Enhancing Long-Context Inference with Context-Position Duo-Mixture

Zhenyu Zhang[1]*, Sharath Nittur Sridhar[2], Zhangyang Wang[1], Souvik Kundu[2]
[1]University of Texas at Austin, [2]Intel Labs

Long-context understanding ability of the existing large language models (LLMs) is generally limited by their pre-training context window, providing limited effectiveness as context length increases. Moreover, even within the range of pre-training context length, LLMs often fail to capture vital information present in the middle of the context-window. Towards mitigating these limitations, we introduce ***context-position duo-mixture*** (CoPMix) of LLMs, a simple yet effective **training-free** method designed to enhance their long-context understanding performance in terms of both effectiveness as well as context awareness. Specifically, we present an input *context chunking and mixing* strategy that divides long sequences into multiple chunks, each accompanied by a shared context sink. The input query attends to all chunks in parallel, enabling the efficient integration of information across chunks. We then introduce an adaptive assignment of positional information to enhance the context awareness. This duo-mixture strategy reduces the quadratic complexity of attention to sub-quadratic while improving long-context processing performance. Extensive experiments across multiple LLMs on diverse long-context datasets demonstrate that CoPMix achieves up to a 9.79% accuracy improvement over the existing alternatives, while reducing the pre-filling latency by up to 69.14% compared to full attention LLM alternative.

1. Introduction

Large language models (LLMs) have significantly enhanced the capabilities of machine learning across various tasks, including summarization and comprehension of extensive texts [1], multi-round conversations [2, 3], and code generation [4, 5]. In these real-world applications, the ability to process long-sequence inputs (for example, long-sequence retrieval tasks) has become a critical requirement. However, LLMs face several challenges in processing long-sequences. Firstly, the limited context-size of the pre-training data often makes the trained LLMs perform poorly on longer contexts during inference, due to positional out-of-distribution (OOD) issue [6]. Secondly, even within the limited context window, LLMs are susceptible to positional biases [7], often overlooking the information in the middle. Additionally, the quadratic compute complexity of the LLM self-attention layers with increased sequence length makes their inference with long-context input significantly compute heavy, increasing the prefill latency [8].

Existing works on long-context modeling of pre-trained LLMs primarily rely on the manipulation of positional information [9–13] in RoPE [14] or employment of selective sparse attention [6, 15–17]. However, even with an extended context window, these approaches often lack sufficient context awareness, risking the omission of critical information and result in an effective context length that remains inadequate [18, 19]. To enhance context awareness, Peysakhovich and Lerer [20] proposed splitting inputs into chunks and dynamically reordering them, placing the most critical chunk at the end of the sequence to improve contextual understanding. Other works [21, 22] enhanced context awareness by blending positional information with different RoPE bases. More recently Zhang et al. [23] introduced multi-scale positional rescaling across different heads. However, these methods struggle to handle sequence lengths beyond the pre-trained context windows.

*Work done during internship at Intel.

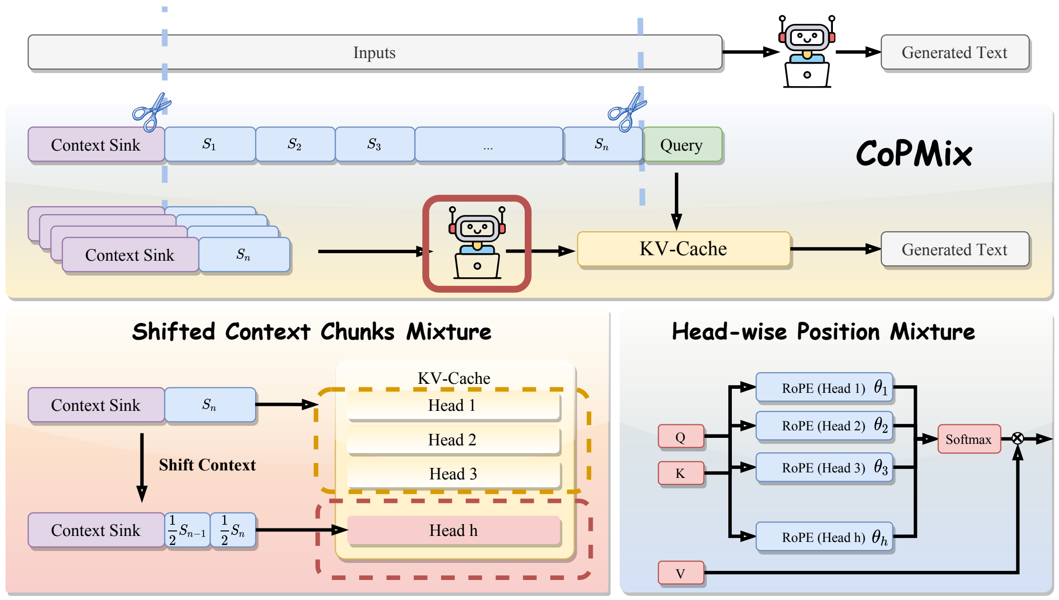

Figure 1: Overall framework of CoPMix . First, the input sample is divided into several context chunks, with each chunk combined with a shared context sink. Each context chunk is then encoded separately, while the KV cache is mixed with shifted context chunks across different attention heads. To enhance context awareness, different RoPE bases θ are assigned to different attention heads.

