Boosting Long-Context Information Seeking via Query-Guided Activation Refilling

Anonymous ACL submission

Abstract

Processing long contexts poses a significant challenge for large language models (LLMs) due to their inherent context window limitations and the computational burden of extensive key-value (KV) activations, which severely impact efficiency. For information-seeking tasks, full context perception is often unnecessary, as a query's information needs can dynamically range from localized details to a global perspective, depending on its complexity. However, existing methods struggle to adapt effectively to this dynamic information needs.

In the paper, we propose a method for processing long-context information-seeking tasks via query-guided **AC**tivation **RE**filling (ACRE). ACRE constructs a *Bi-layer KV Cache* for long contexts, where the layer-1 (L1) cache compactly captures global information, and the layer-2 (L2) cache provides detailed, localized information. ACRE establishes a proxying relationship between the two caches, allowing the input query to attend to the L1 cache and dynamically refill it with relevant entries from the L2 cache. This mechanism integrates global understanding with query-specific local details, thereby enhancing answer decoding. Experiments on a variety of long-context information-seeking datasets demonstrate ACRE's effectiveness, achieving significant improvements in both performance and efficiency. We provide our source codes in *this anonymous repository*.

1 Introduction

Recently, large language models (LLMs) have become widely used for daily information-seeking tasks, such as ChatGPT (OpenAI, 2023). However, their capabilities are inherently limited by the difficulty of updating parametric knowledge. To address this, incorporating external knowledge as context has become a common approach (Zhao et al., 2024). In practice, this external knowledge often involves long contexts, such as long documents or novels, which pose significant challenges due to the large KV activations accumulated during inference, demanding substantial computational resources and reducing efficiency (Xu et al., 2023; Bai et al., 2024b; Zhang et al., 2024c).

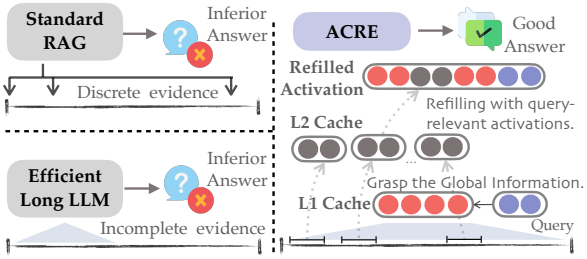

Figure 1: Comparison of ACRE, standard RAG, and efficient long LLMs for information-seeking tasks. Standard RAG retrieves evidence without full-context perception, and long LLMs struggle with contexts exceeding their native window. ACRE overcomes these limitations with a resource-efficient bi-layer KV cache and query-guided refilling, capturing both global and local information while enhancing performance.

To address the challenges posed by excessive KV activations, previous works have proposed various strategies: reducing the precision of activation tensors (Liu et al., 2024; Xu et al., 2024), dividing long contexts into smaller chunks for independent processing (Lee et al., 2024; Yoon et al., 2024), or compressing KV activations into shorter representations through selection or sparse attention (Zhang et al., 2023; Li et al., 2024; Xiao et al., 2024; Jiang et al., 2024). Retrieval-Augmented Generation (RAG) has also emerged as a promising approach, retrieving precise evidence from long contexts to support answer generation (Gao et al., 2024).

However, most existing methods follow a unilateral strategy: either compromising the semantic richness of KV activations to create compact global representations, such as with quantized activations (Liu et al., 2024), or concentrating solely on detailed local information, such as RAG methods (Gao et al., 2024). Moreover, most lightweight

KV methods remain constrained by the native context length limit, leading to significant performance degradation when processing contexts that exceed this limit (Zhang et al., 2024b).

In information-seeking tasks, we argue that the information needs of a user query can dynamically range from localized details to a global perspective, depending on the query's complexity. For instance, given a novel, the query "What are the main characters' names?" involves localized information needs and can be answered using specific local evidence. In contrast, the query "How do the main characters drive the story's development?" requires a global understanding of the entire book.

To address dynamic information needs in information-seeking tasks, we propose ACRE, a method that employs a bilateral strategy to capture a global perspective across the full context and enhance local details using query-guided activation refilling. Figure 1 presents an overview of ACRE's framework along with a comparison against efficient long LLMs and RAG methods.

Specifically, ACRE constructs a bi-layer KV activation cache for long contexts, comprising an L1 cache and an L2 cache. The L1 cache captures compact yet global information from the full context, while the L2 cache retains localized, detailed information. Notably, the L1 cache is significantly smaller than the L2 cache. During the forward pass of the LLM, the L1 and L2 caches are interleaved into a nested structure, with each L1 tensor optimized to proxy the semantics of its corresponding L2 cache. To enhance efficiency, we replace the original full attention mechanism—where each token attends to all preceding tokens—with a tailored selective attention mechanism. In this approach, tokens perform full attention on recent L1 and L2 tokens but only attend to distant L1 tokens. This selective attention mechanism significantly reduces computational costs, enabling ACRE to process long contexts more efficiently.

After the forward pass, the nested KV cache is decomposed back into separate L1 and L2 caches. For an input query, ACRE first uses the query to attend to the compact L1 cache. Based on the resulting attention score distribution, ACRE selectively refills key entries of the L1 cache with the corresponding L2 cache entries, thereby enriching local details. This process is referred to as query-guided activation refilling.

ACRE is trained through an efficient two-stage process. The first stage focuses on constructing the bi-layer KV cache, while the second stage targets query-guided activation refilling. Throughout both stages, ACRE updates only a small subset of model parameters, ensuring training efficiency.

We evaluate ACRE across a wide range of long-context information-seeking tasks (Bai et al., 2024b; Zhang et al., 2024c; Qian et al., 2024b). The experimental results confirm the effectiveness of ACRE. Our key contributions are summarized as follows: (1) We design a flexible and efficient bi-layer KV activation cache mechanism for long contexts, which captures compact global information while preserving local details. (2) We introduce ACRE, a method that leverages the bi-layer KV activation cache with a query-guided activation refilling mechanism to efficiently handle long-context information-seeking tasks. (3) We demonstrate that ACRE achieves superior performance on long-context information-seeking tasks, effectively handling contexts much longer than LLMs' typical context limits, while substantially reducing computational resources and latency.

