LongHeads: Multi-Head Attention is Secretly a Long Context Processor

Anonymous ACL submission

Abstract

Large language models (LLMs) have achieved impressive performance in numerous domains but often struggle to process lengthy inputs effectively and efficiently due to limited length generalization and attention's quadratic computational demands. Many sought to mitigate this by restricting the attention window within the pre-trained length. However, these methods introduce new issues such as ignoring the middle context and requiring additional training. To address these problems, we propose **LongHeads**, a *training-free* framework that enhances LLM's long context ability by unlocking multi-head attention's untapped potential. Instead of allowing each head to attend to the full sentence, which struggles with generalizing to longer sequences, we allow each head to process in-distribution length by selecting and attending to important context chunks. To this end, we propose a chunk selection strategy that relies on the inherent correlation between the query and the key representations, efficiently distributing context chunks to different heads. In this way, **each head ensures it can effectively process attended tokens within the trained length, while different heads in different layers can collectively process longer contexts.** LongHeads works efficiently in linear time, fits seamlessly with many LLMs that use relative positional encoding. LongHeads achieves 100% accuracy at the **128k** length on passkey retrieval task, verifying LongHeads's efficacy in extending the usable context window for existing models.

1 Introduction

LLMs are usually required to handle tasks with long contexts, such as in-context learning (Dong et al., 2023), tool learning (Qin et al., 2023), and retrieval-augmented generation (Gao et al., 2024). However, enabling LLMs to process long contexts presents significant challenges. When the context length exceeds the pre-training length, the model struggles to adapt to longer position encoding, leading to the out-of-distribution (OOD) issue(Han et al., 2023). And quadratic complexity of attention introduces considerable training and inference costs. Although OOD issue could be addressed by zero-shot learning (Jin et al., 2024), fine-tuning (Chen et al., 2023a; Peng et al., 2023), or re-training (Sun et al., 2022; Press et al., 2022), the required memory and computation still increases quadratically with context length, as shown in Figure 1(a).

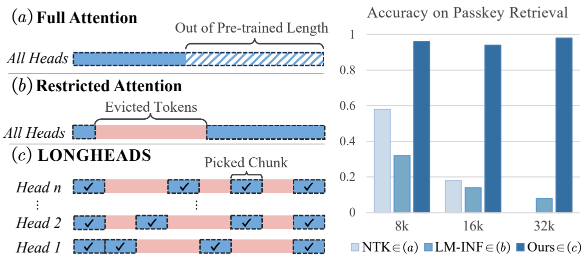


Figure 1: Left: Three types of long-context processors, (a) Attend all contexts **but** struggle with out-of-pre-trained length; (b) Attend local context to generate fluently **but** lose information; (c) Head attends short chunks **and** Heads attend Long context. Right: Accuracy of three specific methods on passkey retrieval task.

To alleviate these issues, recent works restrict the attention window to pre-trained length, which reduces the computation cost and avoids the processing of OOD tokens. One direction is to exclude distant tokens (except for a few initial tokens, Han et al., 2023; Xiao et al., 2023) to restrict the attention window in-distribution, as shown in Figure 1(b). However, these methods could result in losing critical information, degrading performance on downstream tasks. The other way to constrain the attention window is to retrieve chunks of long sequences (Mohtashami and Jaggi, 2023; Zhang et al., 2024), but these approaches usually require special operations and continuous fine-tuning, which makes it difficult for existing LLMs to be directly applicable to long sequences. In summary, improving the ability of LLMs to handle long contexts at a low cost is still challenging.

In this paper, we propose **LongHeads**, a novel framework to enhance LLM's long context ability without additional training. The key idea is to fully unlock the potential of multi-head attention. We utilize the inherent characteristic of multi-head attention: **different heads focus on different subspaces of the context, and each head can effectively process sequences within the pre-training length**(Michel et al., 2019). As shown in Figure 2 (c), we limit each head to selecting and attending to important contextual chunks within pre-trained length, rather than having each head attend to the entire sentence, thereby avoiding the OOD problem. Furthermore, we leverage the model's inherent dot-product attention and propose a chunk selection strategy to find important chunks for each head. Drawing inspiration from the fact that **each head assigns different attention weights to tokens based on the inherent correlation between the query and the key representations**, we break the input into chunks and create chunk-level features for each block. It utilizes native token-level correlation to construct chunk-level queries and key representations, which allows each head to utilize its existing capabilities (dot-product attention) to select chunks based on the attention weights. In this way, each head effectively processes selected context chunks within the trained length, and all heads in all layers work together to handle longer contexts. Meanwhile, all operations are based on the intrinsic capabilities of multi-head attention, allowing LongHeads to enhance LLMs without additional training.

To evaluate the effectiveness of LongHeads, we employ LLaMA-2-7B-Base and LLaMA-2-7B-Chat as base models and evaluate on language modeling, synthetic retrieval task and long context benchmark. LongHeads achieving nearly 100% accuracy across context lengths from 4k to 32k on the Passkey Retrieval task. On LongBench, LongHeads achieves the state-of-the-art (SOTA) performance among *restricted attention* methods. Compared with *full attention* methods, LongHeads achieves comparable performance on 16K test lengths and the best performance on 32K test lengths while enjoying linear computational cost. The experimental results demonstrate that LongHeads enables the LLMs to directly generalize to longer sequences and achieve comparable or even superior performance compared to the methods that require continuous fine-tuning.

Our contributions can be summarized as follows:

- We propose LongHeads, a training-free inference framework that leverages the structural properties of attention heads to process long sequences efficiently and effectively.

- We design a simple yet effective chunk selection strategy that can accurately select useful chunks and cover the full context.

- Experiments demonstrate that LongHeads is a SOTA restricted-attention-based long context processor and works efficiently in linear time, also with comparable performance to full-attention methods.

2 Method

In this section, we describe how the LongHeads utilizes the inherent ability of multi-head attention to encode and generate long sequences without *additional training*.

