Sparse Attention Across Multiple-Context KV Cache

Ziyi Cao[1,*,†], **Qingyi Si**[2,†], **Jingbin Zhang**[2,†], **Bingquan Liu**[1]

[1]Harbin Institute of Technology
[2]Huawei Technologies Co., Ltd.
zyc@stu.hit.edu.cn, siqingyi@huawei.com, zhangjingbin@pku.edu.cn, liubq@hit.edu.cn

Abstract

Large language models face significant cost challenges in long-sequence inference. To address this, reusing historical Key-Value (KV) Cache for improved inference efficiency has become a mainstream approach. Recent advances further enhance throughput by sparse attention mechanisms to select the most relevant KV Cache, thereby reducing sequence length. However, such techniques are limited to single-context scenarios, where historical KV Cache is computed sequentially with causal-attention dependencies. In retrieval-augmented generation (RAG) scenarios, where retrieved documents as context are unknown beforehand, each document's KV Cache is computed and stored independently (termed multiple-context KV Cache), lacking cross-attention between contexts. This renders existing methods ineffective. Although prior work partially recomputes multiple-context KV Cache to mitigate accuracy loss from missing cross-attention, it requires retaining all KV Cache throughout, failing to reduce memory overhead. This paper presents **SamKV**, the first exploration of attention sparsification for multiple-context KV Cache. Specifically, SamKV takes into account the complementary information of other contexts when sparsifying one context, and then locally recomputes the sparsified information. Experiments demonstrate that our method compresses sequence length to 15% without accuracy degradation compared with full-recomputation baselines, significantly boosting throughput in multi-context RAG scenarios.

1 Introduction

Large language models (LLMs) (Vaswani et al. 2017; Devlin et al. 2019) have demonstrated remarkable capabilities across a multitude of domains, including question answering, chatbots, education, and healthcare. They process text-like token sequences submitted by users to provide responses. However, as requests (user query) become more intricate, the challenge of serving LLMs effectively grows more pronounced. This is particularly evident in complex tasks such as multi-context question answering and few-shot learning within retrieval-augmented generation (RAG) scenarios (Lewis et al. 2020). These tasks often involve immutable token chunks (e.g., system messages, examples, and contexts) that remain unchanged and are repeatedly used across different requests. The key challenge lies in how to effectively leverage these recurring text chunks to enhance the efficiency of LLM services.

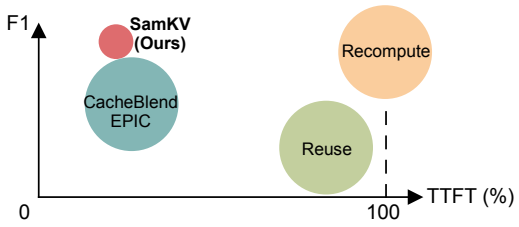

Figure 1: KV Cache methods across multiple contexts. The x-axis shows time to first token (TTFT) as a percentage of full recomputation. The y-axis represents F1 scores, with circle sizes indicating GPU memory usage during inference.

Multi-context methods	CacheBlend	EPIC	Ours
Sequence ratio	100%	100%	14.9%
Recomputation ratio	15.0%	14.1%	14.3%

Table 1: Sequence ratio indicates the proportion of KV Cache that requires loading in the GPU, and recomputation ratio indicates the ratio of tokens to be recomputed.

To address the challenges of LLM serving, context caching has emerged as a key strategy to enhance efficiency by reusing precomputed Key-Value (KV) Caches (Pope et al. 2023) for repeated tokens. Context caching methods can be broadly categorized into two types: single-context and multi-context approaches. Single-context methods often concentrate on sparsifying the KV Cache, aiming to reduce the computational cost and memory footprint associated with LLM inference (Li et al. 2024; Liu et al. 2024; Xiao et al. 2024a). However, these single-context approaches are not designed to address the absence of cross-attention that occurs when multiple contexts are prefilled separately. In contrast, multi-context techniques typically focus on recomputing specific tokens' KV Caches (Yao et al. 2024; Hu et al. 2024). This approach can somewhat alleviate the lack of cross-attention between different text chunks. Yet, it comes at a cost. These methods generally require the

[*]Work done during the internship at Huawei.
[†]These authors contributed equally.

entire KV Cache to be loaded into memory, even if only a fraction of the tokens' KV Caches are recomputed. This means that while they improve the accuracy of the generated responses by addressing cross-attention issues, they do not sparsify the KV Cache. As a result, the GPU memory usage remains high, posing a challenge for efficient LLM serving, especially when dealing with extensive contextual information.

To break through these limitations, we introduce SamKV, a novel method that combines sparsification and selective recomputation to efficiently handle multiple contexts. Our approach consists of two main steps. First, we perform sparsification by filtering out irrelevant content from the current document based on the user query and other documents' information. This step ensures that only the most useful content is retained, significantly reducing the amount of data to be processed. Second, we selectively recompute the KV Caches for a subset of tokens within the sparsified content. This selective recomputation focuses on updating the KV Caches for tokens that are crucial for maintaining cross-attention and generating high-quality responses. By integrating sparsification and selective recomputation, SamKV achieves both acceleration and reduced memory usage, making it more efficient to handle multiple contexts without the need to load the entire KV Cache into memory.