2 Method

2.1 Preliminary

The process of solving information-seeking tasks using LLMs can be succinctly described as $\mathcal{Y} = \mathcal{M}(\mathcal{X})$, where $\mathcal{M}(\cdot)$ denotes the LLM, $\mathcal{Y}$ represents the output answer and $\mathcal{X}$ represents the input sequence. $\mathcal{X}$ can take various forms, ranging from a standalone query to a complex instruction prompt. In this paper, we focus on information-seeking tasks with long contexts. Therefore, we define the input sequence $\mathcal{X}$ as comprising a query q and a long context $\mathcal{C}$, denoted by $\mathcal{X} = (\mathcal{C}, q)$.

For the input $\mathcal{X}$, a Transformer-based LLM computes multi-head attention (MHA) as follows:

$$\boldsymbol{Q} = \boldsymbol{X} \cdot \boldsymbol{W}_Q, \quad (1)$$

$$\boldsymbol{K} = \boldsymbol{X} \cdot \boldsymbol{W}_K, \quad (2)$$

$$\boldsymbol{V} = \boldsymbol{X} \cdot \boldsymbol{W}_V, \quad (3)$$

$$\mathcal{A}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q} \cdot \boldsymbol{K}^\top}{\sqrt{d}}\right) \cdot \boldsymbol{V}, \quad (4)$$

where $\boldsymbol{X}$ represents the hidden states of the input sequence $\mathcal{X}$, and $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_V$ are the projection weight matrices for the query $\boldsymbol{Q}$, key $\boldsymbol{K}$, and value $\boldsymbol{V}$, respectively (Vaswani et al., 2023). The attention function $\mathcal{A}(\cdot)$ is applied iteratively across multiple layers and attention heads. For simplicity, we omit the layer and head indices.

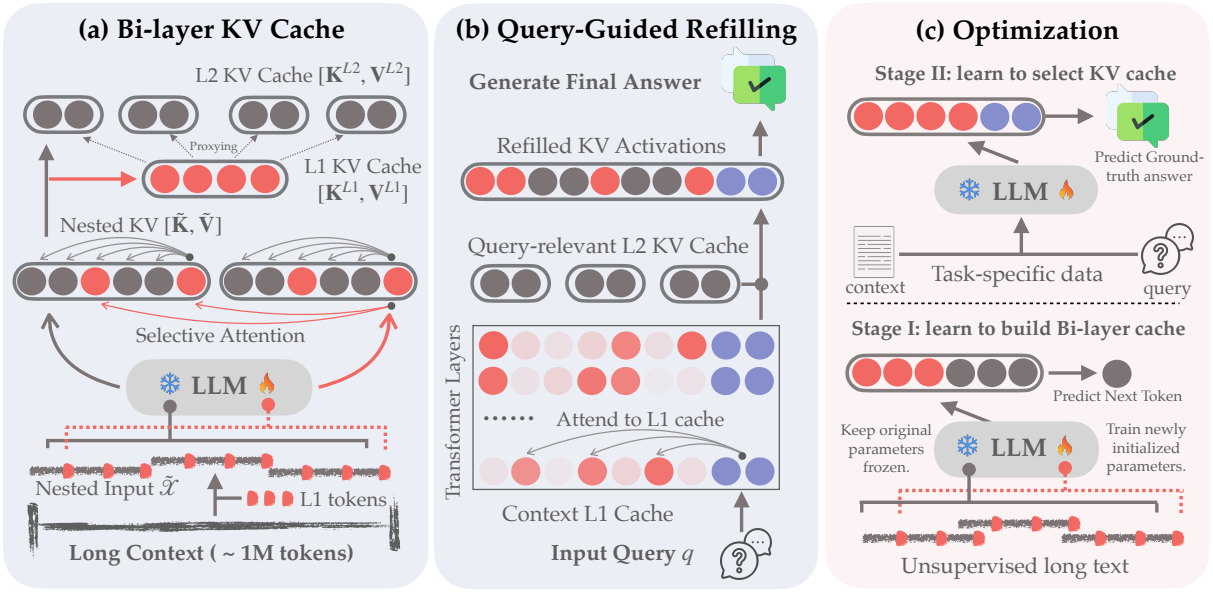

Figure 2: Overview of ACRE. (a) ACRE constructs the Bi-layer KV cache from a long context. (b) For an input query, ACRE refills the L1 KV cache with query-relevant entries from the L2 KV cache and decodes the final answer based on the refilled cache. (c) The two-stage optimization process used to train ACRE is illustrated.

The inference process of LLMs can be divided into two stages: (1) prefilling and (2) decoding (Liu et al., 2024). During the prefilling stage, the input sequence $\mathcal{X}$ is processed through each layer using MHA, and the layer-wise key-value activations $[\boldsymbol{K}, \boldsymbol{V}]$ are cached. These cached activations are reused in the decoding stage to avoid redundant computations, enabling efficient processing. However, as MHA computation has quadratic complexity with respect to the sequence length n, handling long contexts becomes computationally expensive. This often results in slow processing speeds and out-of-memory issues, particularly when dealing with long input contexts (Dong et al., 2023).

To address the challenges posed by oversized KV caches for long contexts, we propose ACRE, a framework that constructs a *Bi-layer KV Cache* and employs a *Query-Guided Refilling* mechanism to enable a flexible KV cache that captures both global context and query-specific local details, ensuring efficient and high-quality answer decoding.

2.2 Overview of ACRE

Figure 2 provides an overview of ACRE. Specifically, for a information-seeking task with a long context $\mathcal{C}$, ACRE organizes the long context into a bi-layer KV activation cache during the pre-filling stage, as shown in Figure 2 (a).

