LLMs See Further: A Survey Comparing Approaches to Extend Context Window in LLMs

[1] **Mrinal Bhan**
Data Science & Artificial Intelligence
IIIT Naya Raipur
mrinal21102@iiitnr.edu.in

[2] **Sai Prabhat Gubbala**
Data Science & Artificial Intelligence
IIIT Naya Raipur
sai21102@iiitnr.edu.in

[3] **Dr. Avantika Singh**
Data Science & Artificial Intelligence
IIIT Naya Raipur
avantika@iiitnr.edu.in

Abstract—**Large language models (LLMs) have emerged as a dominant force in natural language processing, demonstrating remarkable capabilities in various tasks. However, their performance is often restricted by a fundamental limitation: limited context window size. This constraint hinders their ability to handle complex tasks requiring extensive contextual understanding, such as long document summarization, few-shot learning, and code completion across large codebases. To address this challenge, researchers have actively explored various techniques for extending the context window in LLMs. This paper presents a comprehensive comparative analysis of these methods, delving into their theoretical underpinnings and limitations. We will scrutinize how approaches like positional encoding, attention mechanisms, and pre-training on large corpora attempt to capture long-range dependencies and enhance context understanding within LLMs. This analysis transcends mere implementation details, focusing on the theoretical rationale behind each technique and its expected impact on LLM performance. By providing a critical evaluation of existing approaches, this survey aims to serve as a valuable resource for researchers in the field. We hope to stimulate discussions on future advancements in context window extension techniques, ultimately fostering the development of more robust and versatile LLMs.**

Index Terms—**Large Language Models, Context Window, Positional Encoding, Attention Mechanisms, Long-Range Dependencies, Context Understanding**

I. Introduction

A. Background

The accomplishment of Large Language Models (LLMs) are widespread, with the emergence of modern LLMs greatly advancing several Natural Language Processing (NLP) difficulties and reaching unparalleled heights. Among the ambitious initiatives, one notable effort is the extension of LLMs' understandability to encompass very long contexts. OpenAI has introduced the concept of 128 pages of context understandability, while Anthropic has recently proposed an even longer context of over 200 pages. However, a notable absence of scientific rigor is observed and hence several questions arise in this context:
(a) What applications necessitate the understanding of such extended contexts?
(b) How can we effectively measure the performance of applications when LLMs comprehend much longer contexts?
(c) While attention mechanisms are well-studied in NLP, is there a need to devise a new form of attention specifically tailored for longer contexts?

The integration of advanced techniques designed to handle long contexts holds the potential to reshape the landscape of language models. Improved methodologies for managing long contexts could lead to increased model performance, resulting in more accurate and nuanced language understanding. Such advancements are anticipated to enhance the model's ability to capture long-range dependencies, improving its overall effectiveness across various language tasks like:

- **Document Summarization:** Improved long context handling allows for more coherent and concise document summarising, capturing important information across extended text segments and improving the quality of output summaries. A full knowledge of the entire document, together with the identification of keywords and subjects, needs the effective administration of a large contextual scope. Using a shorter frame in this scenario limits the generating capacity, potentially leading to the omission of important details. Furthermore, the use of a longer contextual window aids in reducing ambiguity because it prevents the use of subtle information without a comprehensive understanding of the document's complexities. This, in turn, allows the LLM to approach the summary process with greater discernment and accuracy.

- **Question Answering:** The ability to consider extended contexts improves the model's understanding of complex question-answer interactions, leading to more accurate and contextually relevant solutions. Furthermore, LLMs outperform in QA tasks because the resolution of co-referencing pronouns is inextricably tied to contextual items. Furthermore, when confronted with multi-turn discussions, the enlargement of the context window proves crucial in supporting the consistent tracking of the conversational topic across successive chats.