Specifically, during the sparsification process, SamKV exclusively utilizes the KV Cache from initial and local positions in the sequence to generate query vectors. To enhance cross-document information extraction, we further enrich each context's query vector by incorporating query-relevant signals from other contexts. For instance, when answering a crime-related question, SamKV fetches the KV Cache of semantically linked documents (e.g., legal statutes and jurisprudential analyses). By augmenting the query vector for jurisprudential analysis with features from legal statutes, the system more effectively captures their inter-document correlations, enabling personalized query embeddings per context. Moreover, during token recomputation, SamKV selectively processes only the sparsified KV Cache, prioritizing tokens from initial/local positions and those with high attention scores. This dual strategy reduces computational overhead while preserving model accuracy.

In this paper, we introduce SamKV, a novel approach that achieves significant compression of multi-context KV Caches without compromising accuracy. Specifically, SamKV reduces the KV Cache size to only 15% of the original while maintaining the same level of precision. Furthermore, we find that incorporating information from other contexts into the user query during the sparsification process yields more critical sparse information. Additionally, we observe that integrating the newly recomputed KV values for the sparse tokens back into the original KV Cache can sometimes enhance the overall performance.

Our contributions are primarily as follows:

- We present the first approach to achieve KV Cache sparsification in multi-context scenarios, which significantly reduces GPU memory usage.
- We propose an innovative multi-context sparsification approach and a novel token recomputation method, both of which achieve promising results.
- Our method, validated on the LongBench's Question-Answering (QA) datasets (Bai et al. 2024), demonstrates the ability to reduce computational load and accelerate processing while maintaining accuracy.

2 Related Work

2.1 Single-Context Sparsification

Optimizing LLMs for long-context processing has focused on single-context sparsification techniques. For example, InfLLM (Xiao et al. 2024a) uses sliding window attention with an efficient context memory, while SnapKV (Li et al. 2024) compresses the KV Cache by identifying critical attention features. However, these methods are designed for single contexts and do not address cross-attention issues in multi-context settings. In contrast, SamKV leverages the correlations between multiple contexts, incorporating cross-attention mechanisms to capture dependencies between contexts. When processing a single context, SamKV degrades gracefully to traditional single-context methods, making it a generalization of existing approaches.

2.2 Multi-Context Recomputation

Multi-context methods represent an extension of single-context approaches, further enhancing the efficiency and scalability of LLM serving. Recent works such as CacheBlend (Yao et al. 2024) and EPIC (Hu et al. 2024) have made significant strides in this area. CacheBlend selectively recomputes tokens based on the absence of cross-attention in the first layer, with the scope of updates decreasing progressively across layers. EPIC focuses on updating only the initial and local positions within the context to minimize computational load. In contrast, our proposed method, SamKV, leverages the inherent importance of certain tokens within the context, identified through the concentration of attention weights (as observed in the RaaS (Hu et al. 2025)). By selectively recomputing these critical tokens, in addition to initial and local positions, SamKV reduces both the amount of KV Cache needed and the overall computational requirements, offering a more efficient and targeted approach to LLM serving.

2.3 Attention Sink

In the quest to optimize LLMs, recent studies have uncovered distinct attention patterns. StreamingLLM (Xiao et al. 2024b) first identifies attention concentration at initial positions. Longformer (Beltagy, Peters, and Cohan 2020) emphasizes local contexts, highlighting the importance of nearby tokens. RaaS (Hu et al. 2025) further reveals attention clustering at intermediate positions during reasoning tasks. In contrast, SamKV extends these findings by considering both initial and local tokens and refining the threshold-based selection from RaaS. Using power-law distribution fitting and comparison, SamKV more effectively identifies critical tokens, offering a more rational and comprehensive approach to token selection for enhanced efficiency and performance.

Notation	Description
N	total number of layers
N^*	stable attention layers (see Appendix A (Cao et al. 2025))
D	total number of documents
$KV_{\text{doc-i}}$	document i's KV Cache (also for Q and K Caches)
doc-i$^{\text{old}}$	$KV_{\text{doc-i}^{\text{old}}}$ represents the $KV_{\text{doc-i}}$ before recomputation (also for new), defaulting to old
doc-i$_{\text{ini}}$	initial position of document i's absolute position (also for middle, local, or attention degree positions anchor, max, and min) (attention analysis is in Appendix A)
Q_{que}	generic query vector
$\hat{Q}_{\text{doc-i}}$	query vector for document i only
$P_{\text{doc-i}}$	select Top P in document i
$\langle \cdot, \cdot \rangle$	inner product
$KV^{(n)}$	n-th layer of KV (also for percentages)
$s^{(n)}_{\text{doc-i}_{\text{anc}}}$	$\left\langle Q^{(n)}_{\text{doc-i}}, K^{(n)}_{\text{doc-i}_{\text{anc}}} \right\rangle$ (also for max and min)

Table 2: Summary of terminology.

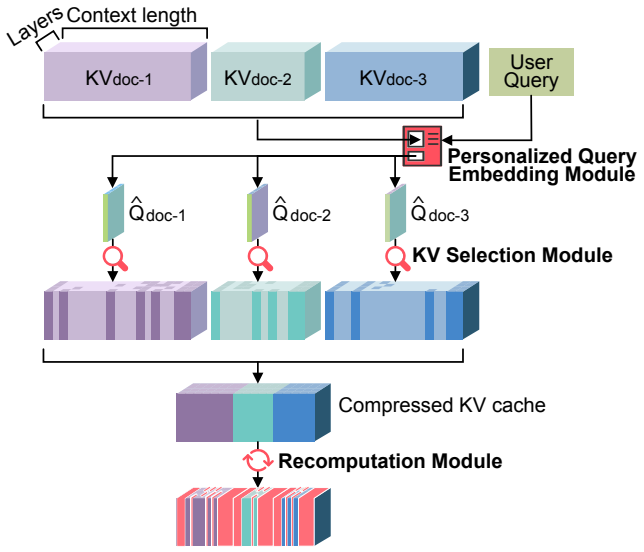

Figure 2: Overview of SamKV: input user query text; generate exclusive personalized query vector for each context; get compressed KV Cache; recompute partial tokens.

