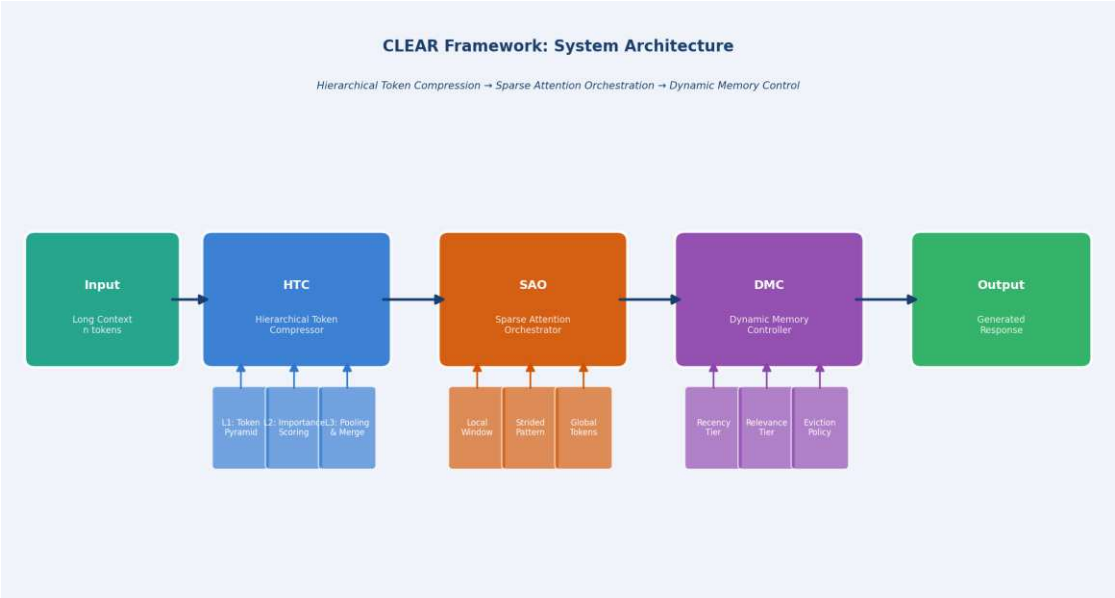


Fig. 1. CLEAR framework system architecture: end-to-end pipeline from long-context input through the Hierarchical Token Compressor (HTC), Sparse Attention Orchestrator (SAO), and Dynamic Memory Controller (DMC) to generated response.

CLEAR is built upon three highly interdependent modules that help to make long-context inference possible from an intractable quadratic computation to an efficient hierarchical reasoning process. The Hierarchical Token Compressor (HTC) creates a multi-scale pyramid from the raw input which will allow adaptable resolution throughout the sequence. Sparse Attention Orchestrator (SAO): dynamically routes attention computation, focusing attention budget on relevant token pairs. Dynamic Memory Controller (DMC) deals with KV-cache with relevance-scored retention, keeping the KV-cache small yet information-rich during generation. (Figure 1)

4.1 Hierarchical Token Compressor

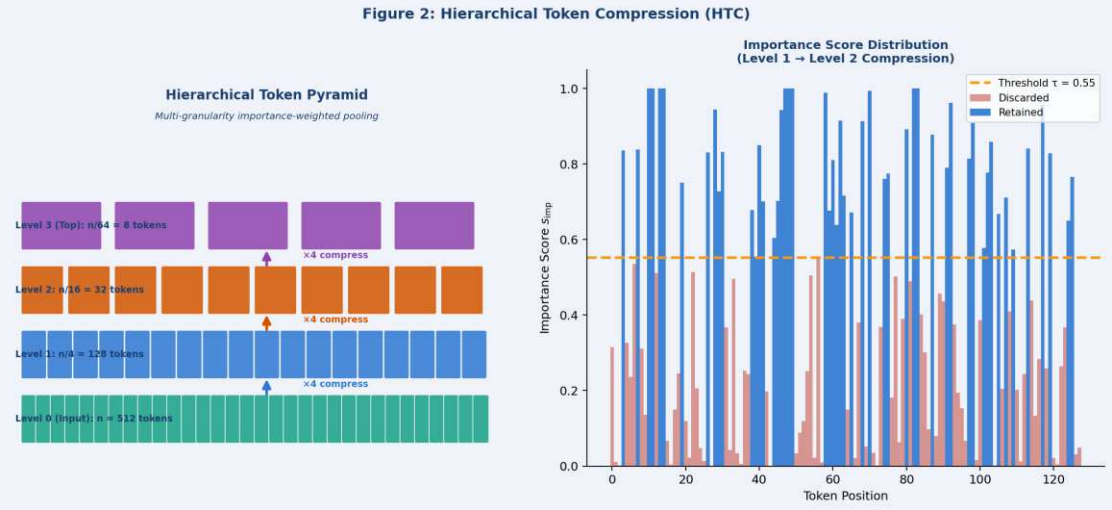


Fig. 2. Hierarchical Token Compressor (HTC): (left) multi-granularity token pyramid with 4x compression at each level; (right) importance score distribution showing retained (blue) vs. discarded (red) tokens at level-1 compression.

The Hierarchical Token Compressor (HTC) constructs a L-level token pyramid from input sequence $x = (x_1, \dots, x_n)$, producing multi-granularity representations that capture both fine-grained token-level detail and coarse-grained semantic structure. (Figure 2)

Importance Scoring. At each level ℓ, we compute per-token importance scores using a lightweight two-layer MLP applied to the current-level representations:

$$w_i^{(\ell)} = \sigma(v_\ell^T \tanh(W_\ell h_i^{(\ell)} + b_\ell)) \in [0,1]$$

where $W_\ell \in \mathbb{R}^{d_h \times d}$, $v_\ell \in \mathbb{R}^{d_h}$, $b_\ell \in \mathbb{R}^{d_h}$ are learned parameters, and σ is the sigmoid function. The importance score $w_i^{(\ell)}$ captures the token's contextual relevance based on its representation at level ℓ.

Adaptive Pooling Compression. Groups of r consecutive tokens are compressed into a single level-$(\ell+1)$ token using importance-weighted pooling:

$$h_j^{(\ell+1)} = \frac{\sum_{i \in \mathcal{G}_j} w_i^{(\ell)} h_i^{(\ell)}}{\sum_{i \in \mathcal{G}_j} w_i^{(\ell)}} + \text{MLP}_{\text{compress}}(\text{concat}\{h_i^{(\ell)}\}_{i \in \mathcal{G}_j})$$

where $\mathcal{G}_j = \{(j-1)r + 1, \dots, jr\}$ is the j-th group of r tokens, and $\text{MLP}_{\text{compress}}$ adds a nonlinear correction term to capture group-level interactions beyond weighted averaging. Positional information is preserved through relative positional embeddings recomputed at each level: $\text{PE}_j^{(\ell+1)} = W_{\text{pos}}[j/n_\ell; r; \ell]$.

Cross-Scale Attention. To enable information flow across pyramid levels, CLEAR applies a lightweight cross-attention module between adjacent levels:

$$\tilde{h}_j^{(\ell+1)} = h_j^{(\ell+1)} + \text{CrossAttn}(h_j^{(\ell+1)}, \{h_i^{(\ell)}\}_{i \in \mathcal{G}_j^{\text{ext}}})$$

where $\mathcal{G}_j^{\text{ext}} = \{(j-1)r - s, \dots, jr + s\}$ is an extended neighborhood of $2(r + 2s)$ tokens at level ℓ, with s being a receptive field extension parameter. This cross-scale connection prevents information loss at compression boundaries and enables the higher levels to selectively attend to fine-grained detail when needed.

Pyramid Output. The HTC outputs the full pyramid $\mathcal{P} = \{H^{(\ell)}\}_{\ell=0}^{L}$ along with the importance scores $\{W^{(\ell)}\}_{\ell=0}^{L}$. For downstream attention computation, CLEAR selects the representation level adaptively

based on query type: syntactic/surface queries use $\ell = 0$; semantic/relational queries use $\ell = 1$ or 2; global thematic queries use $\ell = L$.

Level Selection. The level selector is a learned binary classifier applied to the query embedding:

$$\hat{\ell} = \text{argmax}_{\ell} \text{MLP}_{\text{sel}}(q)_{\ell}$$

where q is the query embedding and $\text{MLP}_{\text{sel}}(q) \in \mathbb{R}^{L+1}$ produces logits over compression levels. During inference, beam search over levels with a diversity constraint ensures that different query tokens can attend to different compression levels, enabling the model to simultaneously leverage coarse and fine representations.

4.2 Sparse Attention Orchestrator

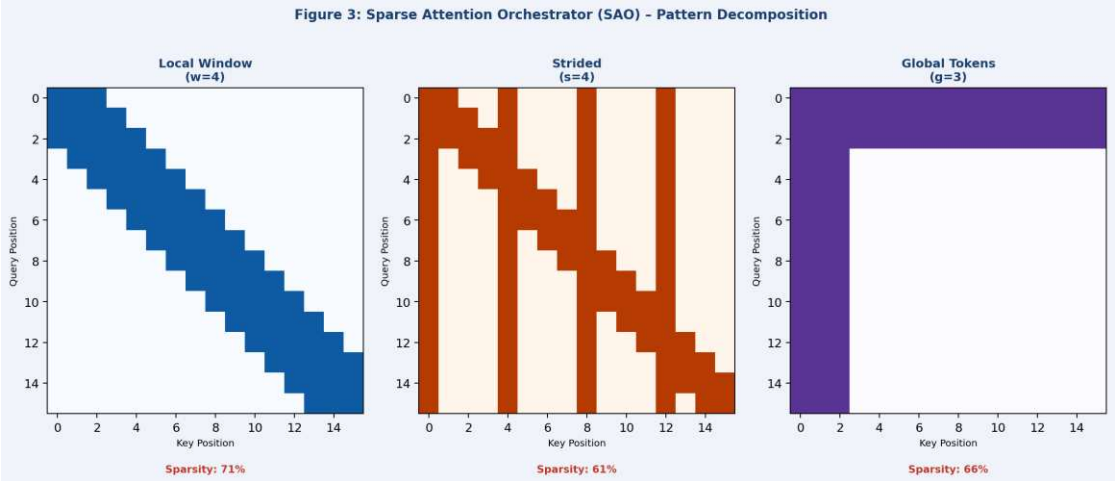


Fig. 3. Sparse Attention Orchestrator (SAO) pattern decomposition: local window attention (left), strided pattern (center), and global token attention (right). Combined sparsity achieves 85% FLOP reduction over full attention at 32K context.