The construction of the *Bi-layer KV Cache* begins by interleaving newly introduced L1 tokens into the input context. Through model forwarding, a nested KV cache $[\tilde{\boldsymbol{K}}, \tilde{\boldsymbol{V}}]$ is obtained. This nested KV cache is then decomposed into a Bi-layer KV cache: the layer-1 (L1) cache, which is compact and stores global information from the full long context, and the layer-2 (L2) cache, which holds detailed and localized information. Each tensor in the L1 cache serves as a semantic proxy for a corresponding sequence of tensors in the L2 cache.

We denote the L1 KV cache as $[\boldsymbol{K}^{L1}, \boldsymbol{V}^{L1}] \in \mathbb{R}^{m \times d}$ and the L2 KV cache as $[\boldsymbol{K}^{L2}, \boldsymbol{V}^{L2}] \in \mathbb{R}^{n \times d}$. Here, the length of the L1 KV cache, m, is significantly smaller than n, the length of the L2 KV cache. To optimize memory usage, the L2 cache can be offloaded to CPU memory, while the L1 cache is retained in GPU memory as a constant cache after constructing the bi-layer KV cache. This design significantly improves memory efficiency in practical applications.

The *Bi-layer KV Cache* is constructed exclusively for input contexts, enabling it to be reused across different information-seeking tasks that share the same context. Given an input query q, ACRE utilizes q to attend to the L1 cache, computing attention scores. Based on these scores, ACRE selectively refills the L1 cache by retrieving the most informative entries from the L2 cache, which are proxied by the corresponding most attentive L1 cache tensors. This process recovers a partial nested cache to support answer decoding and is re-

ferred to as *query-guided activation refilling*, which is shown in Figure 2 (b).

By leveraging both the L1 KV cache and the query-specific L2 KV cache, the final KV cache captures global information from the full long context while preserving local details. This design significantly enhances the performance of long-context information-seeking tasks. In the following sections, we provide the technical details of ACRE.

2.3 Bi-Layer KV Cache

To construct the bi-layer KV cache, we introduce a new type of token, called L1 tokens, denoted as $\mathcal{X}^{L1} = (x_1^{L1}, \cdots, x_m^{L1})$. The original tokens of the input sequence are referred to as L2 tokens, denoted as $\mathcal{X}^{L2} = (x_1, \cdots, x_n)$. By interleaving the L1 and L2 tokens, the input sequence $\mathcal{X}$ is transformed into a nested sequence $\tilde{\mathcal{X}}$:

$$\tilde{\mathcal{X}} = (x_1, \cdots, x_l, x_1^{L1}, x_{l+1}, \cdots, x_n, x_m^{L1}), \quad (5)$$

where each L1 token is inserted after every l L2 tokens, acting as a semantic proxy for the preceding l L2 tokens. We refer to l as the L1/L2 interval. For the L1 tokens, we initialize an additional set of trainable weight matrices $\boldsymbol{W}_Q^{L1}$, $\boldsymbol{W}_K^{L1}$, and $\boldsymbol{W}_V^{L1}$, while keeping the original weight matrices for L2 tokens frozen.

After constructing the nested sequence $\tilde{\mathcal{X}}$, we adapt the attention computation defined in Eq. (4). Specifically, for the key $\boldsymbol{K}$, the original projection $\boldsymbol{K} = \boldsymbol{X} \cdot \boldsymbol{W}_K$ is replaced with:

$$\boldsymbol{K} = \begin{cases} \boldsymbol{x} \cdot \boldsymbol{W}_K^{L1}, & \text{if } x \text{ is an L1 token,} \\ \boldsymbol{x} \cdot \boldsymbol{W}_K, & \text{if } x \text{ is an L2 token,} \end{cases} \quad (6)$$

where $\boldsymbol{x} \in \boldsymbol{X}$. Through multi-head attention, this modification yields the nested key activations:

$$\tilde{\boldsymbol{K}} = [\boldsymbol{k}_1, \cdots, \boldsymbol{k}_l, \boldsymbol{k}_1^{L1}, \cdots, \boldsymbol{k}_n, \boldsymbol{k}_m^{L1}]. \quad (7)$$

Similarly, the nested value activations $\tilde{\boldsymbol{V}}$ are computed as:

$$\tilde{\boldsymbol{V}} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_l, \boldsymbol{v}_1^{L1}, \cdots, \boldsymbol{v}_n, \boldsymbol{v}_m^{L1}]. \quad (8)$$

By decomposing the nested KV cache, we obtain the bi-layer KV cache as follows:

$$\boldsymbol{K}^{L1} = [\boldsymbol{k}_1^{L1}, \cdots, \boldsymbol{k}_m^{L1}], \quad (9)$$

$$\boldsymbol{V}^{L1} = [\boldsymbol{v}_1^{L1}, \cdots, \boldsymbol{v}_m^{L1}], \quad (10)$$

$$\boldsymbol{K}^{L2} = [\underbrace{\boldsymbol{k}_1, \cdots, \boldsymbol{k}_l}_{\boldsymbol{k}_1^{L1}}, \cdots, \underbrace{\boldsymbol{k}_{n-l}, \cdots, \boldsymbol{k}_n}_{\boldsymbol{k}_m^{L1}}], \quad (11)$$

$$\boldsymbol{V}^{L2} = [\underbrace{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_l}_{\boldsymbol{v}_1^{L1}}, \cdots, \underbrace{\boldsymbol{v}_{n-l}, \cdots, \boldsymbol{v}_n}_{\boldsymbol{v}_m^{L1}}], \quad (12)$$

where $\underbrace{\boldsymbol{k}_1, \cdots, \boldsymbol{k}_l}_{\boldsymbol{k}_1^{L1}}$ represents the proxying relationship between the L1 cache and the L2 cache.