2.1 Overview

An overview of LongHeads is shown in Figure 2. We break the text into chunks and calculate the chunk representations for each chunk (Section 2.2). When generating token x_{14}, we pick the relevant k chunks based on the current token's query vector and chunk representations. In this way, each attention head of the LongHeads selectively focuses on different text chunks according to its preference (Section 2.3). The tokens of attended chunks are then restructured, ensuring the subsequent causal attention always performed within the pre-trained length.

When encoding or generating an out-of-length token, a parameter-free chunk selection network picks the relevant k chunks based on the current query vector and chunk representations. Unpicked chunks can be approximated as having zero attention score (Vig, 2019; Abnar and Zuidema, 2020) (this usually holds under the sparsity of the attention mechanism (Correia et al., 2019; Qin et al., 2022)), and do not need to be computed. This allows the attention matrix not to increase with length, significantly reducing the memory and computational cost (Section 2.4). Other works that restrict the scope of attention simply ignore distant tokens beyond a few initial tokens, even if they contain information worthy of attention.

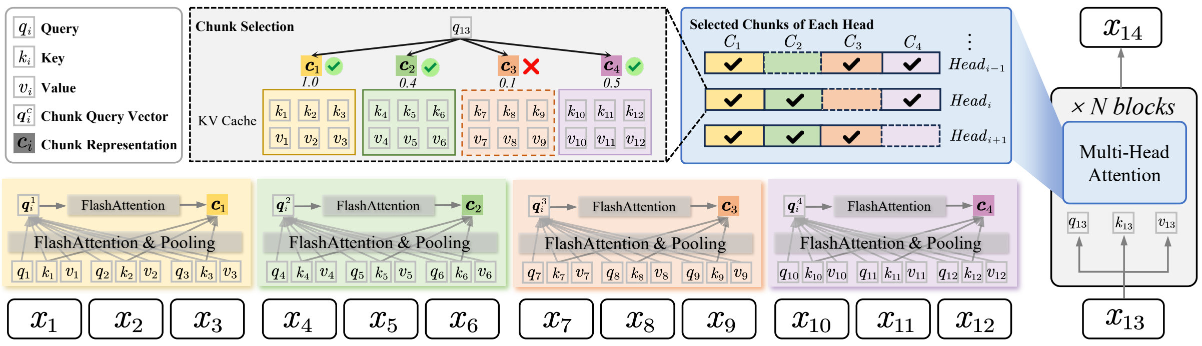


Figure 2: An overview of LongHeads's inference, generating token x_{14} in the current step. During inference, LongHeads keeps the first chunk for stable computation, combined with the last chunk containing recent tokens.

2.2 Chunk Representation

Chunk representation is an indicator of whether the tokens in this chunk should be attended to. We obtain chunk representations in a training-free manner by utilizing the attention's intrinsic abilities.

Formally, given a long input sequence $X = (x_1, ..., x_n)$, we segment it into chunks according to a predefined chunk size l, then the input sequence can be denoted as $X = (C_1, ..., C_m), m = \lceil \frac{n}{l} \rceil$. We use attention's key states to generate chunk representation for each chunk due to the existing attention mechanism that relies on query states. There are numerous straightforward methods to obtain chunk representation, such as mean pooling of the key vectors of all tokens in the chunk. However, they have demonstrated suboptimal performance in preliminary experiments, particularly in selecting the correct chunks. We hypothesize that this is attributed to the significance of individual tokens within a chunk vary substantially.

To address the above problem, we should identify the tokens that can represent the entire chunk. For that purpose, we evaluate each token's significance to the chunk and perform scaled attention aggregation on all tokens' key states to obtain a representative chunk representation as follows:

$$\boldsymbol{c}_i = \text{flash-attention}\left(\boldsymbol{q}_i^c, \boldsymbol{K}_i, \boldsymbol{K}_i\right) \quad (1)$$

where $\boldsymbol{c}_i \in \mathbb{R}^{m \times d}$ is the chunk representation, $\boldsymbol{K}_i \in \mathbb{R}^{l \times d}$ is the attention's all key states of chunk C_i, $\boldsymbol{q}_i^c \in \mathbb{R}^{1 \times d}$ is a query vector to indicate which token's key state is suitable for representing the chunk representation, we utilize flash-attention (Dao et al., 2022) to perform scaled attention. Next, we describe how to create the query vector.

A good chunk query vector should be able to represent the chunk's full semantic information, i.e., the *value* vector of all tokens in the entire chunk. However, different tokens do not contribute equally to the semantic representation, e.g., content words hold a higher semantic weight, while function words contribute less. Utilizing the inherent dot-product similarity between token-level query and key representations, we construct semantic weights for each token through a bidirectional self-attention aggregation. From the perspective of message passing, semantically rich content words will transmit more of their information to other tokens, whereas function words transmit little. Finally, the query vectors $\boldsymbol{q}_i^c$ that successfully summarize the complete semantics are obtained by mean-pooling of the aggregated representations, and can be formalized as follows.

$$\begin{aligned} \boldsymbol{O}_i &= \text{flash-attention}(\boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i) \\ \boldsymbol{q}_i^c &= \text{mean}\left(\boldsymbol{O}_i\right), \end{aligned} \quad (2)$$

where $\boldsymbol{Q}_i$, $\boldsymbol{K}_i$, and $\boldsymbol{V}_i \in \mathbb{R}^{l \times d}$ are all query states, key states, and value states of chunk C_i respectively. Both $\boldsymbol{K}_i$ and $\boldsymbol{V}_i$ can be directly accessed from the KV cache, whereas $\boldsymbol{Q}_i$ requires temporary storage during the calculation of the current chunk's representation and is released thereafter.