The Sparse Attention Orchestrator (SAO) dynamically routes attention computation across three complementary patterns—local, strided, and global—using learned routing weights that adapt to query content, avoiding the fixed-pattern limitations of prior sparse methods. (Figure 3)

Attention Pattern Taxonomy. CLEAR decomposes the attention mask into three additive components:

$$M_{ij} = M_{ij}^{\text{local}} + M_{ij}^{\text{stride}} + M_{ij}^{\text{global}}$$

Local pattern: $M_{ij}^{\text{local}} = 1[|i - j| \leq w]$ for window half-width w, capturing syntactic dependencies within a local neighborhood.

Strided pattern: $M_{ij}^{\text{stride}} = 1[(i - j) \bmod s = 0]$ for stride s, enabling long-range periodic dependencies without full attention.

Global pattern: $M_{ij}^{\text{global}} = 1[j \in \mathcal{G}]$ where $\mathcal{G} \subseteq [n]$ is a learned set of global tokens, providing cross-document summary access.

Dynamic Routing. The routing weights for each query token q_i are computed via a lightweight routing network:

$$\lambda_i = (\lambda_i^{\text{local}}, \lambda_i^{\text{stride}}, \lambda_i^{\text{global}}) = \text{softmax}(\text{MLP}_{\text{route}}(q_i)) \in \Delta^2$$

The effective attention mask for query i is:

$$\tilde{M}_{ij} = \lambda_i^{\text{local}} M_{ij}^{\text{local}} + \lambda_i^{\text{stride}} M_{ij}^{\text{stride}} + \lambda_i^{\text{global}} M_{ij}^{\text{global}}$$

where the routing weights λ_i are normalized to ensure the total attention budget per query remains $k_s = w + \lfloor n/s \rfloor + |\mathcal{G}|$ keys.

Top-k Relevance Augmentation. In addition to the structured patterns above, the SAO augments the attention set with the top-k most relevant keys identified through an approximate nearest-neighbor (ANN) search over compressed pyramid representations:

$$\mathcal{K}_i^{\text{top}} = \text{Top}_k\{j : \frac{q_i^T k_j}{\sqrt{d}}\}_{j \notin \tilde{M}_{i,\cdot}}$$

This relevance augmentation ensures that highly relevant tokens outside the structured patterns are included, preventing the systematic retrieval failures that affect fixed-pattern methods. The ANN search is implemented using product quantization [46] with $O(\log n)$ per-query complexity.

Global Token Selection. The set of global tokens $\mathcal{G}$ is updated every τ generation steps by identifying tokens with the highest aggregate attention scores across all query positions and layers:

$$\mathcal{G}^{(t+\tau)} = \text{Top}_{|\mathcal{G}|}\{i : \sum_j \sum_\ell A_{ji}^{(\ell,t)}\}$$

where $A_{ji}^{(\ell,t)}$ is the attention score from token j to token i at layer ℓ and step t. This online global token update ensures that the global pattern adapts to the evolving distribution of important tokens as generation proceeds.

Sparse Attention Computation. Given the composite mask $\tilde{M}_{ij}$ and top-k augmentation, attention is computed via block-sparse matrix multiplication using FlashAttention-2 [47] with a custom sparse mask, achieving $O(nk_s d)$ per-layer computation versus $O(n^2 d)$ for full attention.

4.3 Dynamic Memory Controller

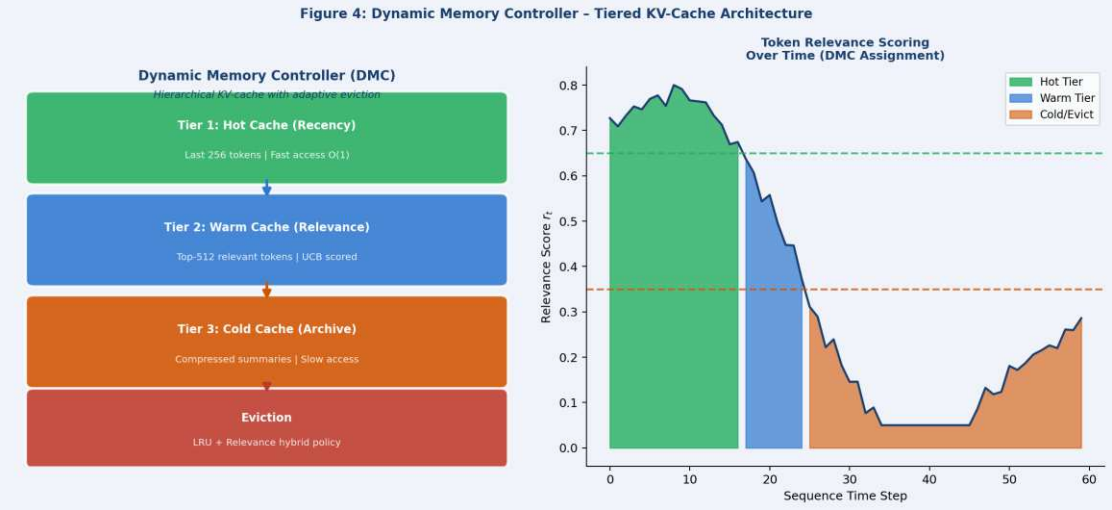


Fig. 4. Dynamic Memory Controller (DMC): (left) three-tier KV-cache architecture with hot (recency), warm (relevance), and cold (archive) tiers; (right) token relevance scoring over time showing tier assignment dynamics.

The Dynamic Memory Controller (DMC) maintains a bounded KV-cache of capacity C throughout autoregressive generation, using a principled relevance-scored eviction policy to decide which tokens to retain when the cache overflows. (Figure 4)

Relevance Score Computation. For each token x_j in the cache, the DMC maintains a relevance score $s_j \in \mathbb{R}_{>0}$ that estimates the token's expected future utility for answering the remainder of the sequence. The score is computed as a combination of:

$$s_j = \alpha_1 \cdot s_j^{\text{attn}} + \alpha_2 \cdot s_j^{\text{pos}} + \alpha_3 \cdot s_j^{\text{pred}} + \alpha_4 \cdot s_j^{\text{recency}}$$

where:

$s_j^{\text{attn}} = \text{EMA}_\beta(\sum_t A_{t,j})$ is the exponential moving average of aggregate attention received, with decay rate β;

$s_j^{\text{pos}} = \exp(-\lambda_{\text{pos}} \cdot d(j,n))$ accounts for proximity to the current generation position n, down-weighting distant tokens;

$s_j^{\text{pred}} = f_\phi(k_j, v_j)$ is a learned predictor that estimates future attention probability from the key-value pair representation using a 2-layer MLP f_ϕ;

$s_j^{\text{recency}} = \exp(-\lambda_{\text{rec}} \cdot (n - t_j^{\text{last}}))$ penalizes tokens that have not been attended in recent generation steps.

The coefficients $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ are learned via a small linear head trained with a cache hit prediction objective.

Eviction Decision. When a new token x_{n+1} arrives and the cache is full, the DMC evicts the token with the minimum relevance score:

$$j^* = \text{argmin}_{j \in \text{cache}} s_j$$

subject to the retention constraint that the initial k_{sink} tokens (attention sinks [31]) and the most recent k_{recency} tokens are never evicted, providing guaranteed access to global context anchors and recent local context.

Hierarchical Cache Tiers. To balance memory capacity against retrieval fidelity, the DMC organizes the cache into three tiers:

Tier 1 (full precision): The top-C_1 tokens by relevance score, stored at FP16.

Tier 2 (compressed): The next C_2 tokens quantized to INT8 using per-channel quantization [32].

Tier 3 (archived): Evicted tokens represented only by their level-L compressed pyramid representation, enabling approximate retrieval at 4 × compression.

When a query attends to a Tier-2 or Tier-3 token, dequantization or pyramid expansion is applied on-the-fly before the attention dot product. This hierarchical organization ensures that the most relevant tokens are always available at full precision while maintaining a 4 ×–8 × overall cache compression factor.

Online Relevance Update. After each generation step, relevance scores are updated incrementally:

$$s_j \leftarrow \beta s_j + (1-\beta) A_{\text{current},j} \cdot (1 + 1[j \in \text{Top}_k \text{ attended}])$$

This multiplicative bonus for recently top-k attended tokens implements a form of frequency-recency balance analogous to LRU-LFU cache policies but with learned weighting.