As previously mentioned, directly computing full attention over the long sequence $\mathcal{X}$ is both computationally expensive and resource-intensive. To efficiently construct the bi-layer KV cache, we propose a *selective attention mechanism*. This mechanism maintains a relatively small working context window $\mathcal{W}$, enabling current tokens to perform full attention on recent L1 and L2 tokens while only attending to distant L1 tokens. For instance, when computing KV activations at step n, we prune the previous KV cache $[\tilde{\boldsymbol{K}}, \tilde{\boldsymbol{V}}]$ as follows:

$$\tilde{\boldsymbol{K}} = [\boldsymbol{k}_1^{L1}, \cdots, \boldsymbol{k}_i^{L1}, \boldsymbol{k}_j, \cdots, \boldsymbol{k}_n, \boldsymbol{k}_m^{L1}], \quad (13)$$

$$\tilde{\boldsymbol{V}} = [\underbrace{\boldsymbol{v}_1^{L1}, \cdots, \boldsymbol{v}_i^{L1}}_{\text{distant L1 tokens}}, \underbrace{\boldsymbol{v}_j, \cdots, \boldsymbol{v}_n, \boldsymbol{v}_m^{L1}}_{\text{recent L1 / L2 tokens}}], \quad (14)$$

subject to the constraints $|\ \tilde{\boldsymbol{K}}\ | \leq \mathcal{W}$ and $|\ \tilde{\boldsymbol{V}}\ | \leq \mathcal{W}$. Through this mechanism, we sequentially process the full sequence $\tilde{\mathcal{X}}$ into KV activations using a short working context window, achieving both high computational efficiency and economical memory usage.

2.4 Query-Guided Activation Refilling

After constructing the bi-layer KV cache for the context, we obtain the L1 KV cache $[\boldsymbol{K}^{L1}, \boldsymbol{V}^{L1}]$, which serves as a global yet compact representation of the full long context, and the L2 KV cache $[\boldsymbol{K}^{L2}, \boldsymbol{V}^{L2}]$, which provides detailed but memory-intensive representations. To optimize memory usage, the L1 KV cache is retained as a constant cache in GPU memory, while the L2 KV cache is offloaded to CPU memory.

For an input query q, relying solely on the L1 KV cache is feasible but lacks query-specific detailed information. To address this limitation, ACRE proposes refilling the compact L1 KV cache with selected entries from the L2 KV cache that are most relevant for answering the query. Specifically, the query state $\boldsymbol{Q}_q$ for the input query q is computed as $\boldsymbol{Q}_q = \boldsymbol{q} \cdot \boldsymbol{W}_Q$. Using this query state, the attention distribution is calculated as: $\boldsymbol{A} = \text{softmax}\left(\frac{\boldsymbol{Q}_q \cdot \boldsymbol{K}^{L1\top}}{\sqrt{d}}\right)$, where $\boldsymbol{A} \in \mathbb{R}^{h \times m \times t}$, h is the number of attention heads, m is the length of L1 cache, and t is the length of the query q. The attention scores $\mathcal{S}$ are then obtained by applying mean pooling:

$$\mathcal{S} = \text{Pool}_{\text{dim}=0,2}(\boldsymbol{A}), \quad \mathcal{S} \in \mathbb{R}^m, \quad (15)$$

where $\mathcal{S}$ serves as a guiding signal to select relevant entries from the L2 KV cache. The selection process is defined as:

$$\mathcal{I} = \arg\operatorname{top}_k(\mathcal{S}), \tag{16}$$

$$k = \left\lfloor \frac{\min(\mathcal{W} - m, \eta)}{l} \right\rfloor, \tag{17}$$

where k is dynamically determined based on the maximum length of the predefined working context window $\mathcal{W}$ or the maximum refilling length η, and $\mathcal{I}$ represents the set of selected indices.

After selection, the L1 KV cache is refilled with the chosen entries from the L2 KV cache. For example, if $\mathcal{I} = \{2\}$, the refilled KV cache becomes:

$$\boldsymbol{K} = [\boldsymbol{k}_1^{L1}, \boldsymbol{k}_{l+1}, \cdots, \boldsymbol{k}_{2l}, \boldsymbol{k}_2^{L1}, \cdots, \boldsymbol{k}_m^{L1}], \tag{18}$$

$$\boldsymbol{V} = [\boldsymbol{v}_1^{L1}, \underbrace{\boldsymbol{v}_{l+1}, \cdots, \boldsymbol{v}_{2l}}_{\text{refilled L2 KV cache}}, \boldsymbol{v}_2^{L1}, \cdots, \boldsymbol{v}_m^{L1}]. \tag{19}$$

This refilling process is performed independently for each layer. With the refilled KV cache, ACRE decodes the final answer $\mathcal{Y}$ in a standard auto-regressive manner.

2.5 Model Optimization

ACRE is characterized by its *Bi-layer KV Cache* structure and *Query-Guided Activation Refilling* mechanism. Its effectiveness relies on two key abilities: (1) the L1 KV activations must faithfully represent the L2 KV activations, and (2) given an input query q, the most relevant L2 KV activations must be efficiently retrieved. To optimize these abilities, we employ a two-stage optimization strategy.

In stage 1, the objective is to maximize the semantic volume of the L1 KV activations to effectively represent the corresponding L2 KV activations. This is achieved by predicting the next token using the previously accumulated L1 tokens and the recent L2 tokens. The optimization can be expressed through a cross-entropy loss:

$$\mathcal{L}_{\text{stage-1}} = -\sum_{t=1}^{T} \log \mathcal{P}(x_t \mid x_{[1:i]}^{L1}, x_{[j:t-1]}), \tag{20}$$

where $x_{[1:i]}^{L1}$ denotes the accumulated L1 tokens, and $x_{[j:t-1]}$ denotes the recent L2 tokens.