3 Our Proposed SamKV

When handling multiple contextual scenarios, two critical challenges emerge: efficiently sparsifying the KV Caches across contexts, and addressing the cross-attention deficiency caused by independent prefilling of each context. To resolve these issues, we propose SamKV, a novel method designed to jointly optimize sparse KV Cache management and cross-context attention recovery. SamKV comprises three core components: (i) Personalized Query Embedding Module, (ii) KV Selection Module, and (iii) Recomputation Module. To elucidate the interplay between these modules, Figure 2 presents a running example with three distinct contexts.

3.1 Generation of Personalized Query Vectors

Existing approaches for multi-context KV Cache necessitate full cache loading (Yao et al. 2024; Hu et al. 2024), which incurs substantial GPU memory overhead. The development of sparse KV Cache techniques provides an effective alternative, where constructing appropriate query vectors represents the crucial initial phase in implementing such sparse solutions. Unlike traditional approaches that employ incremental prefill for user queries, our method introduces document-specific query vector customization, enabling more efficient context-aware processing.

Specifically, multiple contexts and their corresponding KV Caches can be firstly retrieved using the user query in the RAG scenario. Since these contexts are all relevant to the query, they often overlap with each other to some extend. We refer to this overlap as **inter-document consensus**. The consensus, due to its repeated appearance across multiple documents, indicates its particular significance. However, merely using the user query for sparsifying a context is not sufficient to effectively identify the consensus. Therefore, when sparsifying a context, we also need to consider other contexts as query information to recognize the consensus. This results in varying query vectors for each context.

To this end, we propose the Personalized Query Embedding Module, which first generates a generic query vector based on the user query information and then appends consensus information specific to the current context. The details are as follows:

Generate generic query vector. Inspired by the observation that KV Caches at both initial and local positions play a critical role in attention computation (Xiao et al. 2024b; Beltagy, Peters, and Cohan 2020), we propose a novel cache compress strategy. Specifically, we extract and concatenate the KV pairs from these two pivotal positions across all contexts to form a composite Cache unit. This compressed Cache is then employed to incrementally prefill the Q Matrix of user queries through attention computation. Here, the Q Matrix encapsulates the current contextual query representation, which is subsequently transformed into a sparse query vector Q_{que} via mean pooling. As illustrated in Figure 3, the $Q^{(n)}_{\text{que}}$ denotes the Q_{que} of the n-th layer.

Add personalized bias. Although Q_{que} is generated from context, its primary role is to express the user query, which limits its ability to capture targeted consensus. To address this limitation, we propose augmenting Q_{que} with a contextual Q Cache. As before, we focus on Q Caches at the initial and local positions. The initial Q Cache is designed to interact with initial-position K, while the local Q Cache multiplies with all K to complete the information extraction. Consequently, the local Q Cache exhibits the strongest retrieval capability, and we denote document i's local Q Cache as $Q_{\text{doc-i}_{\text{loc}}}$. The next step is determining the proportion and method of integrating $Q_{\text{doc-i}_{\text{loc}}}$ into Q_{que}. To avoid overshadowing the user's original query, the bias introduced by $Q_{\text{doc-i}_{\text{loc}}}$ must be weighted lightly. Empirical analysis reveals that the cosine similarity between Q_{que} and $Q_{\text{doc-i}_{\text{loc}}}$ typically ranges from -0.3 to 0.3. Critically, only when $Q_{\text{doc-i}_{\text{loc}}}$ is in-

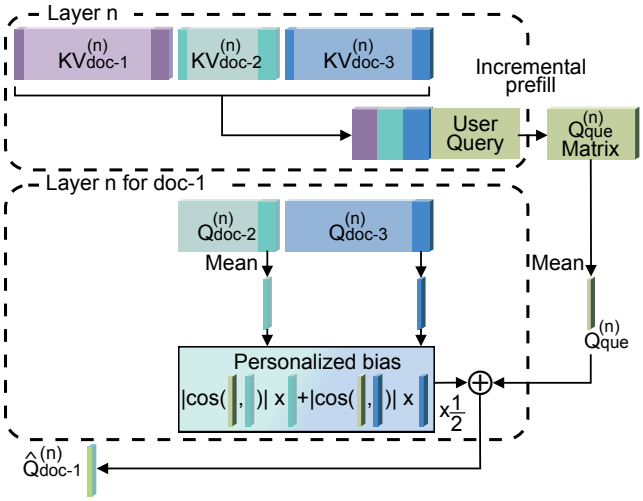

Figure 3: Detail of Personalized Query Embedding Module. The upper half depicts incremental prefill generating the generic query vector Q_{que}, while the lower half visualizes the integration of personalized bias (derived from $Q_{\text{doc-2}_{\text{loc}}}$ and $Q_{\text{doc-3}_{\text{loc}}}$) into Q_{que} to form the document-specific $\hat{Q}_{\text{doc-1}}$.