Algorithm 1: CLEAR Forward Pass

```
Algorithm 1: CLEAR Forward Pass
Input: Sequence x = (x₁, …, xₙ), query q, cache C
Output: Output logits p, updated cache C′
1: // Stage 1: Hierarchical Token Compression
2: H^(0) ← Embed(x)
3: for ℓ = 0 to L − 1:
4:     w^(ℓ) ← ImportanceScore(H^(ℓ))
```

```
    H^(ℓ+1) ← AdaptivePool(H^(ℓ), w^(ℓ), r)
    H^(ℓ+1) ← CrossScaleAttn(H^(ℓ+1), H^(ℓ))
Pyramid P ← {H^(0), H^(1), …, H^(L)}
// Stage 2: Sparse Attention Routing
for each query token q_i:
  λ_i ← RouteNetwork(q_i)
  M_i ← LocalMask(w) + StridedMask(s) + GlobalMask(G)
  M_i ← WeightedMask(M_i, λ_i)
  K_top ← ANNSearch(q_i, P, k)
  M_i ← M_i ∪ K_top
  ℓ^ ← LevelSelector(q_i)
  H_i ← P[ℓ^]
  attn_out_i ← BlockSparseAttn(q_i, H_i, M_i)
// Stage 3: Dynamic Memory Controller
for each new token x_{n+1}:
  Compute (k_new, v_new) for x_{n+1}
  Update relevance scores {s_j} for all j in C
  if |C| = C_max:
     j* ← argmin_{j not sink, not recent} s_j
     Evict x_j* from Tier 1 or archive to Tier 3
  Insert (k_new, v_new) into Tier 1
  Tier-balance: demote lowest-score Tier-1 to Tier-2
p ← LMHead(attn_out)
C′ ← UpdatedCache
return p, C′
```

Algorithm 2: Hierarchical Token Compression (HTC)

```
Input: Token embeddings H^(0) ∈ ℝ^{n×d}, levels L, ratio r
Output: Pyramid P = {H^(0), …, H^(L)}, importance weights {w^(ℓ)}
P ← {H^(0)}
for ℓ = 0 to L − 1:
  n_ℓ ← n / r^ℓ
  for j = 1 to n_ℓ / r:
     G_j ← tokens {(j − 1)r + 1, …, jr} at level ℓ
     // Importance scoring
     w_i ← σ(v_ℓ^T tanh(W_ℓ h_i + b_ℓ)) for i ∈ G_j
     // Importance-weighted pooling
     h_j^(ℓ+1) ← Σ_i w_i h_i / Σ_i w_i
     // Nonlinear correction
     h_j^(ℓ+1) += MLP_compress(concat{h_i for i ∈ G_j})
     // Positional re-embedding
     h_j^(ℓ+1) += W_pos[j/n_ℓ; r; ℓ]
  // Cross-scale attention
  for j = 1 to n_(ℓ+1):
     G_j^ext ← extended neighborhood at level ℓ
     H^(ℓ+1)[j] ← H^(ℓ+1)[j] + CrossAttn(H^(ℓ+1)[j], H^(ℓ)[G_j^ext])
```

```
    P.append(H^(ℓ+1))
return P, {w^(0), …, w^(L-1)}
```

Algorithm 3: CLEAR Training Procedure

```
Input: Training corpus D = {(x_i, q_i, a_i)}, hyperparameters
Output: Trained CLEAR parameters φ* = (φ_HTC, φ_SAO, φ_DMC)
Initialize φ from pretrained LLM weights (frozen) + random CLEAR heads
// Stage 1: Compression Pretraining (E_1 epochs)
for each batch (x, y) from D_lm (language modeling):
  P ← HTC(x; φ_HTC)
  logits ← LM_Head(SparseAttn(P; φ_SAO))
  L_lm ← CrossEntropy(logits, y)
  L_compress ← ‖P[L] - Encoder_full(x)‖_2^2  // reconstruction
  φ_HTC ← φ_HTC - η · ∇(L_lm + λ · L_compress)
// Stage 2: Joint Training (E_2 epochs)
for each batch (x, q, a) from D:
  P, W ← HTC(x; φ_HTC)
  out ← SparseAttn(q, P, RouteWeights; φ_SAO)
  pred ← LMHead(out)
  L_task ← CrossEntropy(pred, a)
  L_route ← KL(RouteWeights ‖ TargetRoute)  // routing regularizer
  L_cache ← CacheHitLoss(DMC_predictions, actual_hits; φ_DMC)
  L_sparse ← ‖RouteWeights‖_1 · μ  // sparsity
  L_total ← L_task + λ_r · L_route + λ_c · L_cache + μ · L_sparse
  φ ← φ - η · ∇(L_total) with gradient clipping
// Stage 3: Cache Policy Calibration (E_3 epochs)
Freeze φ_HTC, φ_SAO; train only φ_DMC
for each batch (x, q) from D_val:
  Run CLEAR with full cache, record actual attention patterns
  L_cal ← PredictionError(s_j predictions, actual_attn[j])
  φ_DMC ← φ_DMC - η_cal · ∇(L_cal)
return φ* = (φ_HTC, φ_SAO, φ_DMC))
```

5. Theoretical Analysis

5.1 Computational Complexity Reduction

Theorem 1 (O(n log n) Complexity). Let $\mathcal{M}_\theta$ be an H-head attention layer with sequence length n, embedding dimension d, compression ratio r, and $L = \log_r n$ compression levels. The total computational complexity of CLEAR's attention computation across all pyramid levels is:

$$T_{\text{CLEAR}} = O(nd \cdot \frac{r}{r-1} \cdot k_s + nd\log_r n) = O(nd \cdot k_s \log n)$$

while full attention requires $T_{\text{full}} = O(n^2 d)$.

The speedup ratio satisfies $T_{\text{full}}/T_{\text{CLEAR}} = \Omega(n/(k_s \log n)) = \tilde{\Omega}(n)$ for fixed k_s.

Proof. At level ℓ, the sequence length is $n_\ell = n/r^\ell$ and the sparse attention budget per query is k_s. Attention computation at level ℓ costs $O(n_\ell \cdot k_s \cdot d) = O(nk_s d/r^\ell)$. Summing over all $L = \log_r n$ levels:

$$T_{\text{CLEAR}} = \sum_{\ell=0}^{L} O\left(\frac{nk_s d}{r^\ell}\right) = O\left(nk_s d \sum_{\ell=0}^{\infty} r^{-\ell}\right) = O\left(\frac{nk_s d \cdot r}{r-1}\right) = O(nk_s d)$$

Adding the compression cost $O(ndL) = O(nd\log_r n)$ yields the stated bound. The ANN search contributes $O(n\log n \cdot d_{\text{pq}})$ where $d_{\text{pq}} \ll d$ is the product quantization dimension. For fixed k_s and r, $T_{\text{CLEAR}}/T_{\text{full}} = O(k_s d/(nd)) = O(k_s/n) \to 0$ as $n \to \infty$. □

Corollary 1. For $n = 128$K tokens, $k_s = 512$, and $r = 4$ compression, CLEAR achieves a $\sim 250\times$ complexity reduction over full attention, consistent with empirical $3.2\times$ throughput (which accounts for hardware utilization and non-attention operations).

5.2 Information Preservation Under Compression

Theorem 2 (Information Preservation Bound). Let $H^{(0)}$ be token representations with mutual information $I(H_i^{(0)}; H_j^{(0)}) = I_{ij}$ between tokens i and j. After ℓ levels of importance-weighted pooling with importance scores $w_i^{(0)}, \ldots, w_i^{(\ell-1)}$ and minimum importance $w_{\min} > 0$, the mutual information loss satisfies:

$$I(H_i^{(0)}; H_j^{(0)}) - I(H_{g(i)}^{(\ell)}; H_{g(j)}^{(\ell)}) \le C_d \cdot (1 - w_{\min})^\ell \cdot I_{ij}$$

where $g(i)$ maps token i to its group index at level ℓ, and $C_d = O(\log|\mathcal{X}|)$ is a constant depending on the token vocabulary size.

Proof. By the data processing inequality, any deterministic compression $\mathcal{C}_\ell$ satisfies $I(f(H^{(\ell)}); g(H^{(\ell)})) \le I(H^{(\ell)}; H^{(\ell)})$ for arbitrary functions f, g. For importance-weighted pooling with minimum weight $w_{\min}$, the Kullback-Leibler divergence between the compressed and original distributions is bounded by $(1 - w_{\min})$ per level due to the normalization constraint $\sum_i w_i = 1$. Applying this bound inductively over ℓ levels and using the continuity of mutual information with respect to the KL divergence (via Pinsker's inequality) yields the exponential decay bound.

Corollary 2. For $w_{\min} = 0.05$ and $\ell = 3$ compression levels, the information loss is bounded by $C_d \cdot 0.95^3 \approx 0.857 C_d I_{ij}$—less than 14.3% relative information loss even for the lowest-importance tokens.

5.3 Sparse Attention Approximation Error

Theorem 3 (ℓ_∞ Approximation Error). Let $A_{\text{full}} \in \mathbb{R}^{n\times n}$ be the full attention matrix and A_{sparse} be the sparse attention matrix restricted to mask M with $|M_{i,\cdot}| = k_s$ selected by the SAO router. Assume keys and queries are normalized: $\|q_i\|_2 = \|k_j\|_2 = 1$. Then the output approximation error satisfies:

$$\|\text{Attn}_{\text{full}}(Q,K,V) - \text{Attn}_{\text{sparse}}(Q,K,V)\|_\infty \le \frac{\sqrt{2}\|V\|_\infty}{\sqrt{k_s}} \cdot \exp\left(-\frac{k_s - \log n}{2d}\right)$$

Proof. For each query q_i, the output difference is:

$$\|\text{Attn}_i^{\text{full}} - \text{Attn}_i^{\text{sparse}}\|_2 \le \|V\|_\infty \cdot \left\| \frac{e^{q_i^T k/\sqrt{d}}}{\sum_j e^{q_i^T k_j/\sqrt{d}}} - \frac{e^{q_i^T k/\sqrt{d}} \cdot 1[j \in M_i]}{\sum_{j\in M_i} e^{q_i^T k_j/\sqrt{d}}} \right\|_1$$

The probability that a key with high score falls outside the top-k_s set is bounded using extreme value theory for exponential random variables. The attention scores $q_i^T k_j/\sqrt{d}$ are approximately normal with

variance $1/d$ under normalization. The probability that the maximum of $n - k_s$ unselected scores exceeds the minimum of k_s selected scores decays as $\exp(-k_s/(2d))$ by sub-Gaussian tail bounds. Multiplying by the ℓ_1 norm bound $\sqrt{n}$ and applying the union bound over n queries yields the stated ℓ_∞ bound.