2.3 Chunk Selection Strategy

During the encoding or generation of the next token (denoted by x_j), we employ a query-aware chunk selection strategy, picking the k most relevant chunks from those already generated. Based on prior knowledge, there are two mandatory chunks. One is aligning with Xiao et al. (2023)'s findings, acknowledging the essential role of the few start tokens of a sentence in preserving the stability of LLMs. If the few start tokens are missing from the context, the pre-trained LLMs will completely lose their expressive ability (i.e., exhibit very high perplexity). To ensure fluency, all attention heads uniformly select the first chunk (i.e., C_1) of the sentence. Otherwise, the LLM cannot handle downstream tasks (as demonstrated in the Ablation Study). The other is assigning the last chunk (i.e., C_{-1}) to all attention heads, in order to provide the model with the local information necessary for generation.

Next, we pick the remaining $k-2$ most relevant chunks for each attention head. In the attention module of LLMs, the dot product score reflects the relevance of the context token to the current token. Inspired by it, we pick target chunks by the dot product similarity between the current token's query state $\boldsymbol{q}_j$ and the chunk representation $\boldsymbol{c}_i$.

$$P = \{C_1\} \cup \{C_i \mid \mathrm{rank}(\boldsymbol{q}_j \cdot \boldsymbol{c}_i) \leq k-2\} \cup \{C_{-1}\}, \tag{3}$$

where P is the final set of selected chunks, and the $\mathrm{rank}(\cdot)$ function outputs the rank of the current chunk's computed similarity among all candidates. In this way, different attention heads across the layers naturally attend to different parts of the context, retrieving the important chunks for inference.

Position Remapping. There are text chunks in the set P that exceed the pre-training length, so the positional encoding of P needs to be remapped. The total length of the selected chunks is controlled to be within the pre-training length L, i.e., $k{*}l < L$. Here, LONGHEADS restructures the picked chunks and concatenates them, while preserving the order of precedence. In Figure 3, the current head attends to chunks $(1, 2, 5, 7)$ among the eight candidate chunks. The positions are assigned as $[1, 4l]$, in contrast to the original text positions, which would be $[1, l] \cup [l{+}1, 2l] \cup [4l{+}1, 5l] \cup [6l{+}1, 7l]$. Position remapping avoids the out-of-distribution problem encountered when extending the context even without further training.

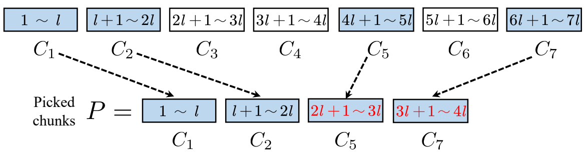


Figure 3: Demonstration of Position Remapping.

2.4 Inference with LONGHEADS

We separately describe the encoding of long inputs and the generation of long outputs during the inference. Here we describe only the modified multi-head causal attention layer.

Computation and Memory in Encoding Phase. When the LONGHEADS receives long inputs, it first computes the representations of all chunks in parallel. This can be quickly achieved through two passes of *flash-attention*, with the number of tokens involved in the attention equal to the chunk size (i.e., l=256, which is much smaller than the length of the input, e.g., n=16k). The second step is to select the k most relevant chunks for each query based on chunk representations and to obtain their key and value representations, making the attention window equals to $k{*}l{=}w$ (e.g., w=2k, which is also much smaller than n). Finally, length-restricted causal flash-attention is performed efficiently.

Computation and Memory in Generation Phase. During the generation process, LONGHEADS first performs chunk selection, then loads the Key-Value representations of the picked k chunks for length-constrained causal attention. When generating with very large inputs (e.g. 100K), the KV cache (except the chunk representations) can be offloaded to CPU to significantly reduce memory usage, and we only load the picked chunks into the GPU memory. We always retain the query-key-value representations of recent tokens (not exceeding the chunk size) during the generation process. When the number of recent tokens equals the chunk size, we compute a chunk representation, similar to the encoding phase, and append it to the previous chunk representations.

Overall, the time complexity approximates an LLM with window attention $O(w^2)$ (window size w is equal to $k * l$). Memory usage of the decoding phase approximates $O(n + w^2)$, and can be further reduced to $O(k * l + w^2)$, avoiding a quadratic increase in costs with sequence length. We empirically evaluate the LONGHEADS' memory footprint and speed in Appendix D.

3 Experiment

We evaluate the proposed LONGHEADS primarily using the LLaMA-2 (Touvron et al., 2023) considering its wide adoption and popularity. The effectiveness of LONGHEADS is evaluated on three kinds of tasks: language modeling, synthetic retrieval task and long context benchmark.

3.1 Settings

Implementation. Our method is applied to LLaMA-2-7B *base* and *chat* models for empirical studies. In our setup, we set the size of each chunk l to be 256. During each inference step, we

Method	PG19 4k	PG19 16k	PG19 32k	Proof-pile 4k	Proof-pile 16k	Proof-pile 32k
Full Attention						
PI-16K	7.42	6.72	$>10^3$	2.98	2.61	$>10^3$
NTK	6.98	9.58	19.3	2.99	3.00	4.05
Restricted Attention						
LLaMA-2-7B	6.98	$>10^3$	$>10^3$	2.99	$>10^3$	$>10^3$
LM-Infinite	6.98	7.33	7.75	2.99	2.96	3.10
Landmark	10.03	10.13	10.14	4.98	4.86	4.92
LONGHEADS	6.98	8.15	8.41	2.99	3.26	3.42

Table 1: Sliding window perplexity of different context window extension methods on PG19 and Proof-pile. LONGHEADS extends the original LLaMA-2's context window length to 32k with 2k attention window.

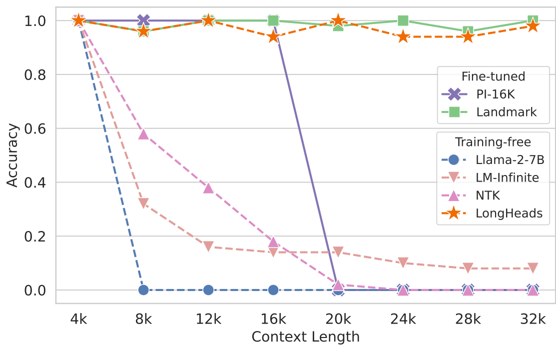


Figure 4: The evaluation of passkey retrieval task at different context lengths. LONGHEADS achieves a comparable performance as Landmark Attention and outperforms other methods.

employ our chunk selection strategy to perform query-aware chunk selection. All evaluations are conducted on a single NVIDIA A100 GPU.