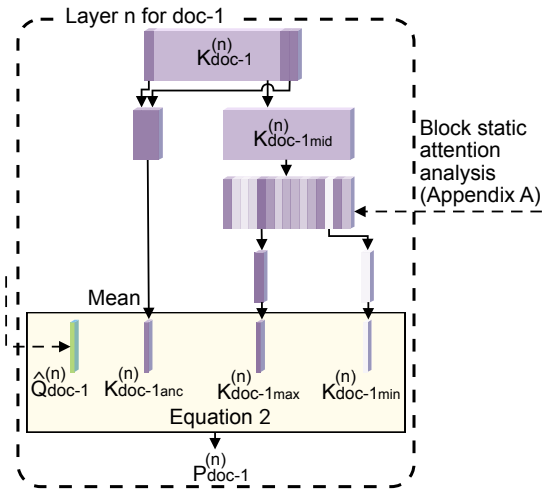

Figure 4: Calculate the Top P to be taken for context 1 in layer n.

corporated into Q_{que} with positive weighting does the inner product of Q_{que} and any K vector effectively retain the multiplicative interaction between $Q_{\text{doc-i}_{\text{loc}}}$ and K. We therefore adopt the absolute cosine value $|\cos|$ as weight constraint. To further safeguard against query information dilution from excessive context, we normalize the integration by the number of contextual documents. The equation composition is expressed as:

$$\hat{Q}_{\text{doc-i}} = Q_{\text{que}} + \frac{1}{D-1} \times \sum_{\text{j}(\text{j}\neq\text{i})}^{D} |\cos(Q_{\text{que}}, Q_{\text{doc-j}_{\text{loc}}})| \times Q_{\text{doc-j}_{\text{loc}}} \tag{1}$$

This equation fundamentally ensures that when sparsifying a document, its query vector assimilates relevant query information from other contexts to facilitate consensus recognition. As illustrated in Figure 3, $\hat{Q}_{\text{doc-1}}$ for context 1 is synthesized through Equation 1 by combining Q_{que} with both $Q_{\text{doc-2}_{\text{loc}}}$ and $Q_{\text{doc-3}_{\text{loc}}}$ according to the specified fusion scheme. The vectors $\hat{Q}_{\text{doc-2}}$ and $\hat{Q}_{\text{doc-3}}$ are derived analogously, following the same augmentation process.

3.2 Selection of Important KV Caches

Current multi-context methods require loading the entire KV Caches during inference, leading to excessive GPU memory consumption. To sparsify the multi-context KV Caches, beyond the query vector generation in Section 3.1, another critical challenge is determining how many KV Caches to select. In our proposed SamKV, we address this by introducing a dynamic Top-P sampling strategy based on anchor points: the proportion P scales adaptively with the number of K Caches more important than the anchor.

Specifically, since KV Caches at initial and local positions are widely recognized as critical for inference (Xiao et al. 2024b; Beltagy, Peters, and Cohan 2020), we retain them in full resolution without sparsification. The primary target of sparsification is instead the middle segments of the KV Caches. Given the importance of initial and local KV Caches, we leverage their corresponding K Caches as anchor points $K_{\text{doc-i}_{\text{anc}}}$ for attention pattern analysis. For the middle blocks, we identify the blocks with maximum and minimum attention scores, denoted as $K_{\text{doc-i}_{\text{max}}}$ and $K_{\text{doc-i}_{\text{min}}}$ respectively (see Appendix A). It is important to note that some tokens (e.g., punctuation marks) may yield high attention scores while being semantically insignificant. To address this issue, we implement block-level KV management and retrieval, where each block is represented by the mean vector of its constituent token caches.

Furthermore, we compute $K_{\text{doc-i}_{\text{anc}}}$, $K_{\text{doc-i}_{\text{max}}}$, and $K_{\text{doc-i}_{\text{min}}}$ inner products with $\hat{Q}_{\text{doc-i}}$ to obtain $s_{\text{doc-i}_{\text{anc}}}$, $s_{\text{doc-i}_{\text{max}}}$, and $s_{\text{doc-i}_{\text{min}}}$, respectively. Under the assumption that $s_{\text{doc-i}_{\text{anc}}}$ and $s_{\text{doc-i}_{\text{min}}}$ remain stable, a larger $s_{\text{doc-i}_{\text{max}}}$ (i.e., a higher upper bound of attention scores) implies more K Caches surpassing the anchor's importance, thus $P_{\text{doc-i}}$ should increase proportionally with $s_{\text{doc-i}_{\text{max}}}$. By similar reasoning, $P_{\text{doc-i}}$ is inversely proportional to both $s_{\text{doc-i}_{\text{min}}}$ and $s_{\text{doc-i}_{\text{anc}}}$. This principled analysis leads to our formulation in Equation 2, i.e.,

$$P^{(n)}_{\text{doc-i}} = \begin{cases} \dfrac{s^{(n)}_{\text{doc-i}_{\text{max}}} - s^{(n)}_{\text{doc-i}_{\text{anc}}}}{s^{(n)}_{\text{doc-i}_{\text{max}}} - s^{(n)}_{\text{doc-i}_{\text{min}}}}, \\ \qquad \text{if } s^{(n)}_{\text{doc-i}_{\text{anc}}} \in \left(s^{(n)}_{\text{doc-i}_{\text{min}}}, s^{(n)}_{\text{doc-i}_{\text{max}}}\right], \\ 0, \quad \text{otherwise.} \end{cases} \tag{2}$$

As formulated in Equation 2, only those K Caches that surpass the anchor point in importance are selected as significant features. While Equation 2 determines the selection ratio $P^{(n)}_{\text{doc-i}}$ specifically for layer n, the final ratio $P_{\text{doc-i}}$ is obtained by consolidating layer-wise ratios through Equa-