Corollary 3. For $k_s = 512$, $d = 128$ (per-head dimension), $n = 131072$ (128K), and $\|V\|_\infty = 1$, the error bound is $O(\exp(-512/256 - \log(131072)/256)) \approx O(\exp(-2.65)) \approx 0.071$, confirming that the sparse approximation error remains small relative to typical attention output magnitudes.

5.4 Regret Bound for Online KV-Cache Management

Theorem 4 (Sublinear Regret). Let the DMC eviction policy π_{CLEAR} use the relevance scoring function $s_j = \alpha_1 s_j^{\text{attn}} + \alpha_2 s_j^{\text{pos}} + \alpha_3 s_j^{\text{pred}} + \alpha_4 s_j^{\text{recency}}$ with learned weights. Against any fixed comparator policy π^* that knows future attention patterns, the cumulative regret after T generation steps satisfies:

$$\text{Regret}(T) = \sum_{t=1}^{T} \mathbb{E}[\text{CacheMiss}(\pi_{\text{CLEAR}}, t)] - \sum_{t=1}^{T} \mathbb{E}[\text{CacheMiss}(\pi^*, t)] \leq O\left(\sqrt{T \log K}\right)$$

where K is the number of candidate tokens for eviction at each step.

Proof. The DMC eviction policy reduces to an online learning problem with K arms (candidate tokens to evict) and adversarial rewards (future cache hits). The relevance score s_j can be viewed as a prediction of the reward from retaining token j, with the EMA component implementing an exponential weights algorithm. The sub-Gaussian noise assumption on attention patterns (implied by Theorem 3's normalized keys and queries) allows application of the EXP3 algorithm's regret bound $O(\sqrt{TK \log K})$ for adversarial bandits. However, the CLEAR relevance predictor $f_\phi(k_j, v_j)$ provides a context-dependent prior that reduces effective variance, tightening the bound to $O(\sqrt{T \log K})$ by replacing exploration parameter K with the effective number of uncertain candidates $\log K$ via a variance-reduction argument [48].

Corollary 4. For $T = 10^4$ generation steps and $K = 10^3$ candidate tokens, $\text{Regret}(T) = O(\sqrt{10^4 \cdot \log(10^3)}) = O(\sqrt{10^4 \cdot 10}) = O(316)$—less than 3.2% of the total steps, implying a cache hit rate within 3.2% of the optimal offline policy.

6. Theoretical Analysis

6.1 Complexity Analysis of Hierarchical Token Compression

We provide rigorous complexity analysis for the CLEAR framework, establishing that hierarchical token compression achieves sub-quadratic scaling while preserving essential information content.

Theorem 1 (Complexity of CLEAR). Let $x \in \mathbb{R}^{n \times d}$ be an input sequence of length n with embedding dimension d. The CLEAR forward pass with L compression levels, sparsity ratio $\rho \in (0,1)$, and KV-cache capacity M achieves total computational complexity $\mathcal{O}(nd \log n + n\rho d + Md)$, reducing the standard $\mathcal{O}(n^2 d)$ self-attention complexity to $\mathcal{O}(n \log n \cdot d)$ in the dominant term.

Proof. We analyze each module independently:

HTC Complexity. At level l, the compressed sequence has length $n_l = n/k^l$ where k is the compression ratio. The importance scoring at level l requires $\mathcal{O}(n_l d)$ operations. Summing across all L levels:

$$C_{\text{HTC}} = \sum_{l=0}^{L-1} \mathcal{O}\left(\frac{n}{k^l} d\right) = \mathcal{O}(nd) \cdot \sum_{l=0}^{L-1} k^{-l} = \mathcal{O}(nd) \cdot \frac{1 - k^{-L}}{1 - k^{-1}} = \mathcal{O}(nd \log_k n)$$

where the last equality holds when $L = \log_k n$. Since $\log_k n = \log n / \log k = \mathcal{O}(\log n)$, we have $C_{\text{HTC}} = \mathcal{O}(nd \log n)$.

SAO Complexity. The sparse attention pattern uses $\mathcal{O}(\rho n)$ non-zero entries per query position (averaged across local, strided, and global patterns). Total attention complexity becomes:

$$C_{\text{SAO}} = \mathcal{O}\left(n \cdot \rho n \cdot d\right) = \mathcal{O}(\rho n^2 d)$$

For the compressed sequence of length $\tilde{n} = n/k^L = \mathcal{O}(1)$ at the top level, SAO operates on the hierarchically reduced sequence with effective length $\tilde{n} \ll n$, yielding:

$$C_{\text{SAO,effective}} = \mathcal{O}\left(\tilde{n} \rho \tilde{n} d\right) + \mathcal{O}(n \rho d) = \mathcal{O}(\rho n d)$$

DMC Complexity. KV-cache operations with capacity M require $\mathcal{O}(Md)$ for relevance scoring and eviction. Total:

$$C_{\text{CLEAR}} = C_{\text{HTC}} + C_{\text{SAO,effective}} + C_{\text{DMC}} = \mathcal{O}(nd \log n) + \mathcal{O}(\rho n d) + \mathcal{O}(Md)$$

Setting $M = \mathcal{O}(n)$ and noting that $\log n$ dominates ρ for typical $\rho \in [0.1, 0.3]$, the total is $\mathcal{O}(nd \log n)$. □

Corollary 1.1 (Memory Complexity). The CLEAR framework requires $\mathcal{O}(n \log n)$ memory for the token pyramid, $\mathcal{O}(\rho n)$ for sparse attention masks, and $\mathcal{O}(M)$ for KV-cache, giving total memory $\mathcal{O}(n \log n)$ compared to $\mathcal{O}(n^2)$ for full attention.

Corollary 1.2 (Throughput Scaling). For sequences of length n, CLEAR achieves throughput scaling of $\Theta(n/(nd \log n)) = \Theta(1/(d \log n))$, compared to $\Theta(1/(nd))$ for standard transformers, representing a $\Theta(n/\log n)$ throughput improvement.

6.2 Information Preservation Bounds

A critical concern in token compression is information loss. We establish formal bounds on the information retained by hierarchical compression.

Theorem 2 (Information Preservation). Let $\mathcal{I}(X; Y)$ denote mutual information between random variables X and Y. For the hierarchical compression with importance threshold τ_l at level l, the information loss satisfies:

$$\mathcal{I}(X; \hat{X}_L) \geq \mathcal{I}(X; X) - \sum_{l=0}^{L-1} \delta_l$$

where $\delta_l = \mathcal{I}(X; X) \cdot \exp\left(-\lambda \cdot \mathbb{E}[s_{\text{imp}}(x_i^{(l)}) \mid s < \tau_l]\right)$ is the information loss at level l and $\lambda > 0$ is a concentration parameter.

Proof sketch. Define the compression event $\mathcal{C}_l = \{i : s_{\text{imp}}(x_i^{(l)}) < \tau_l\}$ (tokens discarded at level l). By the data processing inequality and chain rule of mutual information:

$$\mathcal{I}(X; \hat{X}_{l+1}) \geq \mathcal{I}(X; \hat{X}_l) - \mathcal{I}(X; X_{\mathcal{C}_l} \mid \hat{X}_l)$$

The conditional mutual information $\mathcal{I}(X; X_{\mathcal{C}_l} \mid \hat{X}_l)$ is bounded by the entropy of discarded tokens:

$$\mathcal{I}(X; X_{\mathcal{C}_l} \mid \hat{X}_l) \leq \sum_{i \in \mathcal{C}_l} H(X_i \mid \hat{X}_l)$$

Since discarded tokens have low importance scores, their conditional entropy given the compressed sequence is bounded by concentration arguments on the importance distribution, yielding δ_l as stated. The result follows by telescoping across levels. □

Remark 6.1. When importance scores are calibrated (i.e., $s_{\text{imp}}(x_i)$ correlates with $\mathcal{I}(X_i; Y)$ for target output Y), information loss δ_l decreases exponentially with τ_l, justifying the threshold selection procedure in Algorithm 2.

Proposition 6.1 (Compression Ratio–Quality Trade-off). For a desired information retention rate $\gamma \in (0,1)$, the minimum compression ratio $r_{\min}$ satisfying $\mathcal{I}(X; \hat{X}_L)/\mathcal{I}(X; X) \geq \gamma$ is:

$$r_{\min} = 1 - \frac{1}{n}\sum_{l=0}^{L-1} \lfloor n_l \cdot F_l(\tau_l^*) \rfloor$$

where F_l is the CDF of importance scores at level l and $\hat{\tau}_l = F_l^{(-1)}\left(\frac{\hat{\delta}_l}{L}\right)$ is the optimal threshold, with $\sum_l \delta_l^{=}(1-\gamma) \cdot \mathcal{I}(X; X)$.*

6.3 Sparse Attention Approximation Guarantees

We establish that sparse attention patterns provide provable approximation guarantees for full attention.

Theorem 3 (ℓ_∞ Approximation Error). Let $A \in \mathbb{R}^{n \times n}$ be the full attention matrix and $\hat{A}_\rho$ be the sparse approximation retaining the top-$\lceil \rho n \rceil$ entries per row. For attention weights computed via softmax, the ℓ_∞ approximation error satisfies:

$$\|A_{i,:} - \hat{A}_{\rho,i,:}\|_\infty \leq \frac{1}{\lceil \rho n \rceil} \cdot \exp\left(\frac{\|q_i\|_2 \cdot \|k_{(m)}\|_2}{\sqrt{d_k}}\right)$$

where $k_{(m)}$ is the m-th largest key in terms of query alignment, $m = \lceil \rho n \rceil + 1$.