- **Language Translation:** Improved preservation of context over larger text segments enhances the model's capacity to provide accurate translations, particularly in cases where contextual nuances play a pivotal role. Polysemic lexical items present a substantial impediment in the realm of translation (*Falkum and Vicente, 2015*), and an augmented context window stands as a discernible aid in the contextualization of such lexemes. Furthermore,

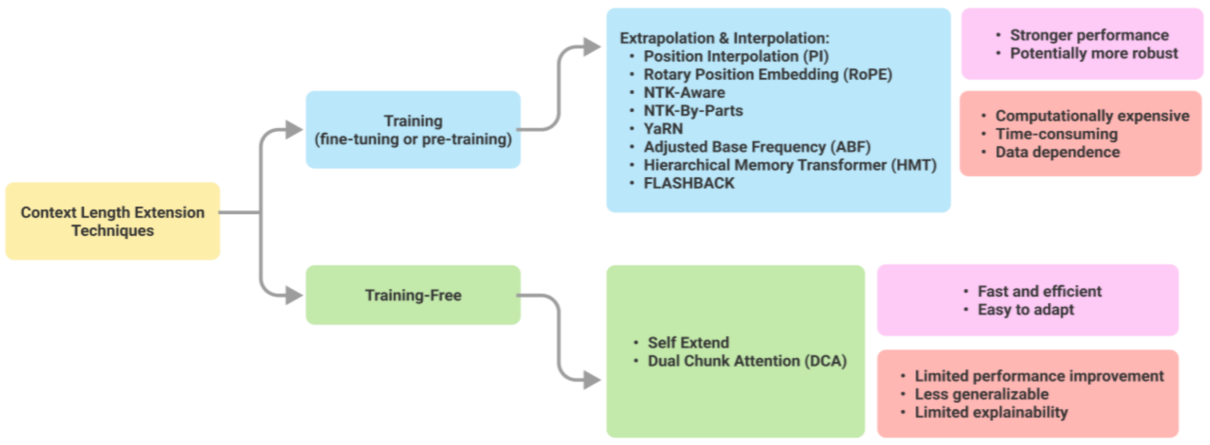
Context Length Extension Techniques
Training (fine-tuning or pre-training)
Training-Free
Extrapolation & Interpolation:
Position Interpolation (PI)
Rotary Position Embedding (RoPE)
NTK-Aware
NTK-By-Parts
YaRN
Adjusted Base Frequency (ABF)
Hierarchical Memory Transformer (HMT)
FLASHBACK
Stronger performance
Potentially more robust
Computationally expensive
Time-consuming
Data dependence
Self Extend
Dual Chunk Attention (DCA)
Fast and efficient
Easy to adapt
Limited performance improvement
Less generalizable
Limited explainability

Fig. 1. A taxonomy is proposed for categorizing strategies that extend the context duration in Language Models (LLMs). The graphic categorizes the strategies into Training and Training Free Approaches.

when confronted with technical jargon, LLMs exhibit enhanced efficacy when endowed with an extended input scope, particularly in accommodating domain-specific contextual nuances. .

- **Conversational AI:** Long context models provide better tracking and understanding of lengthy interactions, resulting in more contextually appropriate responses from conversational AI systems. LLMs rely heavily on extended context windows to place humor, sarcasm, and subtle phrases inside the conversational setting. This is necessary for the development of responses that adhere to the desired tone and stylistic nuances inherent in the ongoing debate.

B. *Contribution*

Despite persistent research efforts, a comprehensive overview encompassing the entire range of techniques for extrapolating context length is still absent. Additionally, the continuous evolution of LLMs has introduced innovative facets for extrapolating context length, posing challenges to existing extension methodologies and underscoring the imperative for thorough, diverse extrapolation approaches. This paper marks the first comprehensive survey of techniques for extending the context length of LLMs.

II. Literature Review

A. *Related Work*

Large Language Models (LLMs) have revolutionized various tasks in natural language processing. However, their potential is limited by a crucial constraint: the **pre-defined context window size** during pre-training. This limitation hinders their performance in tasks requiring extensive context, such as few-shot learning and long document summarization. For instance, LLaMA 2 (*Touvron et al., 2023b*) is pre-trained on sequences of only 4,096 tokens, and exceeding this window often leads to performance degradation. This primarily stems from the limited length extrapolation ability of their position encoding methods (*Kazemnejad et al., 2023*). To address this challenge, recent research has explored extending the context window in LLMs by modifying **Rotary Position Embedding (RoPE)** (*Su et al., 2021*), a popular encoding method adopted by state-of-the-art models such as LLaMA (*Touvron et al., 2023a,b*), PaLM (*Chowdhery et al., 2023; Anil et al., 2023*), and GPT-NeoX (*Black et al., 2022*). These efforts aim to overcome the limitations of LLMs in increasingly context-demanding applications like few-shot learning (*Brown et al., 2020*), long document summarization (Huang et al., 2021), and repository-level code completion (*Liu et al., 2023*). Several RoPE-extension approaches have emerged, including **Position Interpolation (PI)**, **adjusted base frequency (ABF)**, and **YaRN**. While these methods demonstrate improved performance, a comprehensive comparison and understanding of their underlying mechanisms are lacking. Additionally, YaRN's effectiveness in scaling attention logits requires further analysis.