To improve efficiency of context processing earlier works proposed various compute reduction methods including sparse attention [8] and self-attention alternatives such as gated linear attention [24], and state space models [25]. However, these works require significant hardware and/or runtime optimization support to yield latency benefit and often suffer from accuracy drop [26].

To effectively and efficiently handle long-context inputs, chunk-wise context processing has emerged as a promising solution. However, a naive implementation of this strategy can lead to substantial accuracy degradation due to two main factors: (i) the key-value (KV) states for each text depend on preceding in-context information, which is disrupted by chunk-wise processing; and (ii) incoherent information flow across chunks hinders consistent representation and negatively impacts overall performance.

Towards mitigating the above mentioned limitations, in this paper, we introduce **Co**ntext-**P**osition Duo-**Mix**ture (**CoPMix**), a simple yet effective ***training-free*** method to effectively and efficiently process long-context prefill inputs. The key idea behind CoPMix is a *divide-and-conquer* approach, where inputs are partitioned into multiple chunks, processed in parallel, and then integrated back for subsequent generation. This approach allows for the reuse of positional information, mitigating the positional OOD issue. Additionally, this approach enables sub-quadratic computational complexity for LLMs, as the overall computation scales linearly with the number of chunks. CoPMix addresses the incoherent chunk processing and associated accuracy drop concern via three key innovations, as illustrated in Figure 1: (i) Context Sink: Initial chunk of tokens is selected as a shared context sink for each of the following chunk, significantly enhancing the sink information across chunks. (ii) Shifted Context Chunk Mixture: Chunks are mixed with their shifted counterparts across different attention heads to prevent critical information from being split across chunks. (iii) Positional Mixture: we assign different rotational bases across attention heads, further improving their context awareness when processing each chunk.

We empirically demonstrate that CoPMix enhances the long-context understanding capability of different LLMs, achieving up to 9.79% performance improvements on tasks from the LongBench benchmark [27]. Additionally, CoPMix reduces the latency of the pre-filling stage by 69.14%, while remaining fully compatible with existing KV cache compression techniques [28, 29], enabling further efficiency gains during the decoding process.

2. Related Works

2.1. Long-Context Generalization of LLMs

To extend the context window of LLMs and support longer sequences, advancements broadly fall into two main approaches: (i) Manipulating Position Embeddings, with notable methods such as PI [9], CLEX [11], YaRN [10], Self-Extend [12], and ChunkLlama [13]. These methods effectively extend the context window beyond the pretraining length limitation, achieving multiple times the original capacity, yet not unlimited sequence lengths.(ii) Sparse Attention Mechanisms, like StreamingLLM [15], which introduces the concept of "sink tokens" to enable processing of unlimited sequence lengths. However, this approach may overlook critical contextual information. Alternative methods, such as H2O [30], InfLLM [17], and DuoAttention [16], address this limitation by using heuristic selection or head-wise adaptive strategy.

2.2. Context Aware LLMs

Despite the extended context window, LLMs still struggle with limited context awareness, often favoring information from certain positions. Liu et al. [7] demonstrated that LLMs are prone to overlooking middle-context information, which may stem from the side-effect of causal attention mechanisms and rotary position encoding [21, 23, 31]. To tackle this issue, Peysakhovich and Lerer [20] proposed attention sorting, which reorders inputs by placing critical information at the end. Chen et al. [21] employed Attention Buckets for aggregating outputs from multiple forward passes, while Lin et al. [22] developed a lightweight router that dynamically adjusts positional information for each sample. Additionally, Hsieh et al. [32] addressed the problem by denoising attention scores using calibration samples.

2.3. Efficient Computation for Long-Sequence Inputs

To achieve efficient long-sequence computation, previous methods have primarily focused on sparse attention. For instance, Jiang et al. [8] and Lai et al. [33] dynamically select sparse patterns and perform sparse computations on-the-fly to enable efficient prefilling, while Gao et al. [34] introduces a training-based approach to enhance the performance of sparse attention mechanisms. In addition to these algorithmic advancements, innovative architectural designs—such as YOCO [35], Mamba [25], and gated linear attention [24]—present alternative solutions for handling long contexts more effectively. Beyond model-level improvements, system-level optimizations have also been proposed to boost inference efficiency in long-context scenarios. Techniques such as vLLM [36], FlashAttention [37, 38], and RingAttention [39] optimize batch processing and leverage sequence parallelism, resulting in substantial gains in runtime efficiency.