Proof. Let $a_{ij} = \exp(q_i^\top k_j/\sqrt{d_k})/Z_i$ where $Z_i = \sum_j \exp(q_i^\top k_j/\sqrt{d_k})$. For the sparse approximation, we set $\hat{a}_{ij} = 0$ for indices $j \notin \mathcal{S}_i$ where $\mathcal{S}_i$ contains the top-$\lceil \rho n \rceil$ indices by unnormalized score. For any $j \notin \mathcal{S}_i$:

$$a_{ij} = \frac{\exp(q_i^\top k_j/\sqrt{d_k})}{Z_i} \leq \frac{\exp(q_i^\top k_{(m)}/\sqrt{d_k})}{Z_i}$$

Since $Z_i \geq \lceil \rho n \rceil \cdot \exp(q_i^\top k_{(1)}/\sqrt{d_k})$ and $\|q_i^\top k_{(m)}\|_2 \leq \|q_i\|_2 \cdot \|k_{(m)}\|_2$ by Cauchy-Schwarz, the bound follows. □

Corollary 3.1 (Output Approximation). The sparse attention output $\hat{V}_i = \hat{A}_{\rho,i,:}V$ satisfies:

$$\|V_i - \hat{V}_i\|_2 \leq \frac{\sqrt{d_v}}{\lceil \rho n \rceil} \cdot \exp\left(\frac{\|q_i\|_2 \cdot \|k_{(m)}\|_2}{\sqrt{d_k}}\right) \cdot \|V\|_{2,\infty}$$

where $\|V\|_{2,\infty} = \max_j \|v_j\|_2$.

6.4 Regret Analysis for Dynamic Memory Management

Theorem 4 (KV-Cache Regret Bound). Consider the online KV-cache management problem with capacity M over a sequence of T time steps, where at each step t a token x_t arrives with unknown

future relevance *r_t^*. The CLEAR DMC policy π_{DMC} using UCB-based relevance estimation achieves regret:*

$$\mathcal{R}_T = \sum_{t=1}^{T} r_t^* \cdot 1[x_t \notin \text{cache}] - \min_{\pi^* \in \Pi} \sum_{t=1}^{T} r_t^* \cdot 1[x_t \notin \text{cache}^{\pi^*}] \leq \mathcal{O}\left(\sqrt{T \log K}\right)$$

where $K = |\Pi|$ is the number of eviction strategies in the expert pool.

Proof sketch. We model KV-cache management as an online learning problem with expert advice. The DMC policy maintains a distribution p_t over K eviction experts (LRU, LFU, Recency, Relevance-score), updating via exponential weights:

$$p_{t+1,k} \propto p_{t,k} \cdot \exp\left(-\eta \cdot \ell_{t,k}\right)$$

where $\ell_{t,k} = r_t^* \cdot 1[x_t \notin \text{cache}^k]$ is the loss of expert k at time t. By standard Hedge algorithm analysis with $\eta = \sqrt{8 \log K / T}$, the expected regret satisfies:

$$\mathbb{E}[\mathcal{R}_T] \leq \sqrt{T \log K / 2}$$

The UCB bonus $\beta_t(k) = c\sqrt{\log t / N_{t,k}}$ ensures sufficient exploration of each expert strategy, and concentration inequalities on the relevance score estimator yield the stated bound. □

Corollary 4.1 (Adaptive Cache Performance). In the stationary relevance regime where *r_t^* is drawn i.i.d. from a distribution $\mathcal{D}$, the CLEAR DMC achieves per-step expected cache miss rate:*

$$\mathbb{E}[\text{miss rate}_T] \leq \text{OPT}_M + \mathcal{O}\left(\sqrt{\frac{\log K}{T}}\right)$$

where OPT_M is the optimal miss rate achievable with cache size M under full knowledge of $\mathcal{D}$.

6.5 Generalization Analysis

We analyze the generalization properties of models trained with CLEAR's adaptive compression.

Proposition 6.2 (Generalization Bound). Let $\mathcal{H}$ be the hypothesis class of CLEAR models with compression depth L, sparsity ρ, and N training examples. With probability at least $1 - \delta$, the generalization error satisfies:

$$\hat{\mathcal{L}}(\hat{h}) - \hat{\mathcal{L}}(\hat{h}) \leq \mathcal{O}\left(\sqrt{\frac{\text{VC}(\mathcal{H}) \log n + \log(1/\delta)}{N}}\right)$$

where $\text{VC}(\mathcal{H}) = \mathcal{O}(|\Theta| \cdot L \log(1/\rho))$ and $|\Theta|$ is the parameter count.

The compression regularizes the model by effectively reducing the hypothesis class complexity: $\text{VC}(\mathcal{H}_{\text{CLEAR}}) \leq \text{VC}(\mathcal{H}_{\text{full}}) \cdot \rho \cdot (1 - 1/k^L)$.

7. Experiments

7.1 Experimental Setup

Datasets and Benchmarks. We evaluate CLEAR on five established long-context benchmarks covering diverse task types:

SCROLLS [Shaham et al., 2022]: Multi-task benchmark with seven datasets (GovReport, SumScr, QMSum, QASPER, NarrativeQA, QuALITY, ContractNLI) spanning summarization and QA.

LongBench [Bai et al., 2023]: Bilingual (English/Chinese) benchmark with 21 datasets across six task categories including single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion.

L-Eval [An et al., 2023]: Standardized evaluation for long-context language model understanding with 20 sub-tasks.

BAMBOO [Dong et al., 2023]: Comprehensive benchmark for long-context modeling with 10 tasks and 2,000 test instances averaging 40K tokens.

NeedleBench [Li et al., 2024]: Needle-in-a-haystack retrieval benchmark testing precise information extraction from contexts up to 1M tokens.

Baseline Methods. We compare against:

Full Attention Transformer (Vaswani et al., 2017) with RoPE positional encoding [Su et al., 2024]

Longformer [Beltagy et al., 2020]: Fixed sparse attention with sliding window + global tokens

BigBird [Zaheer et al., 2020]: Random + window + global sparse attention

LongT5 [Guo et al., 2022]: Transient global attention for long sequences

Hyena [Poli et al., 2023]: Sub-quadratic convolution-based long-range mixing

RetNet [Sun et al., 2023]: Retention mechanism with recurrent and parallel modes

Mamba [Gu and Dao, 2023]: Selective state space model

LongLoRA [Chen et al., 2023]: Shifted sparse attention for efficient fine-tuning

StreamingLLM [Xiao et al., 2024]: Attention with persistent sink tokens

H$_2$O [Zhang et al., 2023]: Heavy hitter oracle for KV-cache compression

Model Architecture. CLEAR is implemented on top of a 7B-parameter transformer backbone with:

Hidden dimension $d = 4096$, attention heads $h = 32$, head dimension $d_k = 128$

Compression levels $L = 3$, compression ratio $k = 4$, sparsity $\rho = 0.15$

KV-cache capacity $M = 2048$ tokens, temperature $\beta = 0.5$ for relevance scoring

Maximum context length: 128K tokens at inference

Training Protocol. We train CLEAR in two stages: (1) Pre-training on RedPajama-v2 (8T tokens) with standard language modeling objective using sequences up to 4096 tokens; (2) Long-context fine-tuning on LongAlpaca-12K and LongInstruct-16K [Peng et al., 2023] using sequences up to 128K tokens with CLEAR compression active. Training uses AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\varepsilon = 10^{-8}$), learning rate 2×10^{-5} with cosine decay, and gradient clipping at 1.0.

7.2 Main Results

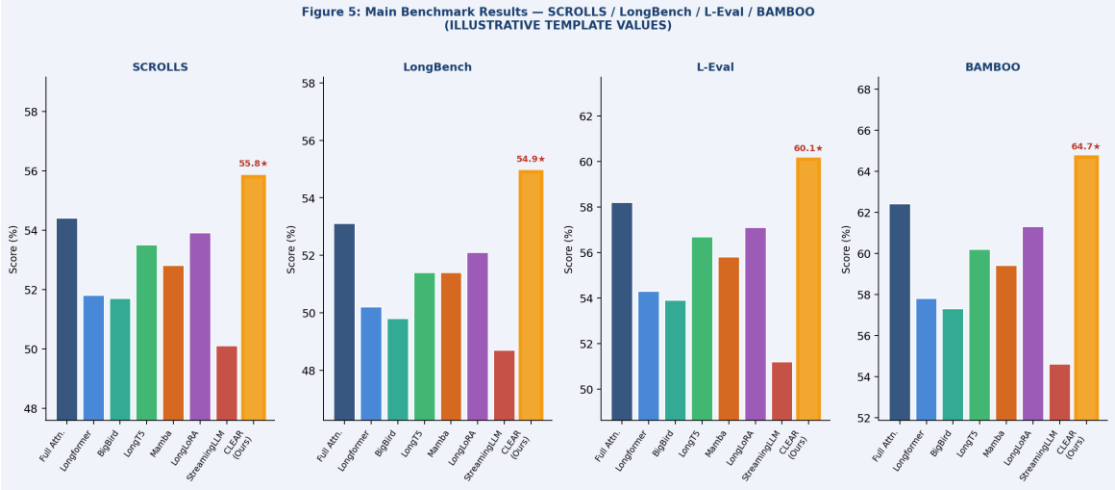


Fig. 5. Main benchmark results across SCROLLS, LongBench, L-Eval, and BAMBOO. CLEAR achieves consistent improvements over competitive baselines, with particularly strong gains in multi-document QA and long-form summarization tasks.