In stage 2, the objective is to enable ACRE to retrieve the most relevant L2 KV activations for refilling the L1 KV cache based on an input query q. Since the L2 KV cache is proxied by the L1 KV cache, accurately attending to the most useful L1 KV activations allows retrieval of the corresponding L2 KV activations via the proxying relationship. To achieve this, we optimize ACRE using task-specific data comprising long contexts and input queries. The optimization employs the following loss function:

$$\mathcal{L}_{\text{stage-2}} = -\sum_{t=1}^{T} \log \mathcal{P}(y_t \mid \mathcal{X}^{L2}, q), \tag{21}$$

where y represents the ground-truth answer, and q is the input query. This loss ensures that ACRE learns to produce accurate answers solely based on the L1 KV cache while maintaining its ability to retrieve the most relevant L2 KV activations.

3 Experiments

3.1 Dataset

We evaluate ACRE and all baseline models across 12 information-seeking tasks from three public long-context benchmarks: LongBench (Bai et al., 2024b), InfiniteBench (Zhang et al., 2024c), and UltraDomain (Qian et al., 2024b). These 12 datasets are categorized as follows: (1) **Complex QA** (Qian et al., 2024b): Financial, Legal, Physics, Biology, Math, and CS. These tasks involve practical, high-level queries with extra-long contexts spanning specialized domains. Many queries demand a global and in-depth understanding of the full context, making them especially challenging. (2) **Single-Document QA**: NarrativeQA (Kociský et al., 2018), Qasper (Dasigi et al., 2021), MultiFieldQA (Bai et al., 2024b), and En.QA (Zhang et al., 2024c). (3) **Multi-Document QA**: 2WikiMQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022).

3.2 Baseline Models

We compare ACRE with the following baselines: **Original**: Directly fits the maximum context length of the underlying LLMs. **KIVI** (Liu et al., 2024): Quantizes KV activations into 4-bit precision. **Beacon** (Zhang et al., 2024a): Compresses the full KV activations into beacon activations. **SelfExtend** (Jin et al., 2024): Applies hierarchical positional encoding to extend the model's context window. **MInference** (Jiang et al., 2024): Dynamically applies different sparse attention mechanisms across all attention heads. **StreamingLLM** (Xiao et al., 2024): Attends only to recent tokens and sink tokens. **RAG**: Uses standard RAG pipelines

Table 1: Main experimental results. The best results are in bold, and the second-best are underlined. All methods use Qwen2.5-3B-Instruct as the underlying LLM. Baselines in the second block directly process the full context, while those in the third block divide the context into chunks and find evidence using a retriever. In the second row, ave($|\mathcal{C}|$)(k) means the average context length.

Dataset	nar	fin	legal	phy	bio	en.qa	math	cs	qas	mul	2wiki	mus
ave($\|\mathcal{C}\|$)(k)	18.4	40.6	51.4	105.8	125.3	192.6	197.9	215.9	3.6	4.6	4.9	11.2
	AVE. CONTEXT LENGTH > 16K								AVE. LENGTH < 16K			
Original	22.0	36.8	42.6	38.2	35.8	20.1	36.3	35.6	37.4	48.5	36.3	22.1
KIVI	21.1	27.0	39.5	35.3	33.2	15.6	32.1	33.4	37.1	46.1	35.0	22.1
Beacon	20.2	37.8	43.9	37.1	33.7	18.3	31.8	32.3	30.4	35.6	24.7	24.7
SelfExtent	20.8	37.5	40.0	29.1	29.9	11.4	31.6	30.4	36.0	49.6	37.1	25.1
StreamingLLM	18.8	27.3	26.2	31.4	27.4	8.3	30.0	26.9	33.4	38.6	32.1	12.2
MInference	22.2	35.6	37.2	32.9	28.5	8.9	30.3	27.1	36.2	48.6	36.0	23.5
RAG	18.9	36.9	38.6	22.1	18.4	11.3	19.2	19.3	38.6	46.6	37.8	20.8
RQRAG	19.0	37.0	39.0	28.0	23.0	12.0	26.1	24.1	37.6	47.3	37.4	21.8
MemoRAG	24.0	41.5	44.8	36.9	33.2	13.2	33.1	33.4	34.1	49.1	**38.0**	26.0
ACRE	**27.8**	**46.4**	**47.7**	**41.6**	**38.3**	**23.6**	**41.9**	**45.9**	**39.6**	**50.0**	36.4	**26.2**

to retrieve relevant evidence from the full context. **RQRAG** (Chan et al., 2024): Rewrites the input query into sub-queries and retrieves evidence for each sub-query. **MemoRAG** (Qian et al., 2024b): Applies a memory model to form a compact global memory over the full context, providing answer clues that assist the retrieval process for better evidence retrieval.

In the main experiments (Section 3.3), we use Qwen2.5-3B-Instruct as the underlying model. To analyze the impact of using different underlying models, we also experiment with Llama3.2-3B-Instruct and Qwen2.5-7B-Instruct in Section 3.4. All three LLMs have a native context window of 128K (Yang et al., 2024; MetaAI, 2024). The implementation details of ACRE and all baselines are in Appendix A.

3.3 Main Results

In Table 3.3, we present the results of the main experiments, demonstrating that ACRE outperforms all baselines across most datasets. These results highlight the effectiveness of ACRE's design. Specifically, we derive the following findings: (1) ACRE consistently outperforms the baseline approach of feeding the full context directly into LLMs. This improvement stems not only from ACRE's ability to process contexts exceeding the native LLM's context window but also from its precise focus on query-relevant local information, effectively filtering out irrelevant details through query-guided activation refilling. (2) Baselines in the second block generally perform worse than directly feeding the full context into LLMs. This is attributed to semantic loss caused by compressing full KV activations. In contrast, ACRE leverages its bi-layer KV cache and query-guided activation refilling to recover local detailed semantics from the L2 cache that are absent in the L1 cache, resulting in superior performance. (3) Baselines in the third block use retrieval tools to extract precise evidence from long contexts. While effective for queries with clear information needs, these methods struggle with complex queries that require a higher-level understanding of the full context. ACRE overcomes this limitation by utilizing the global information in the L1 cache and dynamically refilling it with query-relevant local details from the L2 cache, thereby adapting to the varying information needs of different queries.