Despite extensive prior work on long-context processing, existing approaches typically focus on single challenge—such as extending the context window, enhancing context awareness, or improving computational efficiency—but not all simultaneously. In this work, we aim to develop an effective strategy that addresses all three aspects in a unified manner.

3. Methodology

We now describe the details of CoPMix method. Section 3.1 offers a brief overview of the background knowledge. Sections 3.2 and 3.3 describe the main components of the context and position mixture, respectively. Finally, the complete methodology of CoPMix is outlined in Section 3.4.

3.1. Preliminaries

Self-Attention Mechanism. As the key building block of transformer architecture [40], self-attention operates the interaction among token embeddings. In specific, the self-attention layer output is computed as

$$\mathbf{O} = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}$$

where $\mathbf{Q} \in \mathbb{R}^{l_q \times d}, \mathbf{K} \in \mathbb{R}^{l_k \times d}, \mathbf{V} \in \mathbb{R}^{l_k \times d}, \mathbf{O} \in \mathbb{R}^{l_q \times d}$ represent the query, key, value, and output token embeddings, respectively. Here, l_q and l_k denote the sequence lengths of the query and key states, respectively, while d represents the dimensionality of each state. The $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ tensors are generated via respective linear transformations of the input embedding $\mathbf{X}$ through three separate learned linear transformation blocks.

Positional Encoding. Numerous positional encoding methods have been proposed to capture the relative positional dependency among tokens. These can be broadly categorized as absolute positional encoding [40] and relative positional encoding [14, 41, 42]. In this work, we focus on rotary positional encoding (RoPE [14]), due to its effectiveness in capturing relative dependency and widespread industry adoption. The formulation of RoPE is given by:

$$\begin{aligned} f(\mathbf{H}, m) &= \mathbf{H}e^{im\boldsymbol{\theta}} \\ &= \big[(H_1 + iH_2)e^{im\theta_1}, (H_3 + iH_4)e^{im\theta_2}, \\ &\quad ..., (H_{d-1} + iH_d)e^{im\theta_{d/2}}\big]^\top \end{aligned}$$

where a rotational transformation is applied to the original embedding $\mathbf{H}$ with $\theta_i = \theta_B^{-2i/d}$. θ_B denotes the base angle used for the rotation and H_i is the i^{th} dimension of the representation $\mathbf{H}$. This ensures that the inner product of query and key embeddings incorporates relative positional information. Specifically:

$$f(\mathbf{Q}_m, m)^\top f(\mathbf{K}_n, n) = g(\mathbf{Q}_m, \mathbf{K}_n, m - n)$$

m and n denote the positional indices of the query and key embeddings, respectively.

3.2. Mixture of Context

Since pretraining is generally conducted under a constrained context window due to hardware limitations, LLMs struggle to generate meaningful outputs when processing significantly larger positional indices. This is potentially caused by the positional OOD issue [6, 43], leading to degraded LLM performance in long-context scenarios, or complete performance collapse.

Thus, CoPMix starts with the mixture of context strategy where we can view the original LLMs as a context expert that is capable of handling the input within its effective context length. For longer sequences, the inference process can be structured as a mixture of multiple *small* context chunks. For example, consider an input sequence of length N, which is uniformly divided into multiple chunks, each of size N_C, where $N_C \leq N_P$, with N_P as the pre-training context length. The total number of chunks is then given by $\lceil N/N_C \rceil$. This approach ensures that each chunk remains within the effective context length, mitigating the positional OOD issue. However, directly applying this strategy leads to significant degradation in generative performance, as demonstrated in Figure 2. The primary cause may be attributed to the attention sink phenomenon [15], where the initial tokens play a crucial role in sustaining the generation process. Since LLMs employ causal attention, preceding tokens contribute more significantly to the attention process. Specifically, the i^{th} token can be attended to by the subsequent $S - i$ tokens, where S is the total sequence length. This mechanism leads the model to disproportionately prioritize the initial tokens. When encoding each chunk separately, the absence of shared initial tokens prevents the key-value pairs of different chunks from being directly integrated. To address this issue, we employ context sink [44, 45], where each chunk shares a fraction of initial tokens as the context sink (T), as shown in Figure 1. We consistently use the initial instruction tokens as the context sink.

Context Sink. Let us assume a long input sequence S composed of the main sequence window S_M that is followed by the query observation window S_q, respectively. The total sequence length equals $|S_M| + |S_q|$. We designate the first T tokens as the context sink, denoted as $S_c = S_M[: \mathrm{T}]$.