B. *Objectives*

This paper aims to bridge this gap by addressing three key questions:

1) **Comparative Performance:** Which context-extension method exhibits the best performance in supervised fine-tuning on context-demanding tasks?
2) **Data Efficiency:** How efficiently can each method utilize training data?
3) **Robustness:** How robust is the performance of models trained with these methods across varying context window sizes?

III. Preliminaries

A. *Training-Based*

Rotary Position Embedding (RoPE): Given a position index m [1, c] and an embedding vector x:= [x0, x1, . . . , xd1] , where d is the dimension of each attention head, RoPE considers each pair of elements along the feature dimension of the embedding vector as complex numbers and encodes position information by rotating them. The vector-valued complex function f(x, m) defined by RoPE is as follows:

$$\mathbf{f}(\mathbf{x}, m) = \begin{bmatrix} (x_0 + \mathrm{i}x_1)e^{\mathrm{i}m\theta_1}, \\ (x_2 + \mathrm{i}x_3)e^{\mathrm{i}m\theta_2}, \\ \dots, \\ (x_{d-2} + \mathrm{i}x_{d-1})e^{\mathrm{i}m\theta_{d/2}} \end{bmatrix} \quad (1)$$

i := 1 is the imaginary unit and j = b 2j/d , where b denotes the base frequency of RoPE and is set to 10, 000 by default. In application, RoPE is applied to both query and key embeddings through the following equation:

$$\mathbf{f}(\mathbf{x}, m) = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{d-2} \\ x_{d-1} \end{bmatrix} \otimes \begin{bmatrix} \cos(m\theta_0) \\ \cos(m\theta_0) \\ \cos(m\theta_1) \\ \cos(m\theta_1) \\ \vdots \\ \cos(m\theta_{(d-1)/2}) \\ \cos(m\theta_{(d-1)/2}) \end{bmatrix} + \begin{bmatrix} -x_1 \\ x_0 \\ -x_3 \\ x_2 \\ \vdots \\ -x_{d-1} \\ x_{d-2} \end{bmatrix} \otimes \begin{bmatrix} \sin(m\theta_0) \\ \sin(m\theta_0) \\ \sin(m\theta_1) \\ \sin(m\theta_1) \\ \vdots \\ \sin(m\theta_{(d-1)/2}) \\ \sin(m\theta_{(d-1)/2}) \end{bmatrix} \quad (2)$$

The fundamental components of RoPE are a series of trigonometric coefficients, each encoding position information of different frequencies. We represent these trigonometric coefficients with the following function to uniquely identify RoPE and its variants:

$$h(m, b, t) = \sqrt{t} * cos(\frac{m}{b^{\frac{2j}{d}}}) \text{ or } \sqrt{t} * sin(\frac{m}{b^{\frac{2j}{d}}}) \quad (3)$$

where m is the position index of the query token, b is the base frequency for RoPE, and t is the scaling factor for attention logits. Note that t is used in the equation because RoPE rotates process both the query and key embeddings. Before introducing RoPE-extension methods that enable better context window extension, we define context scaling factor s = c c , which is the ratio between the target context window c and the pre-trained context window c. It is of special use to those methods that extend RoPE according to a given target context window size.

Position Interpolation (PI): PI (*Chen et al., 2023; kaiokendev, 2023*) linearly interpolates the input position index m to m/s so that it falls within the original context window size. *Chen et al.* (2023) demonstrate that direct fine-tuning of LLaMA (*Touvron et al., 2023a*) with an extended context window results in minimal improvement, as the effective context window of the model only increases from 2,048 to 2560 after 10,000 training steps on sequences of length 8,192. By contrast, PI succeeds in extending the context window of LLaMA to 32,768 with only 1,000 training steps.