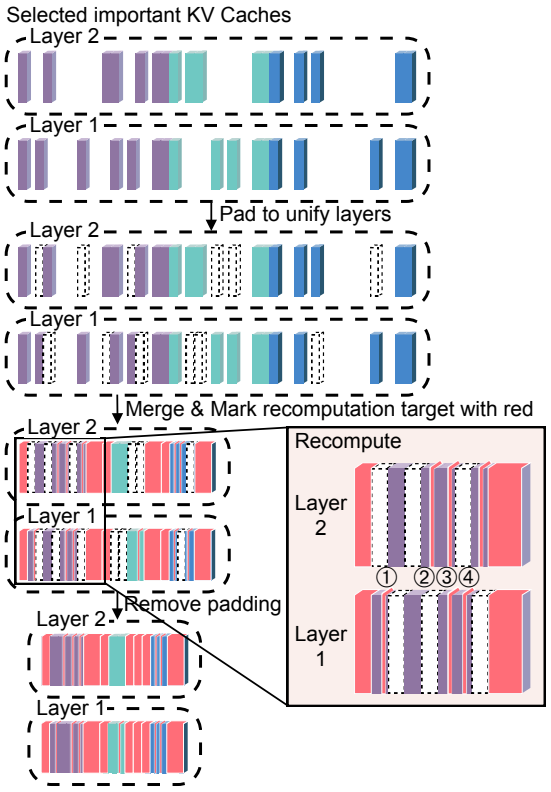

Figure 5: Case of the recomputation for layers 1 and 2. Four token recomputation examples are labels: ① lower layer recomputed but upper layer not, ② neither layer recomputed, ③ upper layer recomputed but lower layer not, and ④ both layers recomputed.

tion 3, i.e.,

$$P_{\text{doc-i}} = \frac{1}{N^*} \sum_{n}^{N^*} P_{\text{doc-i}}^{(n)}. \tag{3}$$

In Equation 3, we selectively employ attention layers that exhibit stable token-wise attention scores, denoted as N^* (see Appendix A).

After selecting the Top $P_{\text{doc-i}}$ KV Cache blocks for document i, we normalize the inner products of all retrieved blocks across contexts before concatenating and comparing them. From this combined set, we retain only the most critical blocks, with the number of selected blocks equal to the total block count divided by the number of contexts. This ensures cross-contextual filtering of the most important blocks.

3.3 Recomputation

Despite obtaining the sparse KV Cache in Section 3.2, the lack of cross-attention between KV Caches from different contexts, which are prefilled separately rather than jointly, leads to performance degradation (Yao et al. 2024). To mitigate this, we need to recompute some tokens in the sparse KV Caches. In addition to tokens with high attention scores from the initial and local positions, we further incorporate tokens that exhibited significant attention weights in the original context for recomputation (see Appendix A for implementation details). However, **sparse KV Caches from multiple contexts cannot be aligned across layers**, posing a significant challenge and explaining why other works avoid sparse KV Caches for multi-context settings. As shown in Figure 5, there are many misaligned blocks between layer 1 and layer 2 in the selected KV Caches.

To address the challenge of cross-layer block alignment, we first align mismatched positions using blank blocks, then apply the following recomputation rules:

1. For any token requiring recomputation at layer n, compute outputs from all preceding $n-1$ layers.
2. At layer n, recompute tokens when necessary while reusing existing Cache entries otherwise.

Following recomputation, remove all padding blocks. Figure 5 illustrates this complete workflow: padding, merging, recomputing, and finally removing padding elements.

Examining the four examples in Figure 5 demonstrates strict adherence to the two rules. Example ① involves recomputing only layer-1 tokens' KV pairs without output computation since layer-2 remains unchanged. Example ② requires no computation under the rule 1. In ③ and ④, layer-2 tokens require recomputation, necessitating computation of corresponding layer-1 token outputs. Rule 2 employs an optimization strategy to maximize reuse of existing KV Cache entries, for instance, in example ③, while layer-2 requires recomputation, layer-1 padding blocks utilize the token's prefilled KV Cache for output computation. This approach effectively reduces computational overhead while compensating for missing cross-document attention.

After recomputing new KV values, conventional approaches directly overwrite the old KV Cache with the new values. In our work, we propose two update strategies:

- **Overwrite**: follow the traditional update mechanism by replacing the old Cache with new KV values;
- **Fusion**: combine new KV values with the old Cache.

While the overwrite mechanism is straightforward and requires no further elaboration, the fusion strategy warrants detailed discussion.

The key to KV fusion lies in determining the appropriate blending ratio between new KV values ($KV_{\text{doc-i}^{\text{new}}}$) and the old Cache ($KV_{\text{doc-i}^{\text{old}}}$), with their weights summing to 1. We employ a simple yet effective approach using the cosine similarity $\theta = \cos(KV_{\text{doc-i}^{\text{new}}}, KV_{\text{doc-i}^{\text{old}}})$. Experimental observations reveal that this similarity measure θ typically yields high values around 0.9. To balance cache updates with preservation of historical information, we formulate the update rule as:

$$KV_{\text{doc-i}^{\text{new}}} = \theta \times KV_{\text{doc-i}^{\text{new}}} + (1-\theta) \times KV_{\text{doc-i}^{\text{old}}}. \tag{4}$$

This equation guarantees both the update of inter-document relationships and the retention of intra-document information, achieving effective knowledge fusion.