The remaining sequence of the main window, $S_M[\mathrm{T}:]$, is evenly divided into several chunks. Each chunk is then attached to the context sink, as $S_i = S[:\mathrm{T}] + S[\mathrm{T}+\mathrm{iC}:\mathrm{T}+(\mathrm{i}+1)\mathrm{C}]$, where C represents the chunk size. Each chunk sharing the same positional information in the RoPE to prevent the positional OOD issue. We then encode each context in parallel, storing and reconstructing the intermediate key-value pairs:

$$\{K, V\} = \{K_0[:\mathrm{T}], V_0[:\mathrm{T}]\} \cup \left(\cup_{i=0}^{n} \{K_i[\mathrm{T}:], V_i[\mathrm{T}:]\} \right)$$

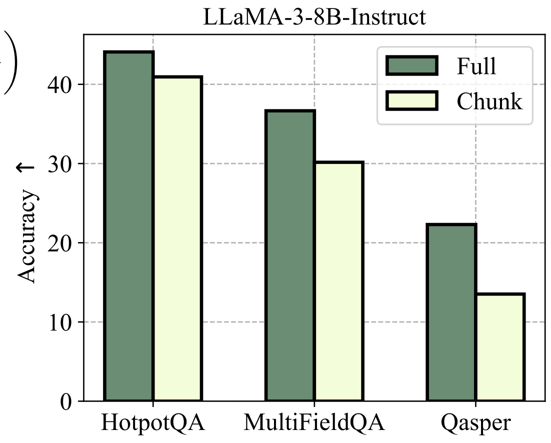

Figure 2: Comparison results of directly applying chunk-wise inference v.s. full attention.

As illustrated in Figure 3, we visualize the attention scores of the final query token with respect to all previous keys. The context sink effectively prevents attention score spikes and facilitates the integration of different context chunks.

Shifted Context Chunks Mixture. Although effective, one limitation of the context mixture strategy is that it might split critical sub-sequences into different chunks. This disrupts the continuity of key information and leads to performance degradation. To investigate this, we conducted preliminary experiments on the multi-document question-answering task [7], where each sample contains ten documents, but only one is relevant to the query. As shown in Figure 4, we divide the inference process into two chunks and manually position the key documents in either the first (1^{st}) or second (2^{nd}) chunk. We then compare performance when the key documents are split across both chunks (`Splitted`). `Full` represents the original inference process. Using the LLaMA-3.2-1B-Instruct and LLaMA-3-8B-Instruct models, we report the average accuracy over 100 test samples. The results indicate a significant performance degradation when key documents are split across different chunks.

To address this issue and better integrate information across different chunks, we propose a shifted context chunks mixture strategy. For each context chunk, we apply a shift to create shifted chunks, defined as $S_i^f = S[:\mathrm{T}] + S[\mathrm{T}+\mathrm{iC}+\mathrm{T}_{\mathrm{offset}} : \mathrm{T}+(\mathrm{i}+1)\mathrm{C}+\mathrm{T}_{\mathrm{offset}}]$. We compute the index $\mathrm{T}+(\mathrm{i}+1)\mathrm{C}$ modulo the sequence length to ensure valid indexing. We then mix the original and shifted context chunks by replacing the KV pairs of p attention heads from the original context chunk with those from the shifted context. Section 4.3 presents a detailed ablation study on selecting the optimal ratio p and offset parameter $\mathrm{T}_{\mathrm{offset}}$. In this way, if critical information is split across two context chunks, its main components can still be preserved in the shifted chunk, enabling effective information encoding for subsequent generation steps.

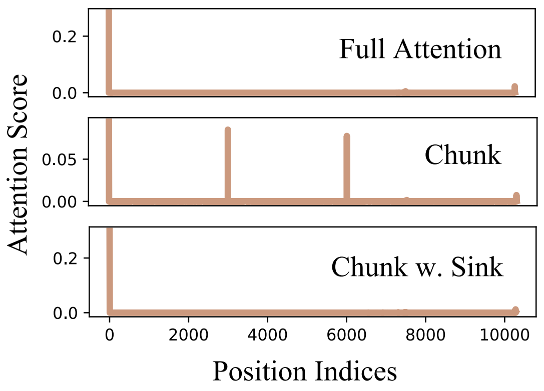

Figure 3: Visualization of attention scores of LLaMA-3-8B-Instruct.

3.3. Mixture of Position

When processing each context chunk, LLMs often struggle with limited context awareness, showing sensitivity to the position of key information [7] and a vulnerability to missing critical details. Previous works [21, 22] suggest that this limitation primarily arises from RoPE. The rotation format in RoPE introduces fluctuations in attention scores, causing LLMs to favor contexts within certain areas.

To address this, we employ a simple yet effective mixture of position strategy, where the base θ of RoPE varies across different attention heads. For the RoPE of attention head i: $f_i(\mathbf{H}, m) = \mathbf{H}e^{im\boldsymbol{\theta}}$ where $\theta_i = \theta_B^{-2i/d}$, we assign θ_B as a linear interpolation between a minimum and maximum base values.

$$\theta_{B,i} = \frac{i}{h}(\theta_{\max} - \theta_{\min}) + \theta_{\min}$$

In our experiments, we set $\theta_{\min} = 10000$ and $\theta_{\max} = 13000$ as default values.