Baselines. The following types of baselines are chosen for comparison. 1) The method with full attention, including "Dynamic NTK" interpolation (NTK, Emozilla, 2023) and Position Interpolation (PI, Chen et al., 2023a). 2) The method with restricted attention, including LM-Infinite (Han et al., 2023) and Landmark-Attention (Mohtashami and Jaggi, 2023). The implementation details of baselines are in Appendix A.

3.2 Long Context Language Modeling

The experiment on long context language modeling is performed with two datasets: PG19 (Rae et al., 2019) and Proof-pile dataset (Azerbayev et al., 2023). Details are shown in Appendix B.1.

The evaluation results are reported in Table 1. Although the PPL of LLaMA-2-7B-Base model and PI remain low within the pre-training context length, it increases significantly when the context exceeds this window. The NTK approach can maintain low PPL values for sequences up to 16k length, but PPL rises significantly at 32k context length. In contrast, LONGHEADS, Landmark Attention and LM-infinite successfully maintain a low PPL score even at a sequence length of 32k.

3.3 Retrieval-Based Evaluation

We conduct experiments on the passkey retrieval task introduced by (Mohtashami and Jaggi, 2023). This task challenges a language model to accurately locate and retrieve a simple passkey (a five-digit random number) in a long text sequence and we show the test example in Appendix E. The passkey is placed with various context lengths (ranging from 4k to 32k with 4k interval). For each context length, we perform 50 tests with the passkey placed at a random position in the context.

In Figure 4, we can see that all the models can output the passkey within the pretrained length. The base model completely fails at the extended length. The NTK and LM-Infinite induce a significant drop in accuracy for models at lengths surpassing 6k tokens, with accuracy falling below 20% when token lengths exceed 16k. LM-Infinite can only access 10% passkey with its local window, despite having low PPL at 32k length. Conversely, Landmark Attention and LONGHEADS consistently retrieve with nearly 100% accuracy regardless of sequence length. We further test LONGHEADS to 128k length after offloading KV cache to CPU, the results are shown in Appendix F.

We further test "Needle in a Haystack" (gkamradt, 2023) passkey retrieval, the results are shown in Appendix G.

3.4 Long Context Benchmark Evaluation

Language modeling tasks have proven to be insufficient metrics for ensuring success in downstream tasks (Sun et al., 2021), while synthetic password retrieval tasks often do not align with real-world scenarios. It is significant to conduct real downstream task evaluations to more comprehensively reflect the model's long sequence capabilities. We opt LongBench (Bai et al., 2023) for downstream NLP task evaluation, the details are shown in Appendix B.2. The results are listed in Table 2. We also conduct experiments on LLaMA-2-7B-Chat model, and the results are shown in Appendix I.

Method	FT Tokens	Single-Doc QA			Multi-Doc QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
		NQA	Qspr.	MulFi	HQA	WMQA	Musq.	GRpt	QMSM	MulN	TREC	TriQA	SMSM	PsgC	PsgR	Lcc	Repo	
									Full Attention									
NTK	-	16.47	29.62	31.42	31.31	28.75	10.20	22.70	17.65	6.31	64.67	77.36	37.95	3.99	5.12	65.64	52.97	31.38
PI-16k	0.85B	21.37	31.78	36.67	37.56	27.47	15.98	13.55	20.69	1.18	63.00	89.24	25.64	5.67	11.33	67.05	56.02	32.76
									Restricted Attention									
LM-Infinite	-	14.34	20.75	26.18	20.37	20.08	5.87	16.70	7.01	2.28	54.67	76.69	15.64	4.30	7.00	62.90	52.74	25.47
Landmark	0.80B	11.35	23.91	20.96	26.95	26.25	5.22	17.74	19.15	**9.84**	42.67	80.73	35.45	**5.73**	7.00	59.74	42.76	27.22
LongHeads	-	14.51	21.58	30.32	30.07	25.28	9.15	**24.74**	20.26	6.30	55.00	83.26	34.27	2.45	9.39	65.01	50.65	30.14
w/ NTK init	-	16.48	28.63	31.36	31.19	**28.67**	13.54	22.85	17.63	6.38	**65.33**	77.49	**38.07**	4.32	4.97	65.56	**52.87**	31.58
w/ PI init	0.85B	**21.43**	**31.78**	**36.64**	**37.63**	27.33	**15.98**	13.36	**20.57**	1.30	63.00	**89.57**	25.86	5.67	**11.33**	**66.93**	48.96	**32.33**
									Extend to 32k									
NTK	-	5.74	29.05	31.39	28.98	27.03	9.34	22.00	15.13	5.40	**64.67**	48.34	34.50	3.89	4.85	57.54	45.29	27.07
PI-16k	0.85B	8.43	30.15	35.20	29.47	24.72	1.74	13.23	12.59	1.30	55.00	66.15	19.16	5.42	**11.33**	33.21	27.21	23.39
LM-Infinite	-	10.87	20.58	26.19	19.48	20.40	**16.52**	5.26	2.51	6.14	55.00	82.78	11.26	4.30	6.67	64.88	**56.02**	25.55
Landmark	0.80B	13.88	23.69	21.06	28.04	25.78	11.52	17.70	19.11	**10.68**	41.00	77.15	**35.61**	**5.70**	7.00	58.22	40.97	27.32
LongHeads	-	13.38	21.81	30.33	29.59	24.90	11.48	**27.43**	19.87	6.07	55.00	81.15	33.56	2.79	10.06	63.75	47.97	29.95
w/ NTK init	-	9.01	27.67	31.68	30.04	**27.06**	8.31	22.44	17.20	5.41	63.33	54.61	35.13	4.09	4.70	60.59	48.92	28.14
w/ PI init	0.85B	**20.28**	**31.39**	**37.15**	**36.45**	26.55	15.30	14.75	**20.68**	1.30	62.00	**88.35**	22.81	5.33	**11.33**	**66.93**	54.28	**32.00**

Table 2: The results of different methods based on the LLaMA-2-7B-Base model on **LongBench**. FT Tokens indicate the number of tokens used for continuous training.