After concatenating all $KV_{\text{doc-i}^{\text{new}}}$ yields $KV_{\text{docs}^{\text{new}}}$, answer inference can proceed. However, both Q_{que} and $\hat{Q}_{\text{doc-i}}$ are generated or extended based solely on the initial and local

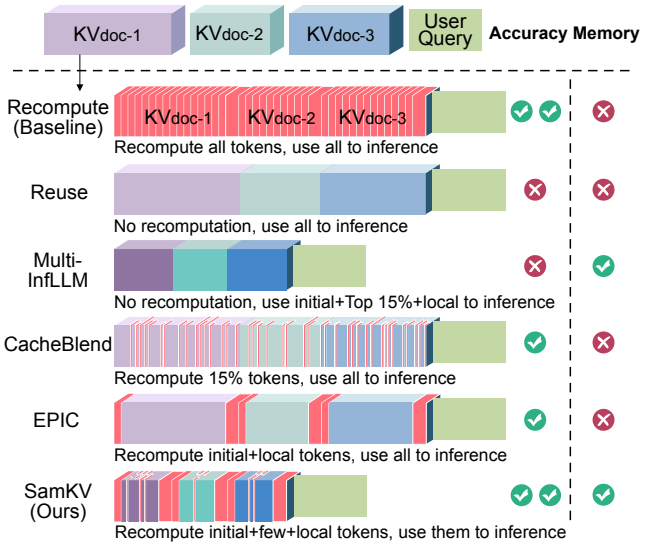

Figure 6: Comparison among multi-context methods. Darker colors indicate KV Caches obtained through sparsification. Red markings denote recomputed positions.

parts of the KV Caches, lacking information from the middle section. Thus, neither is suitable for answer generation. In contrast, $KV_{\text{docs}^{\text{new}}}$ encompasses information from initial, middle, and local sections. Therefore, we re-perform an incremental prefill of the user query based on $KV_{\text{docs}^{\text{new}}}$ and then infer the answer.

4 Evaluation

4.1 Setup

Our proposed SamKV is fundamentally designed as a sparse processing strategy for multi-context scenarios. Given this intrinsic characteristic, our experimental setup including the dataset selection, base LLM configuration, and baseline comparison aligns with established practices in multi-context research. Specifically, we adopt the evaluation framework used by existing multi-context methods such as CacheBlend (Yao et al. 2024) and EPIC (Hu et al. 2024) to ensure fair and meaningful comparisons.

Implementation. For inference hyperparameters, we employ greedy decoding with temperature set to 0. Regarding KV Cache management at the block level, we configure SamKV with a block size of 64, allocating 1 block for the initial position KV Cache and 2 blocks for the local position KV Cache during sparsification.

Datasets. For fair comparison with existing methods, we evaluate performance on three multi-context QA datasets from LongBench (Bai et al. 2024): 2WikiMQA, MuSiQue, and HotpotQA. In addition to these datasets, we conduct further ablation experiments on DuReader. Each dataset contains 200 test samples, with each sample comprising a long context, user query, and corresponding answer. All answers are uniformly evaluated using F1 scores.

Models. In our experiments, we select several representative models including Mistral 7B Instruct (Jiang et al. 2023) and Llama3.1 8B Instruct (Dubey et al. 2024) based on current hardware compatibility. To provide comprehensive evaluation, we incorporate the Qwen2.5 3B Instruct (Team 2024) to examine performance across different model scales.

Method	2WikiMQA	MuSiQue	HotpotQA
Mistral 7B Instruct			
Recompute	25.06	19.94	37.70
Reuse	6.33$_{-18.73}$	1.92$_{-18.02}$	2.32$_{-35.38}$
Multi-InfLLM	23.47$_{-1.59}$	15.77$_{-4.17}$	38.61$_{+0.91}$
CacheBlend	20.42$_{-4.64}$	17.44$_{-2.5}$	33.98$_{-3.75}$
EPIC	19.40$_{-5.66}$	17.22$_{-2.72}$	32.23$_{-5.47}$
SamKV-overwrite	27.04$_{+1.98}$	17.20$_{-2.74}$	**39.10**$_{+1.4}$
SamKV-fusion	**27.88**$_{+2.82}$	**18.54**$_{-1.4}$	37.66$_{-0.04}$
Llama 3.1 8B Instruct			
Recompute	21.48	14.56	24.12
Reuse	3.66$_{-17.82}$	2.23$_{-12.33}$	2.80$_{-21.32}$
Multi-InfLLM	16.20$_{-5.28}$	12.70$_{-1.86}$	29.04$_{+4.92}$
CacheBlend	19.12$_{-2.36}$	12.33$_{-2.23}$	22.18$_{-1.94}$
EPIC	17.36$_{-4.12}$	13.01$_{-1.55}$	23.04$_{-1.08}$
SamKV-overwrite	**23.13**$_{+1.65}$	**13.04**$_{-1.52}$	**35.27**$_{+11.15}$
SamKV-fusion	22.41$_{+0.93}$	12.47$_{-2.09}$	30.52$_{+6.4}$

Table 3: F1 scores of cross-context KV Cache methods.

Methods. We compare our SamKV with five alternative approaches: Recompute, Reuse, Multi-context InfLLM (Multi-InfLLM), CacheBlend (Yao et al. 2024), and EPIC (Hu et al. 2024). Among these, Recompute serves as our baseline by performing complete recalculation of the KV Caches. Multi-InfLLM concatenates KV Caches from all contexts into a single context Cache and applies InfLLM's single-context sparsification method (Xiao et al. 2024a).