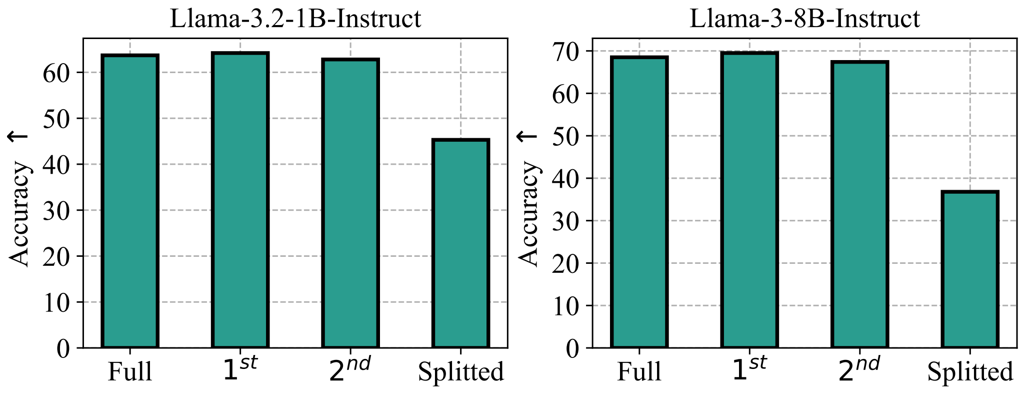

Figure 4: Results on the multi-document question-answering task where key documents are placed at different positions.

3.4. CoPMix Method

The overall framework of CoPMix is illustrated in Figure 1. By leveraging a context-position duo-mixture strategy, CoPMix offers several key advantages: (i) it mitigates positional out-of-distribution (OOD) issues by sharing position indices across different context chunks, (ii) it reduces the pre-filling complexity from quadratic ($O(n^2N_C^2)$) to sub-quadratic ($O(nN_C^2)$) by employing a fixed context chunk size N_C, where n denotes the number of chunks, and (iii) it enhances context awareness by varying the rotation base θ across attention heads. Additionally, CoPMix is fully compatible with KV cache compression techniques [28–30], providing further efficiency during the decoding phase (see Section 4.3, A4).

4. Experiments

4.1. Long-Context Understanding

Setup. We evaluate CoPMix on six tasks from LongBench [27], including MultiFieldQA, HotpotQA, 2WikiMQA, TREC, Qasper, and GovReport. Experiments are conducted using LLaMA-2-7B-Chat [46], Vicuna-7B [47], Mistral-7B-Instruct [48] and LLaMA-3-8B-Instruct [49]. For the full baseline, we adopt the default chunking strategy for samples exceeding the model's context window. This method evicts the middle portion of the input, retaining only the beginning and end segments to ensure the remaining context fits within the allowable window. However, this approach inevitably leads to the loss of critical information in the middle of the sequence. Alternatively, feeding the entire sequence without truncation results in model collapse and zero accuracy due to OOD positional encoding issues. Evaluations are conducted with BF16 precision on H100 GPUs. We consider a contemporary work, StarAttention [45] as the baseline method for comparison. It similarly splits the context into multiple chunks and encodes them in parallel. The original StarAttention implementation uses absolute positional indices across all chunks which is prone to OOD issues in

Table 1: Comparison results for long-context understanding in LongBench dataset.

	MultiFieldQA	HotPotQA	2WikiMQA	TREC	Qasper	GovReport	Avg.
Llama-3-8B-Instruct	36.66	44.09	31.93	72.61	22.30	28.05	39.27
StarAttention	37.45	51.86	32.91	77.07	14.46	27.75	40.25
CoPMix	**38.48**	**54.06**	**32.49**	**78.98**	**24.89**	**28.28**	**42.86**
Llama-2-7B-Chat	33.13	30.57	**27.08**	59.52	20.57	27.25	33.02
StarAttention	16.44	12.93	15.88	32.54	19.89	**28.86**	21.09
CoPMix	**34.71**	**32.59**	25.58	**62.30**	**21.70**	28.08	**34.16**
Vicuna-7B	34.62	20.02	15.70	60.72	19.83	27.82	29.79
StarAttention	30.31	17.74	16.06	44.44	22.21	**31.95**	27.12
CoPMix	**35.33**	**20.46**	**17.02**	**63.89**	**21.96**	28.25	**31.15**
Mistral-7B-Instruct	**46.92**	49.80	33.05	69.41	35.68	32.55	44.57
StarAttention	41.55	53.72	**34.90**	71.18	30.71	**33.10**	44.19
CoPMix	43.01	**53.74**	34.54	**72.35**	**36.93**	31.03	**45.27**

positional encoding. We modify the approach to use relative positions within each chunk, avoiding performance collapse.