Comparison with Restricted Attention Methods. LongHeads surpasses the current methods with restricted attention. Specifically, LongHeads performs better than the method with the sliding window mechanism on LongBench (+4.67 vs. LM-Infinite). Compared to the method with chunking strategy (i.e., Landmark Attention), LongHeads exceeds the average score by 2.92 on LongBench without additional training. This indicates that the chunk selection strategy in LongHeads can accurately supplement LLMs with relevant contextual information, enabling efficient and effective understanding on long sequences.

Comparison with Full Attention Methods. Full attention methods can increase the maximum sequence length of LLMs but also raise computational and memory costs. LongHeads can be augmented with PI or NTK methods during the encoding phase, achieving comparable or even better results with a shorter window size, significantly reducing computational overhead. This suggests that LongHeads has the potential for scalability, and can be strengthened with a stronger base model.

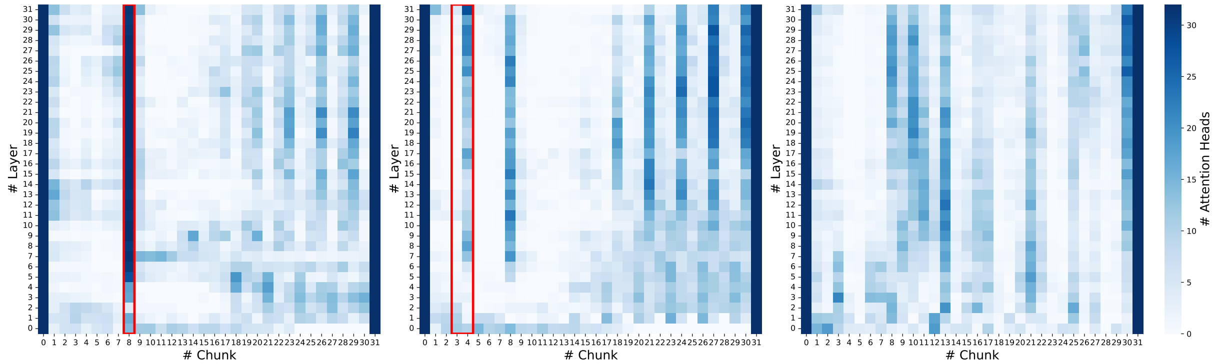
Performance when extending to 32k Context window. A desirable attribute for RoPE-extension methods is that the models should maintain their performance when directly extending to a longer context window. When extending to 32k context windows, PI and NTK methods struggle with the out-of-demonstration issue and tend to compromise model performance. In contrast, LongHeads maintains its performance and outperforms all the baseline methods.

4 Discussion

4.1 Analysis

In this section, we explore how different attention heads handle long contexts and whether they find important information. We set LongHeads's attention window to 2048 and analyze on passkey retrieval and summary tasks. We visualize the tests for both tasks in Figure 5 and show the statistical results in Table 3. The details of analytical experiments are in Appendix C.

Attention heads focus on important parts in context. On the passkey retrieval task, shown in Figure 5(a), all attention heads focused on the same chunk containing the answer and predicted it accurately. Even when the passkey is not successfully predicted in Figure 5(b), the chunks containing the answer are still selected by multiple heads. In contrast, on the summary task in Figure 5(c), the attention heads spread their focus more evenly to summarize the entire information. Similarly, Table 3 reveals a lower uniformity score for the summary task compared to the passkey retrieval task. These findings suggest that our chunk selection strategy results in a more uniform distribution of selections in the summary task, while the distribution in the passkey retrieval task is more concentrated. We attribute this to the specificity of chunks required for the passkey retrieval task, whereas the sum-



(a) Passkey Retrieval Task (Success) (b) Passkey Retrieval Task (Fail) (c) Summarization Task

Figure 5: Visualization of chunks selected by different attention heads at each layer represented by color blocks. For the passkey retrieval task, the chunk containing the passkey is delineated with a red border. In example (b), the red border encompasses two chunks due to the passkey-containing sentence coincidentally spanning two chunks. We conduct a statistical analysis to investigate the influence of chunking the key into different chunks in Appendix H.

Input Length	Cover Rate	Uniformity	Hit Rate	
			Top 1	Top 5
Passkey Retrieval				
4k	100	0.52	0.55	0.96
8k	100	0.52	0.89	0.96
16k	99.2	0.60	0.99	1.00
32k	82.0	0.76	0.98	0.98
Summary				
4k	100	0.31	/	/
8k	100	0.44	/	/
16k	100	0.49	/	/
32k	100	0.57	/	/

Table 3: Statistical results with different sequence lengths. Cover Rate is defined as the percentage of selected chunks out of the total number of chunks. Uniformity of the distribution of chunk selection is evaluated by the Gini coefficient, with lower values indicating a more uniform distribution. Hit Rate means the probability that the top-1 and top-5 selected chunks contain the correct answer in the past key retrieval task.

mary task necessitates various parts of the text to formulate a comprehensive answer. Moreover, the probability of the top 5 selected chunks containing the answer is almost 100% across all test lengths in Table 3. These results suggest that our chunk selection strategy adaptively fits the characteristics of different tasks, and allows different attention heads to concentrate on task-related content.

Attention heads can handle long sequences in a short window. In Figure 5, the lower layer attention heads focus on the more dispersed text in both tasks, while the upper layer attention heads focus more on specific chunks. We speculate that different attention heads naturally focus on different parts of the information in the text at lower layers, collecting and aggregating the entire long document information in a short length, while the upper layer attention heads are responsible for processing the aggregated information, mainly focusing on the chunks needed to complete the task. In Table 3, the Cover Rate is 100% in most cases. Given that different heads in each layer can select varying chunks, the maximum theoretical length accessible by LONGHEADS is $|P| \times$ n_heads $\times$ n_layers (e.g., the maximum length for LLaMA-2-7B with 4k attention window is 512k). These observations demonstrate that we have successfully utilized a limited attention window to capture almost all information from the entire long document.