Ablation studies. The ablation study examines three key aspects: selection of KV Caches, personalized bias, and recomputation. For the selection component, the sparse KV Caches with selection incorporate the initial-position KV Cache, selected important KV Caches, and local-position KV Caches. In contrast, the no-selection variant only utilizes the initial and local KV Caches without selection.

Environment. We conduct experiments on a server equipped with a 64GB NPU and a 256-core Kunpeng 920 CPU@2.60GHz (hyperthreading disabled), supported by 2TB DRAM. The system runs EulerOS 2.0 with kernel version 5.10.0.

4.2 Overall Improvement

From Table 3, it is evident that SamKV outperforms all other multi-context methods and even surpasses Recompute, the baseline, on both 2WikiMQA and HotpotQA. Even better, when based on Llama, SamKV-overwrite achieves an F1 score of 35.27 on HotpotQA, significantly surpassing Recompute's 24.12, which we attribute to the inclusion of excessive irrelevant information in full recomputation that interferes with result generation. Furthermore, the performance of overwrite and fusion strategies varies depending on the model and dataset, with each showing different advantages. Overall, both recomputation methods contribute to improved effectiveness.

#	Method	Condition			Dataset				Avg.
		Selection	PersBias.	Recompute	2WikiMQA	MuSiQue	HotpotQA	DuReader	
	Qwen2.5 3B Instruct								
1	Recompute	-	-	-	33.34	20.96	40.11	16.49	27.73
2	**SamKV**	✗	-	✗	21.86	13.18	31.73	16.30	20.77
3		✗	-	✓	34.01	16.94	36.51	17.70	26.29
4		✓	✗	✗	22.29	10.75	26.22	15.00	18.57
5		✓	✓	✗	23.64	12.76	28.52	14.90	19.96
6		✓	✗	✓	33.10	13.18	38.57	17.82	25.67
7		✓	✓	✓	**35.00**$_{+1.66}$	**17.03**$_{-3.93}$	**39.09**$_{-1.02}$	**18.08**$_{+1.59}$	**27.30**$_{-0.43}$
	Llama 3.1 8B Instruct								
8	Recompute	-	-	-	21.48	14.56	24.12	27.60	21.94
9	**SamKV**	✗	-	✗	10.54	9.49	21.68	16.99	14.68
10		✗	-	✓	22.12	12.44	26.67	25.00	21.56
11		✓	✗	✗	10.63	9.59	23.62	16.14	15.00
12		✓	✓	✗	10.91	9.36	23.60	17.74	15.40
13		✓	✗	✓	20.90	**13.21**$_{-1.35}$	29.26	25.48	22.21
14		✓	✓	✓	**22.41**$_{+0.93}$	12.47	**30.52**$_{+6.4}$	**25.58**$_{-2.02}$	**22.75**$_{+0.81}$

Table 4: Using full recomputation as the baseline, ablations are conducted on three aspects: selecting the KV Cache in the middle section, adding personalized bias (PersBias.), and whether to recompute (with fusion as default). The evaluation metric is the F1 score, with Avg. denoting the average F1 score across datasets.

4.3 Ablation Studies

We conduct comprehensive ablation experiments to evaluate three key design choices: (i) whether to apply selection to the middle segment's KV Cache, (ii) the inclusion of personalized bias, and (iii) the use of recomputation. Following the comparative analysis of overwrite versus fusion recomputation strategies in Section 4.2, we fix the recomputation method to fusion for all ablation trials to isolate variable effects. The detailed analysis is presented as follows.

Selection of middle KV Caches. The key consideration in adopting selection lies in whether to extract crucial KV from the middle part of the KV Cache. Here, ✓ denotes selection, while ✗ indicates no selection, meaning only the initial and local parts of the KV Caches are used. Comparing the average results (Avg.) in Table 4, specifically rows 2 and 4 as well as rows 9 and 11, it can be observed that when recomputation is disabled, the absence of the middle KV Cache yields slightly better performance than its inclusion. This suggests that relying solely on the initial and local parts of the KV Cache can achieve reasonably good results. However, this performance improvement precisely becomes the factor that limits its further enhancement. Only when extending to incorporate the middle part can additional personalized bias be introduced. For instance, when recomputation is enabled, comparing the cases without selection (rows 3 and 10) and those with personalized bias (rows 7 and 14) reveals that selection with personalized bias delivers superior performance. Evidently, adopting selection is a prerequisite for further enhancement.

Personalized bias. Comparing rows 4 and 5, rows 6 and 7, rows 11 and 12, and rows 13 and 14 in Table 4 reveals that while both cases involve selection, the inclusion of personalized bias in the query vectors yields better performance. The results show that introducing a small amount of Q Cache into the query vector may help identify consensus for better performance.

Recomputation. Comparing rows 2 and 3, rows 4 and 6, rows 5 and 7, rows 9 and 10, rows 11 and 13, and rows 12 and 14 in Table 4, it can be observed that recomputing only a portion of the sparsified KV Cache consistently improves the F1 score by 6% to 7%, regardless of whether selection is applied or personalized bias is incorporated. This indicates that recomputing only the sparse subset of tokens can significantly alleviate the missing cross-attention between documents without requiring the full KV Cache to be carried throughout the entire inference process.

In terms of time consumption, compared to methods like CacheBlend and EPIC that utilize full KV Caches, our approach introduces an additional sparsification process which incurs extra time overhead. However, these methods require carrying the complete KV Caches during recomputation, resulting in high time complexity. In contrast, our method only recomputes a small subset of tokens from the sparsified KV Caches, and the sparsification process is accelerated by vector databases, leading to lower overall time consumption.