Main Results. In Table 1, CoPMix consistently outperforms both Full Attention and StarAttention across majority of the tasks, achieving up to 9.97% improvement in accuracy and an average gain of 0.70% to 3.59%. These results are primarily attributed to the limitations of Full Attention in handling inputs that exceed the pretrained context window, leading to degraded performance on long sequences. In contrast, StarAttention, which resembles the context-sink-only variant of CoPMix, often disrupts continuity by splitting consecutive information across separate chunks. This fragmentation weakens the model's ability to capture dependencies and leads to performance degradation.

4.2. Prefilling Efficiency

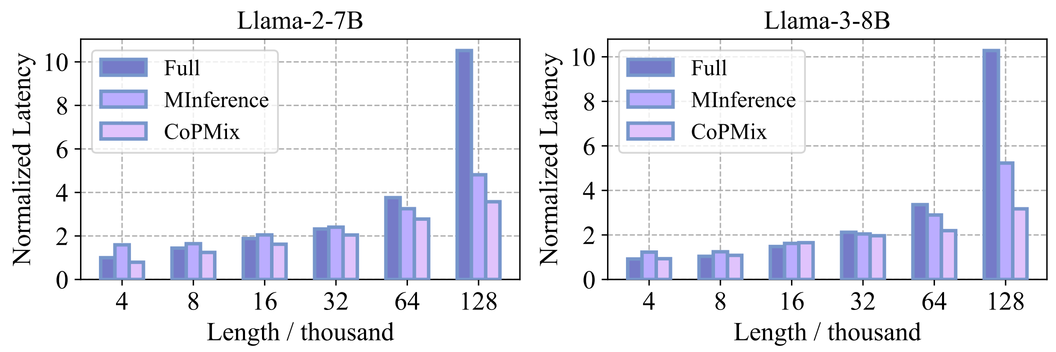

Figure 5: Latency of prefilling across different prompt length on H100 devices.

Next, we evaluate the efficiency of CoPMix across varying sequence lengths. Experiments are conducted on H100 GPU, with sequence lengths ranging from 2K to 128K tokens. The chunk size is fixed at 1024, and the context sink is set to 16. We measure latency during the prefilling stage, and the results are shown in Figure 5. As illustrated, Full Attention incurs significantly higher latency as the sequence length increases. In contrast, CoPMix achieves consistently lower latency compared to both Full Attention and MInference [8]. For Minference, we use the official implementation with the Hugging Face backend and extend it to support LLaMA-2-7B following the instructions. These results further demonstrate the efficiency and scalability of CoPMix for long-context processing.

4.3. Analysis and Ablation Studies

In this section, we further evaluate the effectiveness of CoPMix by addressing the following key questions: *Q1*: How do the different components of CoPMix impact overall performance? *Q2*: How does CoPMix perform when varying the number of attention heads in the shifted context chunk mixture? *Q3*: What's the optimal ratio of shifted context chunk? *Q4*: Is CoPMix compatible with KV cache compression strategies?

Table 2: Ablation study of different components in CoPMix. Experiments are conducted with Llama-2-7B-chat on HotpotQA.

Methods	Context Sink	Shifted Context	Position Mixture	HotPotQA	MultiFieldQA
Full	✗	✗	✗	44.09	36.66
Chunk	✗	✗	✗	40.94	30.16
	✓	✗	✗	51.65	34.75
	✓	✓	✗	52.94	37.51
CoPMix	✓	✓	✓	**54.06**	**38.48**

A1: Ablation study of each component. We begin by examining the individual effects of each component in CoPMix, as presented in Table 2. Several key observations can be made: (i) Context Sink is crucial for stabilizing the encoding of each chunk in parallel. Without it, performance significantly degrades, which is expected since, during the pre-filling stage, KV pairs heavily depend on the initial tokens. When each chunk starts with different initial tokens, the resulting KV pairs are distributed across different representational spaces, making integration challenging. (ii) Shifted context mixture is beneficial as it prevents important information from being split across different chunks, thereby improving coherence and results the accuracy improvement of up to 2.76% (iii) Position mixture further enhances performance, yielding an additional accuracy gain of 1.12% and 0.97%, respectively. This improvement stems from the enhanced context awareness within each chunk. Overall, each component contributes to strengthening the model's ability to handle long contexts, and their benefits can be effectively combined for greater improvements.