4.2 Ablation Study

We conduct ablation experiments to investigate the influence of chunk selection strategy, attention heads flexibility, number of chunks K, and chunk size l. The ablation study is constructed on LongBench and the results are presented in Table 4.

Effect of Chunk Selection Strategy. We find that the performance when selecting the highest-scoring chunks significantly surpasses that of the lowest-scoring (Last K) chunks, and even Random $P \backslash \{C_1, C_{-1}\}$ yields better results than Last K Selection. We also observe a significant performance degradation when the first chunk is not preserved(Random P and w/o C_1). This is because the absence of the first chunk results in the model's output distribution collapsing directly. Our findings are consistent with StreamingLLM (Xiao et al., 2023) and LM-Infinite (Han et al., 2023).

Method Setting	LongBench Avg.
LongHeads	**30.14**
- Random P	7.12
- Random $P \backslash \{C_1, C_{-1}\}$	28.77
- Last K Selection	26.22
- w/o C_1	14.06
- Fix Head	29.46
- Fix Layer	28.78
- Fix Head & Layer	28.72
- Number of Chunks $K = 8$	29.09
- Number of Chunks $K = 4$	26.64
- Chunk Size $l = 512$	29.95
- Chunk Size $l = 128$	29.35

Table 4: Ablation study on **LongBench**, by default $l = 256$, $K = 16$, and Top K Selection. Random P means all chunks are randomly selected and Random $P \backslash \{C_1, C_{-1}\}$ means keep the first and last chunk and randomly select the remaining chunks.

Effect of Heads Flexibility. When the flexibility of attention heads is constrained, the model's performance is compromised to varying degrees (-0.68 Fix Head, -1.36 Fix Layer, -1.42 Fix Head&Layer). This demonstrates that within the LongHeads framework, the collaboration of different attention heads in each layer plays a crucial role.

Effect of Number of Chunks & Chunk Size. Increasing the number of chunks in a text may provide more information, but the benefits show a diminishing return. This indicates that four chunks provide enough information to ensure performance, and eight chunks are already adequate to access the entire sequence's information with chunk selection strategy, Different chunk sizes do not lead to a significant impact on the results, indicating larger or smaller chunk sizes are feasible for LongHeads.

5 Related Work

Expanding Positional Encoding (PE). Context extension studies typically target the popular RoPE encoding, aiming to scale unseen PE into the space of positions seen during pre-training. Chen et al. (2023a), and concurrently kaiokendev (2023) discovered that interpolating the position indices within the pre-trained limit works well with the help of a small amount (a few billion, Chen et al., 2023a) of fine-tuning. However, position interpolation (PI) equally stretches all dimensions of RoPE, neglecting the variations in frequency. As an alternative, Bloc97 (2023b) proposed the "NTK-aware" interpolation by taking the loss of high-frequency components into account. Subsequently, Emozilla (2023) proposed the "Dynamic NTK" interpolation method, which performs well without the need for fine-tuning. Bloc97 (2023a) introduced the "NTK-by-parts" interpolation method, which performs the best when fine-tuned on a small amount of longer-context data. Peng et al. (2023) proposed YaRN, an improved method to efficiently extend the context window by fine-tuning on less than 0.1% of the original pre-training data. This work directly modifies the PE to expand to a theoretically infinite context length. In contrast, our method does not require modifying the PE, and only a finite chunk participates in the attention calculation, which improves efficiency and reduces memory usage.

Restricted Attention. In addition, the global causal attention could be restricted to local attention, thus avoiding exceeding the pre-trained position length. ReRoPE (Su, 2023) truncates all context lengths to the max length during pretraining. LM-Infinite (Han et al., 2023) restricted the global attention window into a chevron-shaped window, retaining only a few tokens from the beginning of the text and a local window. Mohtashami and Jaggi (2023) insert a learnable landmark token after each text fragment with a fixed length, and use these landmarks to retrieve relevant fragments. Zhang et al. (2024) similarly insert a learnable beacon token and use its representation to summarise the corresponding whole fragment. Although restricted attention offers advantages in terms of memory usage and inference speed, they risk losing valuable context information. Existing methods employ local windows that are either fixed or selected through fine-tuning. In our approach, local windows are flexibly composed of chunks from the context and do not rely on additional fine-tuning.

6 Conclusion

We present LongHeads, a novel, training-free framework for efficiently processing long contexts in pre-trained LLMs. Utilizing the intrinsic capabilities of attention heads, LongHeads smartly segments and assigns long text to relevant heads, streamlining the handling of extended sequences without extra computational load. Experiment results validate LongHeads's superiority in restricted attention setups and its competitive edge against full attention methods when applied to the LongBench suite. Our approach paves the way for performance breakthroughs in long context LLM operations, leveraging existing model structures to unlock new potential without further training.

Limitations

We summarize the limitations of our method as follows: (1) Splitting the text into chunks may disrupt the continuity of the content. When the correct answer is in the middle of two chunks, this kind of splitting can affect the performance of downstream tasks. (2) The theoretical maximum length accessible by LONGHEADS is confined to $|P| \times$ n_heads $\times$ n_layers. LONGHEADS cannot fully access inputs that surpass this threshold. However, LONGHEADS can still perform well on long document tasks by selecting important parts from the context. (3) The success of LONGHEADS in downstream tasks depends on the non-parametric chunk selection function. For complex comprehension tasks, the effectiveness of the selection function may be affected.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics.

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. 2023. Longbench: A bilingual, multitask benchmark for long context understanding.

Bloc97. 2023a. Add NTK-Aware interpolation "by parts" correction.

Bloc97. 2023b. NTK-Aware Scaled RoPE allows LLaMA models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. Extending context window of large language models via positional interpolation.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023b. Longlora: Efficient fine-tuning of long-context large language models.