5 Conclusion

In this paper, we present SamKV, the first method designed for KV Cache sparsification across diverse contexts. Experimental results demonstrate that SamKV achieves comparable accuracy to full recomputation while reducing the required KV Cache during inference to only 15% of its original length. Further investigations reveal that incorporating inter-document consensus during sparsification along with localized recomputation after sparsification not only reduces computational overhead and the amount of KV Cache needed for inference but also significantly mitigates cross-attention deficiencies between documents. These findings provide valuable insights for optimizing ultra-long multiple contexts processing and establish an important reference for future research in this domain.

Acknowledgments

The research in this article is supported by the National Natural Science Foundation of China (62176074).

References

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. 2024. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. In Ku, L.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, 3119–3137. Association for Computational Linguistics.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The Long-Document Transformer. *CoRR*, abs/2004.05150.

Cao, Z.; Si, Q.; Zhang, J.; and Liu, B. 2025. Sparse Attention across Multiple-context KV Cache. arXiv:2508.11661.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 4171–4186. Association for Computational Linguistics.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; Goyal, A.; Hartshorn, A.; Yang, A.; Mitra, A.; Sravankumar, A.; Korenev, A.; Hinsvark, A.; Rao, A.; Zhang, A.; Rodriguez, A.; Gregerson, A.; Spataru, A.; Rozière, B.; Biron, B.; Tang, B.; Chern, B.; Caucheteux, C.; Nayak, C.; Bi, C.; Marra, C.; McConnell, C.; Keller, C.; Touret, C.; Wu, C.; Wong, C.; Ferrer, C. C.; Nikolaidis, C.; Allonsius, D.; Song, D.; Pintz, D.; Livshits, D.; Esiobu, D.; Choudhary, D.; Mahajan, D.; Garcia-Olano, D.; Perino, D.; Hupkes, D.; Lakomkin, E.; AlBadawy, E.; Lobanova, E.; Dinan, E.; Smith, E. M.; Radenovic, F.; Zhang, F.; Synnaeve, G.; Lee, G.; Anderson, G. L.; Nail, G.; Mialon, G.; Pang, G.; Cucurell, G.; Nguyen, H.; Korevaar, H.; Xu, H.; Touvron, H.; Zarov, I.; Ibarra, I. A.; Kloumann, I. M.; Misra, I.; Evtimov, I.; Copet, J.; Lee, J.; Geffert, J.; Vranes, J.; Park, J.; Mahadeokar, J.; Shah, J.; van der Linde, J.; Billock, J.; Hong, J.; Lee, J.; Fu, J.; Chi, J.; Huang, J.; Liu, J.; Wang, J.; Yu, J.; Bitton, J.; Spisak, J.; Park, J.; Rocca, J.; Johnstun, J.; Saxe, J.; Jia, J.; Alwala, K. V.; Upasani, K.; Plawiak, K.; Li, K.; Heafield, K.; Stone, K.; and et al. 2024. The Llama 3 Herd of Models. *CoRR*, abs/2407.21783.

Hu, J.; Huang, W.; Wang, H.; Wang, W.; Hu, T.; Zhang, Q.; Feng, H.; Chen, X.; Shan, Y.; and Xie, T. 2024. EPIC: Efficient Position-Independent Context Caching for Serving Large Language Models. *CoRR*, abs/2410.15332.

Hu, J.; Huang, W.; Wang, W.; Li, Z.; Hu, T.; Liu, Z.; Chen, X.; Xie, T.; and Shan, Y. 2025. RaaS: Reasoning-Aware Attention Sparsity for Efficient LLM Reasoning. *arXiv preprint arXiv:2502.11147*.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de Las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. *CoRR*, abs/2310.06825.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

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. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Liu, D.; Chen, M.; Lu, B.; Jiang, H.; Han, Z.; Zhang, Q.; Chen, Q.; Zhang, C.; Ding, B.; Zhang, K.; Chen, C.; Yang, F.; Yang, Y.; and Qiu, L. 2024. RetrievalAttention: Accelerating Long-Context LLM Inference via Vector Retrieval. *CoRR*, abs/2409.10516.

Pope, R.; Douglas, S.; Chowdhery, A.; Devlin, J.; Bradbury, J.; Heek, J.; Xiao, K.; Agrawal, S.; and Dean, J. 2023. Efficiently Scaling Transformer Inference. In Song, D.; Carbin, M.; and Chen, T., eds., *Proceedings of the Sixth Conference on Machine Learning and Systems, MLSys 2023, Miami, FL, USA, June 4-8, 2023*. mlsys.org.

Team, Q. 2024. Qwen2.5: A Party of Foundation Models.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 5998–6008.

Xiao, C.; Zhang, P.; Han, X.; Xiao, G.; Lin, Y.; Zhang, Z.; Liu, Z.; and Sun, M. 2024a. InfLLM: Training-Free Long-Context Extrapolation for LLMs with an Efficient Context Memory. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Xiao, G.; Tian, Y.; Chen, B.; Han, S.; and Lewis, M. 2024b. Efficient Streaming Language Models with Attention Sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yao, J.; Li, H.; Liu, Y.; Ray, S.; Cheng, Y.; Zhang, Q.; Du, K.; Lu, S.; and Jiang, J. 2024. CacheBlend: Fast Large Lan-

guage Model Serving for RAG with Cached Knowledge Fusion. *CoRR*, abs/2405.16444.