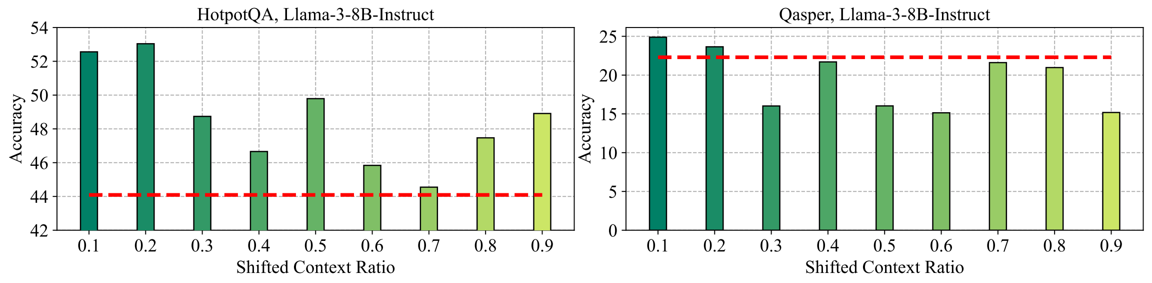

Figure 6: Ablation study of shifted context ratio. Experiments are carried out with LLaMA-3-8B-Instruct on the HotpotQA and Qasper task, respectively. The red line denotes the performance of the original full attention baseline.

A2: Integrate shifted context across 40% of the attention heads. As illustrated in Figure 6 (Right), we sequentially select attention heads and replace their KV pairs with those from the shifted context chunks. The shifted context chunks maintain a fixed shift ratio of 20%, while we progressively increase the number of attention heads incorporating these shifted chunks. Our observations indicate that the optimal mixture ratio is approximately 40%, corresponding to 12 attention heads for LLaMA-3-8B-Instruct.

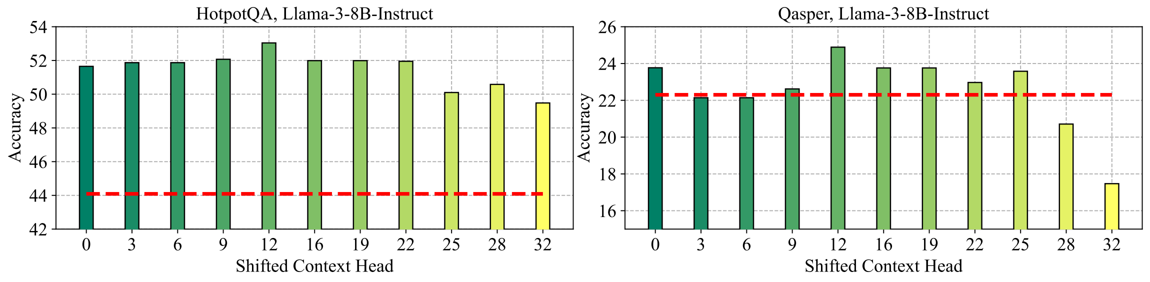

Figure 7: Ablation study of shifted context heads. Experiments are carried out with LLaMA-3-8B-Instruct on the HotpotQA and Qasper task, respectively. The red line denotes the performance of the original full attention baseline.

Table 3: Comparison of CoPMix with existing KV cache compression techniques. All experiments are conducted on LLaMA-3-8B-Instruct.

Methods	HotPotQA	MultiFieldQA
Full	44.09	36.66
CoPMix	**54.06**	38.48
w. KIVI [28]	53.58	37.40
w. SnapKV [29]	53.68	**39.59**

A3: Shifting the context by 10% to 20% yields the optimal results. Furthermore, while keeping the number of attention heads with shifted context chunks fixed, we then investigate the impact of varying the shifted ratios. Specifically, for each context chunk of size C, we remove the first $r * C$ tokens and append the same number of tokens from the subsequent chunk, where r represents the shifted ratio. The chunk size is determined by the maximum context window of the model—for instance, 8K tokens for LLaMA-3-8B-Instruct. As shown in Figure 6 (Left), the optimal shifted ratio is found to be around 10% to 20%.

A4: Compatibility with KV Cache Compression. In real-world applications, the memory transfer overhead of KV caches poses a significant bottleneck. To evaluate the compatibility of CoPMix with recent KV cache compression techniques, we explore its integration with both quantization-based and eviction-based methods. Specifically, we consider KIVI [28] for KV cache quantization and SnapKV [29] for eviction. For KIVI, we quantize the key and value states to 4 bits. For SnapKV, we constrain the KV cache size to 4096 and apply an attention score window size of 32. Results in Table 3 demonstrate that CoPMix is highly compatible with these compression techniques—both quantization and eviction methods achieve performance comparable to using CoPMix alone, while surpassing the full baseline by up to 9.59%.

5. Conclusion

In this paper, we introduce a novel strategy with Context-Position Duo-Mixture for effective and efficient long-context processing. Our approach leverages chunk-wise context encoding combined with a position mixture mechanism to preserve contextual coherence while maintaining compatibility with existing KV cache compression techniques. Through extensive evaluations, we demonstrate that CoPMix not only achieves superior performance compared to baseline methods such as Full Attention and StarAttention, but also offers substantial gains in computational efficiency, scaling linearly with input length. These results highlight the potential of CoPMix as a practical and scalable solution for long-context language modeling.