3.4 Ablation Study

To thoroughly validate the effectiveness of our method design, we perform detailed ablation studies as follows:

(1) **Method Design and Model Selection**: Figure 3 presents ablation results across different LLMs and variations in model design. First, we evaluate the role of training stages in model performance. Without the two-stage training process, ACRE reverts to a vanilla LLM, which performs significantly worse than ACRE. Stage-1 training enables ACRE to construct the bi-layer KV activation cache, thereby improving its long-context processing capabilities. When both stages are applied, ACRE achieves the best performance, demonstrating the effectiveness of its optimization design.

Second, to determine if ACRE's effectiveness

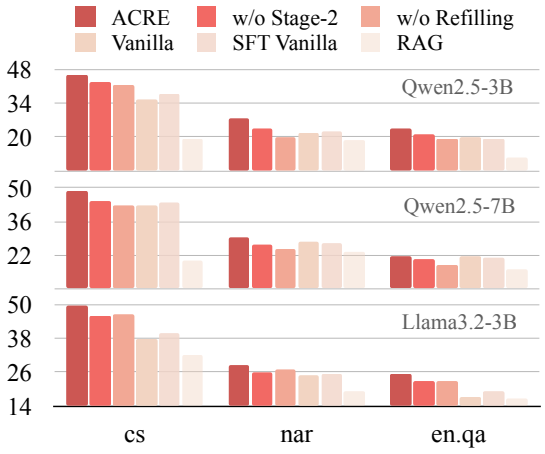

Figure 3: Ablation Study on Model Design Variations Across Different LLMs.

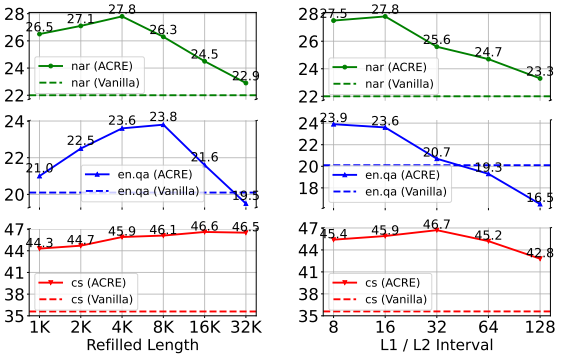

Figure 4: Analysis of the maximum refilling length η (left) and the impact of the L1/L2 interval l (right).

stems from its training data, we fine-tune a vanilla model using ACRE's training data via SFT, producing *SFT Vanilla*. While SFT improves the vanilla model by enhancing its QA capabilities, it still underperforms compared to ACRE. This highlights the unique advantages of ACRE'sdesign.

Lastly, we replace ACRE's underlying LLM with Qwen2.5-7B (a scaled-up version of the same model) and Llama3.2-3B (a model of similar scale but different architecture). As shown in Figure 3, ACRE's design consistently proves effective across models of varying scales and architectures, confirming its generalizability.

(2) **Impact of Parameter Choice**: As described in Section 2, ACRE's performance may be influenced by two hyperparameters: the maximum refilling length of KV activations η and the L1/L2 interval l. To investigate their impact, we conduct experiments with different values of η and l. Figure 4 presents the results of this analysis.

Specifically, in the left figure, we observe that the impact of the refilled activation length varies by task. For tasks with queries requiring explicit information (e.g., `nar` and `en.qa`), answer decoding relies on precise local information. Here, ACRE's performance peaks at a reasonable refilled length but declines as excessive refilling introduces noise, which biases the decoding process. Conversely, for tasks with queries requiring the integration of global information, ACRE's performance consistently improves with longer refilled lengths. This is because the L1 cache already provides global information, and additional refilled activations enhance local context.

The right figure shows the impact of the L1/L2 interval. We find that ACRE's performance generally decreases as the L1/L2 interval increases. Larger intervals require L1 tokens to summarize more semantics from subsequent L2 tokens, potentially overloading the L1 cache. However, larger intervals result in a compact L1 KV cache, offering efficiency. In practical applications, users can adjust parameters to balance efficiency and effectiveness based on available resources.

In summary, ACRE outperforms directly using vanilla LLMs in most parameter settings, requiring significantly fewer computational resources while achieving higher efficiency.

3.5 Efficiency Analysis

To evaluate ACRE's efficiency compared to baselines in processing long contexts at different scales, we conduct comparative experiments using the vanilla LLM, the efficient attention method MInference, and ACRE.

The results, presented in Table 2, lead to the following conclusions: (1) ACRE consistently processes long contexts at different scale with comparable or lower GPU resource usage. This efficiency is attributed to the bi-layer KV activation design, which avoids directly processing the full KV activations. (2) ACRE's efficiency advantage becomes more pronounced with extremely long contexts (e.g., over 512K), where the vanilla LLM runs out of memory, and MInference faces a high risk of out of memory while require longer latency than ACRE. (3) Thanks to its query-guided activation refilling mechanism, ACRE utilizes only the compact L1 KV activations and query-relevant L2 KV activations for answer decoding. This enables ACRE to process contexts longer than the native window of the LLM while maintaining answer quality. In contrast, baseline models generate nonsensical answers when exceeding LLM's native context length.

Table 2: Efficiency comparison of Vanilla LLM, MInference, and ACRE. Peak GPU memory (`mem`, GiB), time latency (`lat`, seconds/query), and answer readability (`rdbl`) are evaluated using 20 samples with contexts over 1024K, truncated to target lengths, and a max generation length of 100 tokens. Tests are conducted on a single NVIDIA A800 80G GPU. Average scores are reported, with the best in each block highlighted in bold.