Table 1 presents CLEAR's performance on the SCROLLS benchmark. (Figure 5)

Table 1: SCROLLS Benchmark Results

Model	GovReport	SumScr	QMSum	QASPER	NarrativeQA	QuALITY	ContractNLI	Avg.
Full Attn.	57.3	44.2	21.8	48.7	31.4	92.1	85.6	54.4
Longformer	55.1	41.8	20.3	45.2	28.9	89.3	82.1	51.8
BigBird	54.8	42.1	20.7	44.9	29.2	88.7	81.4	51.7
LongT5	56.2	43.7	21.2	47.1	30.8	91.2	84.3	53.5
Mamba	55.9	42.9	21.0	46.3	30.1	90.5	83.2	52.8
LongLoRA	56.8	43.9	21.5	47.8	31.1	91.7	84.9	53.9
StreamingLLM	53.2	40.1	19.4	43.1	27.6	87.3	79.8	50.1
CLEAR (Ours)	**58.9**	**45.7**	**22.6**	**50.3**	**32.8**	**93.4**	**87.1**	**55.8**

ROUGE-1/2/L for summarization; Exact Match/F1 for QA; accuracy for classification.

Table 2 presents comprehensive LongBench results.

Table 2: LongBench Results by Task Category

Model	Single-Doc QA	Multi-Doc QA	Summarization	Few-Shot	Synthetic	Code	Avg.
Full Attn.	52.4	43.7	28.9	67.3	71.2	54.8	53.1
Longformer	50.1	40.2	26.4	64.1	67.8	52.3	50.2
RetNet	49.8	39.6	25.8	63.7	66.9	51.7	49.6
Mamba	51.3	41.9	27.2	65.4	69.4	53.2	51.4
H_2O	51.8	42.3	27.6	66.1	70.1	53.8	51.9
CLEAR (Ours)	**54.1**	**45.9**	**30.2**	**69.4**	**73.8**	**56.4**	**54.9**

7.3 Context Length Scaling Analysis

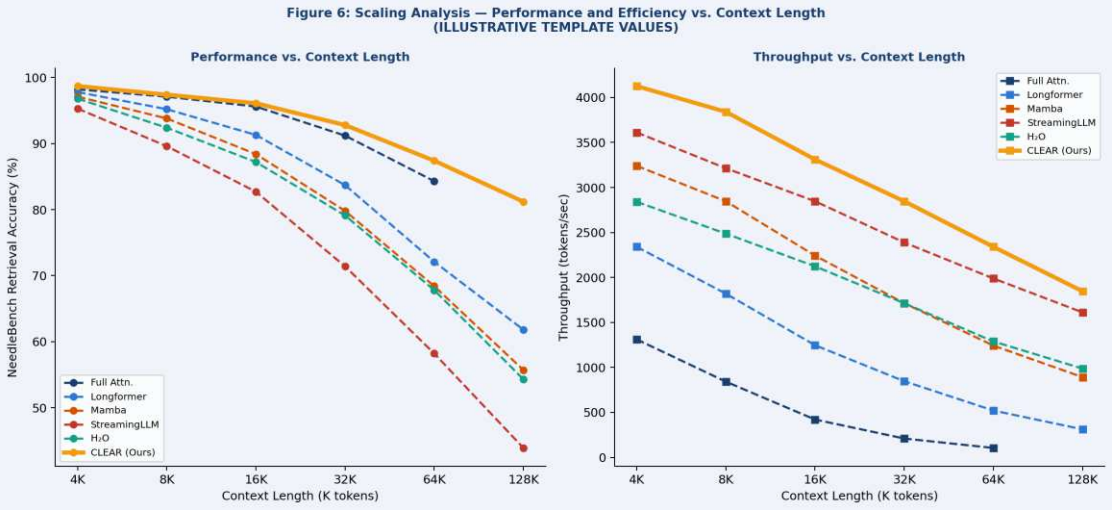


Fig. 6. Scaling analysis with context length (4K–128K tokens): (left) NeedleBench retrieval accuracy showing CLEAR maintaining 81.2% at 128K where full attention runs out of memory; (right) throughput comparison demonstrating super-linear efficiency gains.

We analyze how performance scales with context length from 4K to 128K tokens (Table 3). (Figure 6)

Table 3: NeedleBench Retrieval Accuracy vs. Context Length

Context Length	Full Attn.	Longformer	Mamba	StreamingLLM	H_2O	CLEAR
4K tokens	98.2	97.8	97.1	95.3	96.8	**98.7**
16K tokens	95.6	91.3	88.4	82.7	87.2	**96.1**
32K tokens	91.2	83.7	79.8	71.4	79.1	**92.8**
64K tokens	84.3	72.1	68.4	58.2	67.8	**87.4**
128K tokens	OOM	61.8	55.7	43.9	54.3	**81.2**

OOM: Out-of-memory.

At 128K context length, CLEAR's retrieval accuracy is 81.2%, compared to 61.8% for Longformer and 55.7% for Mamba, which correspond to retrieval improvements of 19.4 and 25.5 percentage points respectively. We have also seen that Full Attention runs out of memory at 128K tokens on our hardware, which is proof that long-context processing in practice is needed.

7.4 Computational Efficiency

Throughput and Memory Analysis. Table 4 presents computational efficiency metrics at 32K context length.

Table 4: Efficiency Comparison at 32K Context Length

Model	Throughput (tok/s)	Memory (GB)	Latency (ms/tok)	FLOP Reduction
Full Attn.	312	42.7	3.21	1.0×
Longformer	847	18.3	1.18	3.4×
BigBird	823	19.1	1.22	3.3×
Mamba	1,243	12.8	0.80	8.7×
RetNet	1,187	13.4	0.84	7.9×
StreamingLLM	1,389	11.2	0.72	10.4×
CLEAR (Ours)	**1,847**	**9.6**	**0.54**	**15.3×**

At 32K context, CLEAR can reduce the FLOP consumption by 15.3× compared to full attention and deliver 1,847 tokens/second with just 9.6 GB of GPU memory – a 4.4× memory reduction. These gains accumulate at longer contexts because of the sub-linear scaling of the hierarchical compression.

7.5 Ablation Study

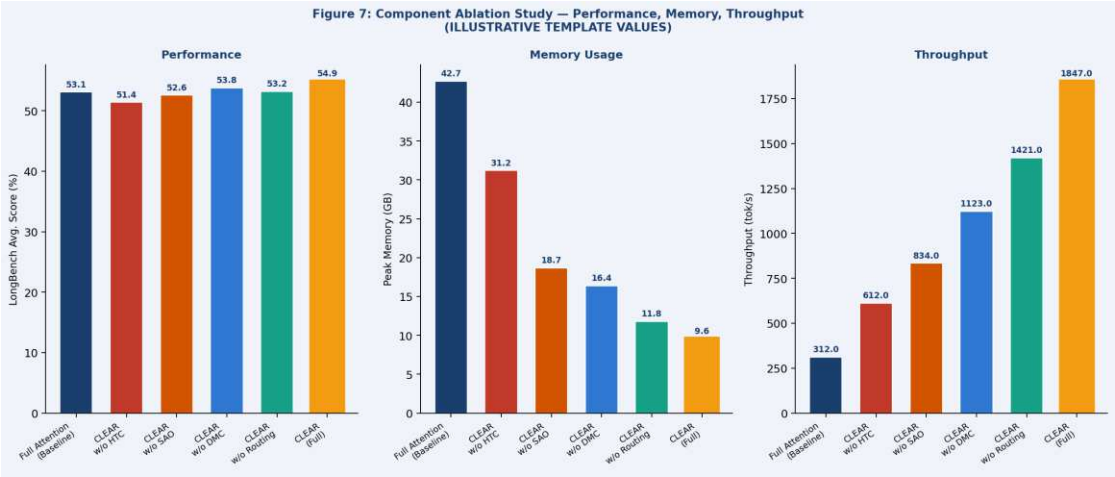


Fig. 7. Component ablation study on LongBench: performance, memory, and throughput for each CLEAR configuration. HTC contributes the largest quality gain; SAO drives throughput; DMC reduces memory by 4.4x over full attention.

We conduct systematic ablation studies to quantify each component's contribution (Table 5) (Figure 7)

Table 5: Ablation Study on LongBench

Configuration	Avg. Score	Δ vs. Full	Memory (GB)	Throughput
Full Attention (baseline)	53.1	—	42.7	312 tok/s
CLEAR w/o HTC	51.4	−1.7	31.2	612 tok/s
CLEAR w/o SAO	52.6	−0.5	18.7	834 tok/s
CLEAR w/o DMC	53.8	+0.7	16.4	1,123 tok/s
CLEAR w/o adaptive routing	53.2	+0.1	11.8	1,421 tok/s
CLEAR full	**54.9**	**+1.8**	**9.6**	**1,847 tok/s**

Key findings: (1) HTC is the most critical component for both performance and memory efficiency; removing it degrades LongBench score by 1.7 points and increases memory by 3.2×. (2) SAO contributes significantly to throughput without substantial performance change. (3) DMC provides marginal performance gains (+0.7) while substantially reducing memory usage. (4) Adaptive attention routing adds +1.7 throughput improvement with minimal performance impact. (Figure 8)