References

[1] Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. Booksum: A collection of datasets for long-form narrative summarization. *arXiv preprint arXiv:2105.08209*, 2021.

[2] Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed H Awadallah, Dragomir Radev, and Rui Zhang. Summˆ n: A multi-stage summarization framework for long input dialogues and documents. *arXiv preprint arXiv:2110.10150*, 2021.

[3] Ming Zhong, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. Dialoglm: Pre-trained model for long dialogue understanding and summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11765–11773, 2022.

[4] Xueying Du, Mingwei Liu, Kaixin Wang, Hanlin Wang, Junwei Liu, Yixuan Chen, Jiayi Feng, Chaofeng Sha, Xin Peng, and Yiling Lou. Classeval: A manually-crafted benchmark for evaluating llms on class-level code generation. *arXiv preprint arXiv:2308.01861*, 2023.

[5] Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x. In *KDD*, 2023.

[6] Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*, 2023.

[7] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

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

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

[10] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

[11] Guanzheng Chen, Xin Li, Zaiqiao Meng, Shangsong Liang, and Lidong Bing. Clex: Continuous length extrapolation for large language models. *arXiv preprint arXiv:2310.16450*, 2023.

[12] Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. Llm maybe longlm: Self-extend llm context window without tuning. *arXiv preprint arXiv:2401.01325*, 2024.

[13] Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. Training-free long-context scaling of large language models. *arXiv preprint arXiv:2402.17463*, 2024.

[14] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[15] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[16] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*, 2024.

[17] Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[18] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[19] Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. Why does the effective context length of llms fall short? *arXiv preprint arXiv:2410.18745*, 2024.

[20] Alexander Peysakhovich and Adam Lerer. Attention sorting combats recency bias in long context language models. *arXiv preprint arXiv:2310.01427*, 2023.

[21] Yuhan Chen, Ang Lv, Ting-En Lin, Changyu Chen, Yuchuan Wu, Fei Huang, Yongbin Li, and Rui Yan. Fortify the shortest stave in attention: Enhancing context awareness of large language models for effective tool use. *arXiv preprint arXiv:2312.04455*, 2023.

[22] Hongzhan Lin, Ang Lv, Yuhan Chen, Chen Zhu, Yang Song, Hengshu Zhu, and Rui Yan. Mixture of in-context experts enhance llms' long context awareness. *arXiv preprint arXiv:2406.19598*, 2024.

[23] Zhenyu Zhang, Runjin Chen, Shiwei Liu, Zhewei Yao, Olatunji Ruwase, Beidi Chen, Xiaoxia Wu, and Zhangyang Wang. Found in the middle: How language models use long contexts better via plug-and-play positional encoding. *arXiv preprint arXiv:2403.04797*, 2024.

[24] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

[25] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[26] Arghadip Das, Arnab Raha, Shamik Kundu, Soumendu Kumar Ghosh, Deepak Mathaikutty, and Vijay Raghunathan. Xamba: Enabling efficient state space models on resource-constrained neural processing units. *arXiv preprint arXiv:2502.06924*, 2025.

[27] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

[28] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

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

[30] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H _2 o: Heavy-hitter oracle for efficient generative inference of large language models. *arXiv preprint arXiv:2306.14048*, 2023.

[31] Yijiong Yu, Huiqiang Jiang, Xufang Luo, Qianhui Wu, Chin-Yew Lin, Dongsheng Li, Yuqing Yang, Yongfeng Huang, and Lili Qiu. Mitigate position bias in large language models via scaling a single dimension. *arXiv preprint arXiv:2406.02536*, 2024.

[32] Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long T Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, et al. Found in the middle: Calibrating positional attention bias improves long context utilization. *arXiv preprint arXiv:2406.16008*, 2024.

[33] Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. *arXiv preprint arXiv:2502.20766*, 2025.

[34] Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. Seerattention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*, 2024.

[35] Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. *arXiv preprint arXiv:2405.05254*, 2024.

[36] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[37] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

[38] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. 2023.

[39] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[41] Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

[42] Olga Golovneva, Tianlu Wang, Jason Weston, and Sainbayar Sukhbaatar. Contextual position encoding: Learning to count what's important. *arXiv preprint arXiv:2405.18719*, 2024.

[43] Xiaoran Liu, Hang Yan, Shuo Zhang, Chenxin An, Xipeng Qiu, and Dahua Lin. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*, 2023.

[44] Xinyu Yang, Tianqi Chen, and Beidi Chen. Ape: Faster and longer context-augmented generation via adaptive parallel encoding. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*.

[45] Shantanu Acharya, Fei Jia, and Boris Ginsburg. Star attention: Efficient llm inference over long sequences. *arXiv preprint arXiv:2411.17116*, 2024.

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

[47] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

[48] Mistral AI. Mistral-7b-instruct-v0.3, 2023. URL `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3`. Accessed: 2025-03-28.

[49] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.