Together Computer. 2023. Redpajama: An open source recipe to reproduce llama training dataset. `https://github.com/togethercomputer/RedPajama-Data`.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning.

Emozilla. 2023. Dynamically Scaled RoPE further increases performance of long context LLaMA with zero fine-tuning.

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.

gkamradt. 2023. Needle in a haystack - pressure testing llms.

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

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.

kaiokendev. 2023. Things iḿ learning while training superhot.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one?

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023. Tool learning with foundation models.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. 2022. cosformer: Rethinking softmax in attention.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2019. Compressive transformers for long-range sequence modelling.

Jianlin Su. 2023. Rectified rotary position embeddings. https://github.com/bojone/rerope.

Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. 2021. Do long-range language models actually use long-range context?

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. 2022. A length-extrapolatable transformer.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Jesse Vig. 2019. Visualizing attention in transformer-based language representation models.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks.

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

A Baseline Implementation Details

We conduct experiments on 4 methods as our baselines.We illustrate the details of each baseline as follows:

For NTK, we set the scale factor of NTK to 2.0 for base model and 1.0 for chat model. For LM-Infinite, we set the number of preserved initial tokens to 10 and the local window at the end to 4096 tokens. In the context of training-free methods, we did not evaluate StreamingLLM (Xiao et al., 2023) as their framework does not support inputs exceeding 4K tokens, and their method is similar to LM-Infinite. For Position Interpolation method performed on 8K and 16K context, we use the Redpajama (Computer, 2023) dataset for training. Following (Chen et al., 2023b), we set the per-device batch size as 1 and gradient accumulation steps as 8, which means that the global batch size equals 64, using 8 GPUs. We train the models for 1000 steps. For Landmark-Attention, we adopted their configuration settings for consistency. We finetune LLaMA-2-7B Base model for 15000 steps using their method. We fine-tune the model with context length 512 on Redpajama dataset.

B Evaluation Details

B.1 Language Modeling Evaluation Details

We evaluate the long context language modeling performance on the book corpus dataset PG19 (Rae et al., 2019) and the cleaned Arxiv Math proof-pile dataset (Azerbayev et al., 2023). For both datasets, a subset of one hundred instances from the test corpus is utilized to gauge language modeling proficiency. Following (Press et al., 2022), we evaluate perplexity by using a sliding window approach with S = 256.

B.2 Long Context Benchmark Evaluation Details

Following Jin et al. (2024); Zhang et al. (2024), we opt Longbench (Bai et al., 2023) for downstream NLP task evaluation, including Single-Document Question Answering (QA), Multi-Document QA, Summarization, Few-shot Learning, and Code Completion. To ensure a more balanced and rational evaluation of the model's long-text capabilities, we employed tasks from LongBench-E to replace the corresponding tasks in Longbench for our testing. We follow LongBench (Bai et al., 2023) to evaluate the models on 16k context window sizes by truncating the prompt from the middle when the

Method	4k			16k			32k			128k		
	Time(s)	Mem(GB)	Acc(%)	Time(s)	Mem(GB)	Acc(%)	Time(s)	Mem(GB)	Acc(%)	Time(s)	Mem(GB)	Acc(%)
Llama2	**0.03**	18.8	**100**	-	OOM	-	-	OOM	-	-	OOM	-
Flash Atten	**0.03**	**17.2**	**100**	0.11	30.8	0	0.21	49.0	0	-	OOM	-
LM-Infinite	0.05	**17.2**	32	0.10	38.6	14	0.17	65.6	8	-	OOM	-
LongHeads	**0.03**	19.0	96	**0.08**	**29.9**	**94**	**0.11**	**47.1**	**98**	**4.14***	**42.3***	**100***

Table 5: Statistical results with decoding speed, memory usage, and passkey retrieval accuracy. Decoding speed (seconds / per token) is averaged over 100 token inferences at each length. Memory consumption corresponds to peak GPU usage during inference. * denotes LONGHEADS with offloading the Key-Value (KV) cache to the CPU. Passkey retrieval accuracy is tested by 50 tests at each length. All tests are conducted on a single NVIDIA A100 80GB GPU.

Input Prompt: There is an important info hidden inside a lot of irrelevant text. Find it and memorize them. I will quiz you about the important information there. [Garbage context] The pass key is {pass_key}. Remember it. {pass_key} is the pass key. [Garbage context]
Instruction: What is the pass key? The pass key is

Table 6: Prompt details for passkey retrieval.

task length exceeds a designated context window size.

For LONGHEADS, the attention window size is set to 4k. LONGHEADS can be integrated with other extrapolation methods belonging to the Full Attention methods, significantly reducing their computational cost. LONGHEADS w/ NTK init refers to integrated "Dynamic NTK" interpolation (Emozilla, 2023). LONGHEADS w/ PI init refers to integrated Position Interpolation (Chen et al., 2023a).

C Analysis Experiments Details

We conduct analytical experiments on the tasks of passkey retrieval and summary. For the passkey retrieval task, we compiled statistics for the results with sequence lengths of 4k, 8k, 16k, and 32k, as mentioned in Section 3.3. Regarding the summary task, we select the government report dataset from the LongBench, from which we chose 5 samples each for lengths of 4k, 8k, 16k, and 32k for statistical analysis.

D Efficiency Analysis

We empirically evaluate the LONGHEADS' memory footprint and speed. In comparison to the full attention method with Flash-Attention 2 (Dao et al., 2022), as the context length increases, LONGHEADS exhibits superior throughput and reduced memory consumption (achieving a speedup of **1.4x** on 16k and **1.9x** on 32k). Compared to current methods such as LM-Infinite(Han et al., 2023), LONGHEADS demonstrates distinct advantages in memory and throughput across various lengths.

LONGHEADS also offers a trade-off between memory and time by offloading the Key-Value (KV) cache to the CPU. After this offloading process, the model achieves 100% accuracy on the passkey retrieval task at a text length of 128k, with the peak GPU memory usage being only 42.3 GB. The offloading operation is flexible and is triggered when memory is insufficient.