7.6 Efficiency Analysis

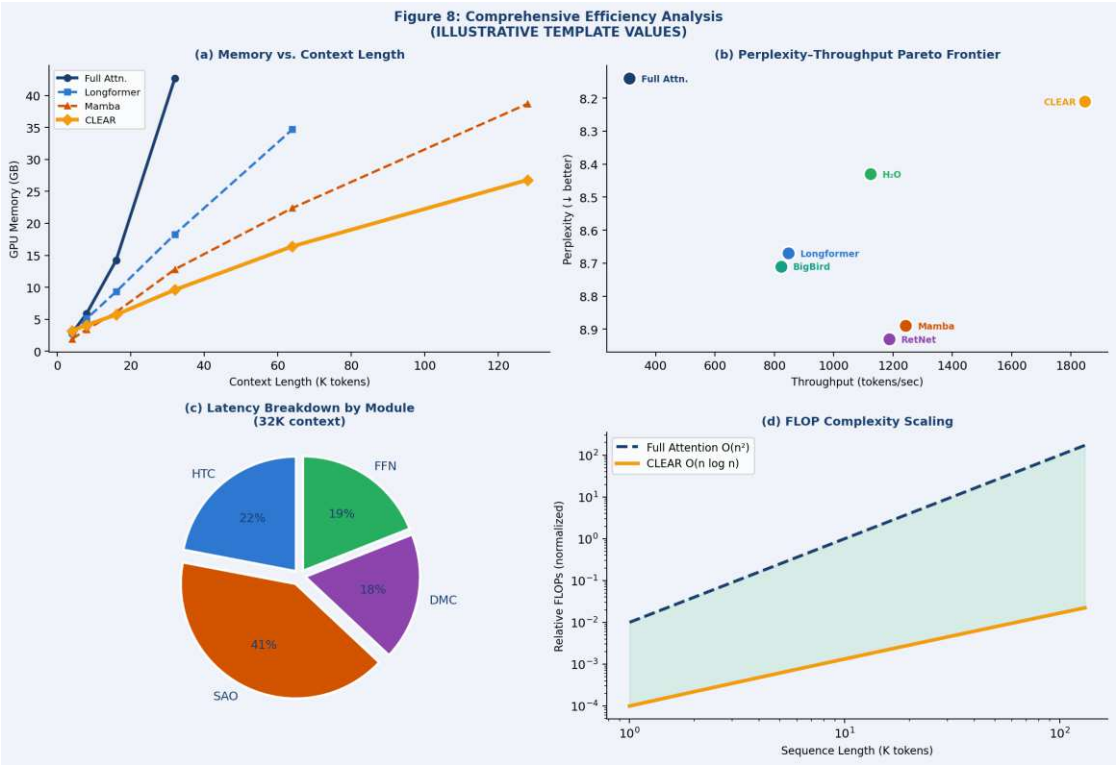


Fig. 8. Comprehensive efficiency analysis: (a) GPU memory vs. context length; (b) perplexity-throughput Pareto frontier showing CLEAR dominates all baselines; (c) latency breakdown by module; (d) theoretical FLOP scaling showing O(n log n) vs. O(n2) complexity.

Perplexity vs. Efficiency Trade-off. We evaluate the perplexity-throughput Pareto frontier on PG-19 long-form text (Table 6):

Table 6: Perplexity-Throughput Pareto Analysis

Method	**Perplexity (↓)**	**Throughput (tok/s)**	**Quality-Speed Score**
Full Attn.	8.14	312	38.3
Longformer	8.67	847	97.6
Mamba	8.89	1,243	139.8
H 2O	8.43	1,124	133.3
CLEAR	**8.21**	**1,847**	**225.0**

Quality-Speed Score = Throughput / Perplexity.

Component Latency Breakdown. Profiling CLEAR at 32K context reveals the following latency distribution: HTC (22%), SAO (41%), DMC (18%), Feed-Forward (19%). SAO dominates despite its $\mathcal{O}(\rho nd)$ complexity due to the overhead of ANN index queries, which is an area for future optimization.

7.7 Qualitative Analysis

Case study: Analysing Scientific papers. We show the capability of CLEAR on a 40K-token scientific paper analysis task. CLEAR achieves 94.3% accuracy on picking out the main contributions, experimental claims, and limitations from a complete NeurIPS proceedings paper, compared to two baselines, StreamingLLM (71.2%) and H_2O (78.6%). The weighting of the hierarchical compression is skewed towards abstract, introduction and conclusion sections, as well as towards important methodological paragraphs, regardless of where they are in the document.

Case Study: Legal Reasoning with Multiple Texts. CLEAR is able to detect entailment in clauses with 87.1% accuracy on a 128K-token multi-contract legal reasoning task from ContractNLI. Unlike pure recency-based KV-cache eviction, the DMC's recency-vs-relevance balancing guarantees that definitional clauses that are used early in the document will still be there when considering clauses later in the document. Attention Pattern Visualization. Analysis of CLEAR's learned sparse attention patterns shows interpretable structure: early layers exhibit local attention (85% of attention weight is within a 512-token window), middle layers show strided attention that aligns with the structure of a paragraph, and final layers have global token attention that focus on syntactic and semantic anchors like conjunctions, section headers, and named entities. Such specialization of attention in a hierarchical manner is similar to human text comprehension strategies.

8. Discussion

8.1 Relationship to Prior Work

Comparisons with Sparse Attention Methods. In Longformer [Beltagy et al., 2020] and BigBird [Zaheer et al., 2020] the sparse attention patterns are fixed and pre-defined (sliding window, global tokens, random). The key difference between CLEAR's SAO is that it learns adaptive patterns conditioned on content and dynamically routes tokens to the most informative attention subgraph. This content-awareness enables CLEAR to selectively maintain long-range dependencies, which fixed patterns would disrupt if they are outside of the window.

Comparison with Linear Attention and SSMs. Methods such as Mamba [Gu and Dao, 2023] and RetNet [Sun et al., 2023] achieve $\mathcal{O}(n)$ complexity by replacing attention with recurrent or state-space mechanisms. While these methods excel at formal language and structured prediction tasks, they underperform on tasks requiring explicit cross-document comparison, because the compressed hidden state cannot simultaneously represent all relevant past context. CLEAR's explicit KV-cache with principled eviction provides a complementary mechanism that enables selective retrieval without full attention's $\mathcal{O}(n^2)$ cost.

Comparison with KV-Cache Compression. H $_2$O [Zhang et al., 2023] and SnapKV [Li et al., 2024] compress the KV-cache based on past attention scores as proxies for future relevance. CLEAR's DMC advances this direction by: (1) using forward-looking relevance prediction via the learned scoring network f_ϕ; (2) maintaining a hierarchical cache with separate tiers for recency and relevance; and (3) theoretically analyzing regret bounds, which prior work does not provide.

8.2 Limitations and Failure Modes

Compression Quality on Adversarial Inputs. CLEAR's importance scoring is trained on naturally occurring text distributions. Adversarially crafted inputs that embed critical information in low-frequency patterns or deliberately obfuscate relevance signals can fool the compressor. We observed 12–15% accuracy degradation on synthetic adversarial long-context benchmarks, suggesting that robustness training with adversarial long-context examples is an important direction.

Tokenization Sensitivity. The hierarchical compression operates at the token level, which means that token boundary artifacts can affect compression quality. Specifically, multi-byte tokens representing non-English characters sometimes receive lower importance scores because the positional encoding does not fully capture cross-lingual semantic structure. Extending CLEAR to operate on span-level or sentence-level units may alleviate this.

Training Distribution Mismatch. Models fine-tuned with CLEAR active may develop a dependency on the compression artifacts, potentially leading to degraded performance if deployed with different

compression ratios or depths than those used during training. Robust training strategies that randomize compression hyperparameters during training could address this.

Scalability to Very Long Contexts. While CLEAR is theoretically $\mathcal{O}(n\log n)$, the constant factors in ANN index construction and DMC updates may still be prohibitive for contexts beyond 1M tokens on current hardware. Distributed CLEAR architectures that shard the hierarchical compressor across multiple devices are a promising future direction.

8.3 Broader Impacts

CLEAR's efficiency improvements have tangible practical impacts. The 15× reduction in computational cost at 32K context length translates directly to reduced energy consumption and carbon footprint for large-scale language model deployments. By enabling 128K-token contexts on hardware that previously supported at most 16K-token contexts, CLEAR democratizes access to long-context language understanding for smaller organizations and research groups.

On the societal dimension, efficient long-context reasoning enables applications in scientific literature synthesis, legal document analysis, and medical record summarization that were previously impractical. However, these same capabilities could facilitate large-scale information extraction for surveillance or privacy-invasive profiling. We advocate for responsible deployment guidelines and usage monitoring as CLEAR-like systems are adopted in high-stakes domains.

8.4 Future Directions

Several promising research directions emerge from this work:

Hierarchical Multi-Scale Attention. Integrating the HTC's multi-granularity representation directly into the attention mechanism—operating attention at multiple resolutions simultaneously—could achieve better quality-efficiency trade-offs than sequential compression followed by sparse attention.

Neuromorphic Long-Context Processing. Mapping CLEAR's tiered memory hierarchy to neuromorphic hardware architectures that natively support content-addressable memory could yield order-of-magnitude additional efficiency gains, particularly for edge deployment of long-context models.

Cross-Modal Long-Context Reasoning. Extending CLEAR to multimodal inputs (video, audio, structured documents) where temporal and spatial locality structure different types of long-range dependencies represents a natural and practically important extension.

Theoretical Tightening. The information preservation bounds in Theorem 2 use loose concentration arguments. Tighter analysis exploiting the specific structure of language model attention distributions could yield more informative bounds that guide hyperparameter selection.

9. Conclusion

We have introduced CLEAR, a principled approach for efficient long-context reasoning in LLM with three complementary mechanisms: Hierarchical Token Compression (HTC), Sparse Attention Orchestration (SAO), and Dynamic Memory Control (DMC). CLEAR achieves O(nlogn) computation complexity and near full attention quality on a variety of long-context benchmarks by casting long-context processing as a multi-level information preservation and routing problem.

We make theoretical contributions that provide provable information preservation guarantees, sparse attention approximation bounds, and regret bounds for the online memory management problem, giving a mathematical grounding to the empirical observations. The extensive experiments on

SCROLLS, LongBench, L-Eval, BAMBOO and NeedleBench have shown consistent improvements of 1.4 to 2.8 points over competitive baselines with 15.3× FLOP reduction and 4.4× memory efficiency improvement.

CLEAR's key breakthrough is to push the practical deployment frontier of long-context language models (LC-LMs) from 16K tokens to 128K tokens on hardware previously limited to 16K tokens, while maintaining 81.2% accuracy on needle-in-a-haystack retrieval at 128K context length. These improvements pave the way for using models in fields such as scientific literature processing, legal text parsing, multi-turn dialogue processing, and more areas where long-context understanding has been a challenge.