Length	64K			128K			256K			512K			1024K		
	mem	lat	rdbl	mem	lat	rdbl	mem	lat	rdbl	mem	lat	rdbl	mem	lat	rdbl
QWEN2.5-3B-INSTRUCT-128K															
Vanilla	18.5	12.1	✓	27.9	36.3	✓	49.1	103.2	✗	OOM	-	✗	OOM	-	✗
MInfer.	**15.5**	29.2	✓	**22.0**	33.6	✓	28.0	57.1	✗	**39.1**	58.9	✗	47.2	79.6	✗
ACRE	20.8	**8.4**	✓	23.0	**14.3**	✓	**27.6**	**28.1**	✓	44.3	**48.2**	✓	**46.8**	**53.6**	✓
QWEN2.5-7B-INSTRUCT-128K															
Vanilla	31.9	21.2	✓	46.1	45.3	✓	78.3	129.6	✗	OOM	-	✗	OOM	-	✗
MInfer.	**27.9**	29.1	✓	**34.3**	35.6	✓	48.1	81.2	✗	74.2	132.7	✗	OOM	-	✗
ACRE	31.3	**10.5**	✓	35.1	**18.0**	✓	**43.0**	**37.1**	✓	**72.1**	**85.6**	✓	**75.6**	**90.4**	✓

In summary, ACRE demonstrates significant advantages in handling long contexts efficiently and reliably compared to baseline methods.

4 Related Work

Long-context processing is a critical capability of LLMs (Zhao et al., 2024). The most fundamental approach to enhancing this ability is training LLMs on long texts, either sampled from raw corpora or synthesized (Xiong et al., 2024; Mohtashami and Jaggi, 2024; Fu et al., 2024; Bai et al., 2024a). Consequently, the native context window of popular LLMs has increased significantly, from the earlier 4K to the current 128K (Peng et al., 2023; Touvron et al., 2023; Yang et al., 2024).

In addition to directly increasing the context window, some methods employ strategic positional encoding to enable LLMs to process contexts longer than their native window, as demonstrated by (Chen et al., 2023b; Song et al., 2023; Liu et al., 2023; Jin et al., 2024). However, when processing long contexts, LLMs generate large key-value (KV) activations, which consume substantial resources and reduce efficiency. To address this, many works aim to make KV activations more compact and lightweight (Liu et al., 2024; Xu et al., 2024). For example, KIVI focuses on reducing the precision of KV activations to 2-bit, resulting in significantly lighter KV representations (Liu et al., 2024). Other methods selectively attend to a small portion of KV activations through compression or sparse attention mechanisms. For instance, StreamingLLM proposes attending only to recent tokens and sink tokens to maintain compact KV activations (Xiao et al., 2024), similar idea also adopted by (Li et al., 2024; Zhang et al., 2023; Jiang et al., 2024; Zhang et al., 2024a). Beyond optimizing KV activations, alternative methods such as agent-based approaches (Qian et al., 2024a; Lee et al., 2024) and retrieval-augmented generation (Xu et al., 2023; Zhu et al., 2024) have been applied to facilitate long-context processing. These methods split the long context into chunks and retrieve evidence using retrievers or agents. They work well for explicit queries but struggle with implicit ones requiring full-context aggregation.

Most existing methods either compact global KV activations into a lightweight form or prune them into shorter forms, often failing to balance global perspective with local informativeness. This limitation can compromise performance in information-seeking scenarios, where information needs may dynamically range from global to local.

5 Conclusion

In this paper, we propose a method, ACRE, designed to adapt to the dynamic information needs of long-context information-seeking tasks. ACRE constructs a bi-layer KV activation cache structure for long contexts, where the L1 KV cache stores compact, global information, and the L2 KV cache captures detailed, local information. Using query-guided activation refilling, ACRE identifies query-specific evidence from the L2 KV cache and refills this local information into the L1 KV cache, resulting in nested KV activations that effectively combine a global perspective with local details. Through experiments on a wide range of information-seeking datasets, we demonstrate the effectiveness of ACRE in simultaneously improving the performance and efficiency of long-context processing for information-seeking tasks.

Limitation

In this paper, we propose ACRE, a method designed to adapt to the dynamic information needs of long-context information-seeking tasks. ACRE constructs a bi-layer KV activation cache to balance global context perception and local detail preservation, leveraging query-guided activation refilling to enhance performance and efficiency. While ACRE demonstrates significant advancements, several limitations are worth noting:

(1) Our method is primarily designed for information-seeking tasks, a major subset of long-context processing. This focus is largely driven by the availability of training data, as information-seeking tasks benefit from abundant QA datasets. While ACRE has the potential to adapt to general long-context tasks, further exploration with diverse task-specific data would be necessary to validate its broader applicability.

(2) ACRE introduces additional parameters for constructing the bi-layer KV cache, increasing the model size. For example, using Qwen2.5-3B-Instruct, ACRE adds approximately 17.2% more parameters, requiring additional GPU memory to load the model. However, in long-context tasks, the majority of GPU memory is consumed by KV activations rather than model parameters. Our efficiency analysis confirms that ACRE reduces overall GPU memory consumption when processing long contexts, mitigating this limitation to some extent.

(3) A portion of our training data is synthetically generated by commercial LLMs (e.g. GPT-4), which may introduce biases inherited from the original corpus or the LLMs used. While such biases could impact performance, many current commercial LLMs incorporate robust safeguards that help mitigate these issues. Nonetheless, addressing potential biases in synthetic data remains an area for future improvement.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024a. Longalign: A recipe for long context alignment of large language models. *arXiv preprint arXiv:2401.18058*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024b. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3119–3137. Association for Computational Linguistics.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. RQ-RAG: learning to refine queries for retrieval augmented generation. *CoRR*, abs/2404.00610.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2023a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2309.07597.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023b. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. *Preprint*, arXiv:2309.12307.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610.