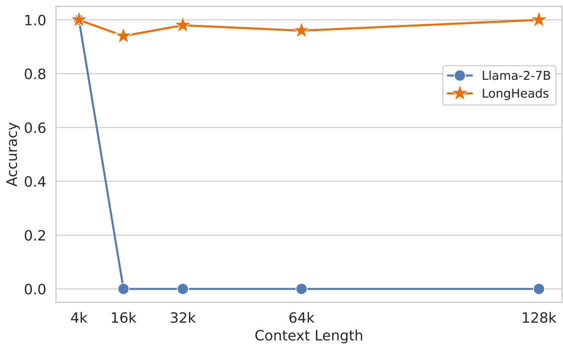


Figure 6: The evaluation of passkey retrieval task from 4k to 128k.

E Passkey Retrieval Example

We provide the prompt details for the passkey retrieval test in Table 6. For tests of different lengths, we use garbage context of varying lengths to pad the text, ensuring that the position of the passkey is randomly inserted.

F LONGHEADS on 128k Context

We further extend LLaMA-2-7b to 128k with LONGHEADS without additional training. LONGHEADS achieves 100% accuracy at 128k length on passkey retrieval task, the results are shown in Figure 6. After offloading the KV cache to CPU,

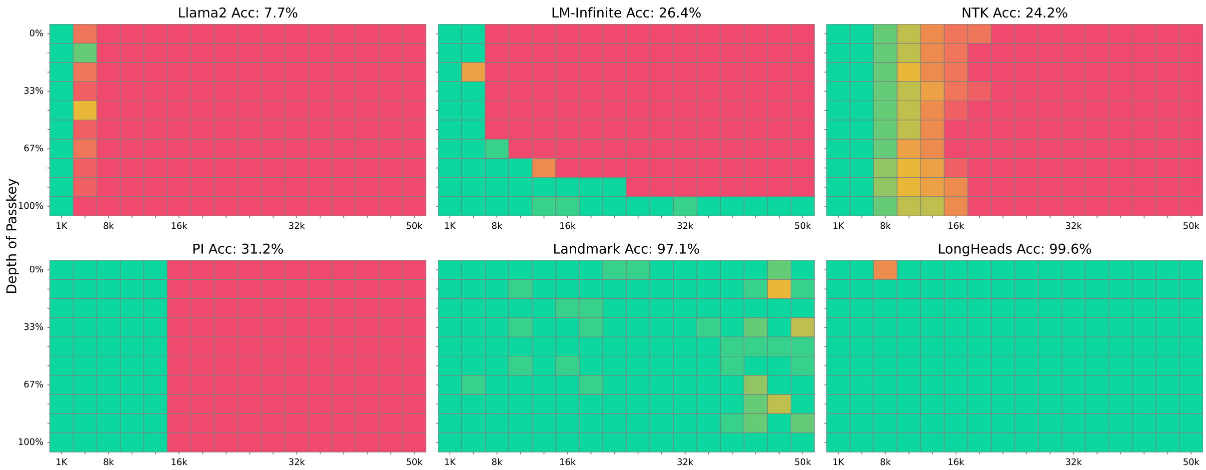


Figure 7: Testing "Needle in a Haystack" Passkey Retrieval with a 50K Context. The X-axis represents the input context length, and the Y-axis indicates the depth of the passkey within the document. For each depth, we run 10 different test cases.

	Passkey Unsplit	**Passkey Split**
Acc.	96.9% (690/712)	87.5% (77/88)

Table 7: Statistical analysis of the effects of splitting the passkey into different chunks.

peak GPU memory usage is 26.51GB and 44.48 GB when inference with 64k and 128k context.

G "Needle in a Haystack" Passkey Retrieval

Following (gkamradt, 2023), We place the passkey at various document depths, ensuring that they are distributed uniformly. For each document depth, we run 10 times with different passkeys and we test the input sequence length from 1k to 50k with a 3k interval. The performance results are shown in Figure 7. Notably, LongHeads outperforms other baselines and achieves an overall accuracy score of 99.6% across all examples tested.

H Chunking Influence

We conduct a statistical analysis to investigate the influence of chunking the key into different chunks on the performance of the passkey retrieval task. We statistic all test samples (800 in total) of different lengths in the passkey task, calculating the accuracy when the passkey is divided and undivided into different chunks, as shown in Table 7.

I More Results on LongBench

Tabel 8 shows that LongHeads also has strong performance on LLaMA2-7b-Chat models. When encoding is enhanced with NTK, LongHeads is able to achieve comparable performance to the full attention method.

Method	**FT**	**Single-Doc QA**			**Multi-Doc QA**			**Summarization**			**Few-shot Learning**			**Synthetic**		**Code**		**Avg.**
	Tokens	NQA	Qspr.	MulFi	HQA	WMQA	Musq.	GRpt	QMSM	MulN	TREC	TriQA	SMSM	PsgC	PsgR	Lcc	Repo	
Chat Model																		
LM-Infinite	-	0.00	18.57	25.33	9.87	11.73	0.48	11.30	2.99	8.72	32.50	29.22	13.82	5.61	5.20	34.19	24.55	14.63
NTK	-	15.18	**30.89**	36.14	35.10	25.79	**13.53**	**31.48**	20.21	23.86	**61.67**	**80.94**	**39.43**	**7.40**	13.33	48.96	42.45	**32.90**
LongHeads	-	11.61	22.98	23.76	31.28	24.10	8.87	25.36	20.24	16.18	50.67	79.98	36.74	6.39	9.67	**53.85**	**44.22**	29.12
w/ NTK init	-	**16.87**	30.32	**38.59**	**36.04**	**26.72**	10.21	31.28	**20.91**	**24.46**	55.67	76.72	39.07	6.07	**14.67**	49.97	40.27	32.37

Table 8: The results of different methods based on the LLaMA-2-7B-Chat model on **LongBench**.