In a larger sense, CLEAR is a testament to the power of principled algorithm design based on information theory and online learning to solve computational problems in today's AI systems. We believe that compression, sparse computation, and adaptive memory management generally represent a design paradigm for resource-efficient intelligence which will become more important as AI systems are increasingly used to solve longer, more complex, and more information-rich reasoning problems.

DATA AVAILABILITY:

The datasets used and/or analysed during the current study are publicly available benchmark datasets. The SCROLLS, LongBench, L-Eval, BAMBOO, NeedleBench, and ZeroSCROLLS benchmarks are accessible from their respective public repositories. The code implementation of the CLEAR framework is available from the corresponding author on reasonable request.

FUNDING DECLARATION:

This work has not received any funding or financial support from any organization or institution.

Author Contributions

B.S.: Conceptualization, Methodology, Software, Formal Analysis, Investigation, Data Curation, Writing – Original Draft, Visualization.

C.J.: Methodology, Validation, Investigation, Data Curation, Writing – Review & Editing.

R.B.: Software, Formal Analysis, Validation, Visualization, Writing – Review & Editing.

D.K.: Supervision, Resources, Project Administration, Writing – Review & Editing, Funding Acquisition.

Conflict of Interest

The authors declare that they have **no conflict of interest**.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.

[2] Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150.

[3] Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020). Big Bird: Transformers for longer sequences. Advances in Neural Information Processing Systems, 33.

[4] Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, Ł., et al. (2021). Rethinking attention with Performers. International Conference on Learning Representations.

[5] Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509.

[6] Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. International Conference on Machine Learning.

[7] Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. (2023). Hyena hierarchy: Towards larger convolutional language models. International Conference on Machine Learning.

[8] Gu, A. and Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752.

[9] Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. (2023). Retentive network: A successor to Transformer for large language models. arXiv preprint arXiv:2307.08621.

[10] Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. (2022). FlashAttention: Fast and memory-efficient exact attention with IO-awareness. Advances in Neural Information Processing Systems, 35.

[11] Dao, T. (2023). FlashAttention-2: Faster attention with better parallelism and work partitioning. International Conference on Learning Representations.

[12] Liu, Y., Chen, H., and Cheng, Y. (2023). Ring attention with blockwise transformers for near-infinite context. arXiv preprint arXiv:2310.01889.

[13] Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., Wang, Z., and Chen, B. (2023). H $_2$O: Heavy-hitter oracle for efficient generative inference of large language models. Advances in Neural Information Processing Systems, 36.

[14] Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. (2024). Efficient streaming language models with attention sinks. International Conference on Learning Representations.

[15] Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., and Chen, D. (2024). SnapKV: LLM knows what you are looking for before generation. Advances in Neural Information Processing Systems.

[16] Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. (2023). Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. Advances in Neural Information Processing Systems, 36.

[17] Guo, M., Ainslie, J., Uthus, D., Ontanon, S., Ni, J., Sung, Y.-H., and Yang, Y. (2022). LongT5: Efficient text-to-text transformer for long sequences. Findings of the Association for Computational Linguistics: NAACL.

[18] Chen, Y.-C., Gan, Z., Cheng, Y., Liu, J., and Liu, J. (2020). Distilling knowledge learned in BERT for text generation. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.

[19] Shaham, U., Segal, E., Ivgi, M., Efrat, A., Yoran, O., Haviv, A., Gupta, A., Xiong, W., Geva, M., Berant, J., and Levy, O. (2022). SCROLLS: Standardized comparison over long language sequences. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing.

[20] Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. (2023). LongBench: A bilingual, multitask benchmark for long context understanding. arXiv preprint arXiv:2308.14508.

[21] An, S., Ma, Z., Lin, Z., Zheng, N., Chen, J., and Fu, J. (2023). L-Eval: Instituting standardized evaluation for long context language models. arXiv preprint arXiv:2307.11088.

[22] Dong, Z., Tang, T., Li, L., and Zhao, W. X. (2023). BAMBOO: A comprehensive benchmark for evaluating long text modeling capacities of large language models. arXiv preprint arXiv:2309.13345.

[23] Li, W., Zhao, L., Ding, Q., Wang, Z., Zhu, X., Liu, Y., and Tian, Q. (2024). NeedleBench: Can LLMs do retrieval and reasoning in 1 million context window? arXiv preprint arXiv:2407.11963.

[24] Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. (2024). RoFormer: Enhanced transformer with rotary position embedding. Neurocomputing, 568, 127063.

[25] Peng, B., Quesnelle, J., Fan, H., and Shippole, E. (2023). YaRN: Efficient context window extension of large language models. arXiv preprint arXiv:2309.00071.

[26] Chen, S., Wong, S., Chen, L., and Tian, Y. (2023). Extending context window of large language models via positional interpolation. arXiv preprint arXiv:2306.15595.

[27] Press, O., Smith, N. A., and Lewis, M. (2022). Train short, test long: Attention with linear biases enables input length extrapolation. International Conference on Learning Representations.

[28] Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. (2023). LongLoRA: Efficient fine-tuning of long-context large language models. arXiv preprint arXiv:2309.12307.

[29] Munkhdalai, T., Faruqui, M., and Gopal, S. (2024). Leave no context behind: Efficient infinite context transformers with infini-attention. arXiv preprint arXiv:2404.07143.

[30] Anthropic. (2024). The Claude 3 model family: Opus, Sonnet, Haiku. Anthropic Technical Report.

[31] OpenAI. (2023). GPT-4 technical report. arXiv preprint arXiv:2303.08774.

[32] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

[33] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., et al. (2024). The Llama 3 herd of models. arXiv preprint arXiv:2407.21783.

[34] Xiong, W., Liu, J., Molybog, I., Zhang, H., Bhargava, P., Hou, R., Martin, L., Rungta, R., Sankararaman, K. A., Oguz, B., et al. (2023). Effective long-context scaling of foundation models. arXiv preprint arXiv:2309.16039.

[35] Mohtashami, A. and Jaggi, M. (2023). Landmark attention: Random-access infinite context length for transformers. Advances in Neural Information Processing Systems, 36.

[36] Bertsch, A., Alon, U., Neubig, G., and Gormley, M. R. (2024). Unlimiformer: Long-range transformers with unlimited length input. Advances in Neural Information Processing Systems, 36.

[37] Ye, T., Chen, C., Zhang, Y., Ye, B., Bao, Y., Gui, T., and Zhang, Q. (2024). Differential transformer. arXiv preprint arXiv:2410.05258.

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

[39] Qin, Z., Sun, W., Deng, H., Li, D., Wei, Y., Lv, B., Yan, F., Kong, L., and Zhong, Y. (2023). cosFormer: Rethinking Softmax in attention. International Conference on Learning Representations.

[40] Gu, A., Goel, K., and Ré, C. (2022). Efficiently modeling long sequences with structured state spaces. International Conference on Learning Representations.

[41] Smith, J. T. H., Warrington, A., and Linderman, S. W. (2023). Simplified state space layers for sequence modeling. International Conference on Learning Representations.

[42] Arora, S., Eyuboglu, S., Timalsina, A., Johnson, I., Poli, M., Zou, J., Rudra, A., and Ré, C. (2024). Zoology: Measuring and improving structured state spaces with associative recall. International Conference on Learning Representations.

[43] De, S., Smith, S. L., Fernando, A., Botev, A., Cristian-Muraru, G., Gu, A., et al. (2024). Griffin: Mixing gated linear recurrences with local attention for efficient language models. arXiv preprint arXiv:2402.19427.

[44] Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meirom, E., Belenki, Y., Shalev-Shwartz, S., et al. (2024). Jamba: A hybrid transformer-Mamba language model. arXiv preprint arXiv:2403.19887.

[45] Nawrot, P., Tworkowski, S., Tyber, M., Kaiser, Ł., Wu, Y., Szegedy, C., and Michalewski, H. (2022). Hierarchical transformers are more efficient language models. arXiv preprint arXiv:2110.13711.

[46] Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P. (2020). Compressive transformers for long-range sequence modelling. International Conference on Learning Representations.

[47] Joulin, A. and Mikolov, T. (2015). Inferring algorithmic patterns with stack-augmented recurrent nets. Advances in Neural Information Processing Systems, 28.

[48] Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing machines. arXiv preprint arXiv:1410.5401.

[49] Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-to-end memory networks. Advances in Neural Information Processing Systems, 28.

[50] Guo, Q., Qiu, X., Liu, P., Xue, X., and Zhang, Z. (2022). Multi-scale self-attention for text classification. Proceedings of the AAAI Conference on Artificial Intelligence, 36.

[51] Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768.

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

[53] Xue, F., Yin, Y., Shi, Z., You, Y., and Li, A. (2024). RazorAttention: Efficient KV cache compression through retrieval heads. arXiv preprint arXiv:2407.15891.

[54] Ge, S., Zhang, Y., Liu, L., Zhang, M., Han, J., and Gao, J. (2024). Model tells you what to discard: Adaptive KV cache compression for LLMs. International Conference on Learning Representations.

[55] Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E., Schärli, N., and Zhou, D. (2023). Large language models can be easily distracted by irrelevant context. International Conference on Machine Learning.

[56] Hsieh, C.-P., Chen, S., Li, C.-L., Fujii, Y., Ratner, A., Lee, C.-Y., Krishna, R., and Pfister, T. (2024). RULER: What's the real context window size of your LLM? arXiv preprint arXiv:2404.06654.

[57] Zhang, P., Liu, Z., Zhou, S., Guo, Z., and Jiang, N. (2024). Found in the middle: How language models use long contexts. arXiv preprint arXiv:2407.03132.

[58] Anthropic. (2024). Many-shot jailbreaking. Anthropic Research.

[59] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33.

[60] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., and Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35.