Zican Dong, Tianyi Tang, Lunyi Li, and Wayne Xin Zhao. 2023. A survey on long text modeling with transformers. *arXiv preprint arXiv:2302.14502*.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024. Data engineering for scaling language models to 128k context. *Preprint*, arXiv:2402.10171.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *arXiv preprint arXiv:2407.02490*.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen,

and Xia Hu. 2024. Llm maybe longlm: Self-extend llm context window without tuning. *Preprint*, arXiv:2401.01325.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Trans. Assoc. Comput. Linguistics*, 6:317–328.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John F. Canny, and Ian Fischer. 2024. A human-inspired reading agent with gist memory of very long contexts. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*.

Xiaoran Liu, Hang Yan, Chenxin An, Xipeng Qiu, and Dahua Lin. 2023. Scaling laws of rope-based extrapolation. In *The Twelfth International Conference on Learning Representations*.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. In *Forty-first International Conference on Machine Learning*.

MetaAI. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Amirkeivan Mohtashami and Martin Jaggi. 2024. Random-access infinite context length for transformers. *Advances in Neural Information Processing Systems*, 36.

OpenAI. 2023. Gpt-4 technical report. `https://cdn.openai.com/papers/gpt-4.pdf`.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Yujia Zhou, Xu Chen, and Zhicheng Dou. 2024a. Are long-llms a necessity for long-context tasks? *Preprint*, arXiv:2405.15318.

Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. 2024b. Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery. *Preprint*, arXiv:2409.05591.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Woomin Song, Seunghyuk Oh, Sangwoo Mo, Jaehyung Kim, Sukmin Yun, Jung-Woo Ha, and Jinwoo Shin. 2023. Hierarchical context merging: Better long context understanding for pre-trained llms. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. 2024. Effective long-context scaling of foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 4643–4663. Association for Computational Linguistics.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Retrieval meets Long Context Large Language Models. *arXiv*. Experimental.

Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. 2024. Think: Thinner key cache by query-driven pruning. *arXiv preprint arXiv:2407.21018*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. 2024. Compact: Compressing retrieved documents actively for question answering. *Preprint*, arXiv:2407.09014.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2024a. Soaring from 4k to 400k: Extending llm's context with activation beacon. *arXiv preprint arXiv:2401.03462*.

Peitian Zhang, Ninglu Shao, Zheng Liu, Shitao Xiao, Hongjin Qian, Qiwei Ye, and Zhicheng Dou. 2024b. Extending llama-3's context ten-fold overnight. *Preprint*, arXiv:2404.19553.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024c. ınftybench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15262–15277. Association for Computational Linguistics.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. *Preprint*, arXiv:2306.14048.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2024. A survey of large language models. *Preprint*, arXiv:2303.18223.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zhicheng Dou, and Ji-Rong Wen. 2024. Large language models for information retrieval: A survey. *Preprint*, arXiv:2308.07107.

A Implementation details

For ACRE training, in stage 1, we sample long text spans from the RedPajama (Soboleva et al., 2023) dataset to create a training set of 2 billion tokens. The sampled text lengths are limited to a minimum of 4K and a maximum of 64K tokens. We randomly choose L1/L2 interval from $l \in \{8, 16, 32, 64, 128\}$. The model is trained for one epoch with a batch size of 8 and a learning rate of 5×10^{-5}. In stage 2, we collect 28,400 QA SFT data points from LongAlpaca (Chen et al., 2024) and synthetic data from (Zhang et al., 2024a; Qian et al., 2024b). We apply the same L1 token insertion strategy during training. The model is trained for three epochs with a batch size of 8 and a learning rate of 1×10^{-5} for two epochs. Stage-1 training takes around 7 hours while stage-2 training takes around 13 hours.

During the two-stage training process, we optimize only the newly initialized parameters, keeping the original parameters frozen. The number of trainable parameters varies depending on the model. For instance: (1) When using Qwen2.5-3B-instruct, ACRE has around 503M trainable parameters, accounting for 17.2% of the original parameters. (2) When using Llama3.2-3B-instruct, ACRE has around 780M trainable parameters, accounting for 25.6% of the original parameters. This difference arises from variations in the implementation of multi-head attention.

For the main experiments, we configure ACRE with an L1/L2 interval l of 16, a maximum refilling length η of 4,096, and the maximum working context window $\mathcal{W}$ of 32K tokens. For the Bi-Layer KV Cache construction, we utilize the following prompt. During the Query-Guided Activation Refilling process, we adopt task-specific prompts from the official benchmark repositories, without inserting the context into the task prompt.

Prompt for Bi-Layer KV Cache Construction

You are provided with a long article. Read the article carefully.
After reading, you will be asked to perform specific tasks based on the content of the article.
Now, the article begins:
Article Content: [context]
The article ends here.
Next, follow the instructions provided to complete the tasks.

For RAG, RQ-RAG, and MemoRAG, we employ BGE-M3 (Chen et al., 2023a) as the retriever and set the hit number to 5. For methods that divide the long context into chunks, we use the semantic-text-splitter tool, chunking the context to a maximum length of 512 tokens.

For KIVI, we quantize the KV activations to 4-bit precision. For Beacon, we use the official training code to fine-tune Qwen2.5-3B-Instruct, setting the compression ratio to 8 during inference. For SelfExtend, we set the group size to 32 and the window size to 2048, which is approximate by the official recommended strategy. For StreamingLLM, we use the SinkCache implementation from Transformers, configuring the window size to 4096 and

the number of sink tokens to 8. Lastly, for MemoRAG, we utilize the officially released memorag-qwen2-7b-inst as the memory model.

All methods are evaluated using the task prompts provided in the official repositories of their corresponding benchmarks[1]. Additionally, we use the same generation hyper-parameters (task-dependent) for ACRE and all baseline models.

All training and evaluation experiments were conducted using 8 NVIDIA A800-80G GPUs.

[1]LongBench: https://github.com/THUDM/LongBench, InfiniteBench: https://github.com/OpenBMB/InfiniteBench