NTK-Aware: NTK-Aware scaling (bloc97, 2023b) hypothesize that interpolating all dimensions equally, as done by PI, may result in loss of high-frequency information. Therefore, NTK-Aware scaling introduces a nonlinear interpolation strategy by adjusting the base frequency b of RoPE to b(̂d/d2) . This modification scales the low-frequency components of RoPE to a similar extent as PI, while only slightly altering the high-frequency components to avoid disturbing high-frequency information. NTK-Aware extends models' context window size without training. However, this method can't benefit as much as PI from additional training on longer sequences as suggested by (*Peng et al., 2023*).

NTK-By-Parts: NTK-By-Parts (*bloc97, 2023a*) holds that stretching all the RoPE components either by a scaling factor s or a base transformation results in token embeddings being closer to each other, impeding LLMs from effectively capturing local relationships between adjacent tokens. To address this issue, NTK-By-Parts scales (j) by a factor 1(j) s + (j), with (j) being assigned 0 for high frequencies, 1 for low frequencies, and a predetermined constant within the range of 0 to 1 for intermediate frequencies. According to (*Peng et al., 2023*), this method performs better than PI and NTK-Aware scaling.

Adjusted Base Frequency (ABF) : ABF (*Xiong et al., 2023)* simply changes the base frequency of RoPE to 50,000. Both theoretical analysis and experiment are conducted to validate the efficacy of this method. *Xiong et al.* (2023) proves that ABF minimizes the distance of its embedded vectors from the ones using the original RoPE, which helps leverage the pre-training results. They empirically validate the efficacy of ABF by showing a lower perplexity on language modeling tasks and a longer effective context window in the first sentence-retrieval task.

YaRN [1] deviates from Linear and NTK interpolation methods by using a ramp function that alters the blending of Linear interpolation.

and NTK interpolation across many dimensions. Moreover, it includes a temperature factor to counterbalance the distribution shift in the attention matrix caused by long inputs.

Hierarchical Memory Transformer (HMT) [2] : Hierarchical Memory Transformer (HMT) replicates the memory structure of the brain, as described by Burgin in 2011. It utilizes both learned memory tokens and prior input tokens. The Hierarchical Model of Memory (HMT) categorizes memory into three distinct levels: sensory memory, short-term memory, and long-term memory. These levels of memory interact with each other. The HMT (Hierarchical Memory Transformer) utilizes a memory recall mechanism by storing encoded memory embeddings that are formed from earlier iterations. It then searches for relevant information depending on the current token segments. Additionally, it efficiently manages the process of transitioning between various pertinent subjects. HMT, or Hierarchical Memory Transformer, follows a structured approach to process input in segments. Let me break down each step for clarity:

Representation Extraction: This initial step involves encoding a partial segment of tokens into a single embedding. Essentially, it condenses the information from that segment into a compact representation.

Memory Search: In this step, the generated embedding serves as a query to search for relevant information within the memory or history. This process helps HMT retrieve context or knowledge from previous segments or the broader context.

Prepending Sensory Memory: After obtaining relevant information from memory, HMT augments the current segment. It incorporates tokens from the previous segment and possibly relevant historical information, enhancing the segment's content.

Segment Processing and Summarization: Finally, the augmented segment undergoes processing to produce hidden embeddings. These embeddings are then used to generate logits, which are probabilities used in decision-making or predicting the next steps. Additionally, HMT creates a memo or summary to capture the essential aspects of the segment processing and the information retrieved from memory.

FLASHBACK : Efficient Retrieval-Augmented Language Modeling for Long Context Inference : FLASHBACK, tackles limitations in handling long contexts. It focuses on improving inference efficiency in Retrieval-Augmented Language Models (RALMs). Existing RALMs prepend retrieved content, causing slow inference. Unlike traditional RALMs that prepend retrieved documents, FLASHBACK utilizes a more efficient approach by:

1. *Appending Retrieved Documents*: By appending retrieved documents, FLASHBACK better aligns with the access patterns of the model's cache. This simple change significantly reduces inference runtime compared to prepending.

2. *Learnable Marking Token*: To mitigate any potential performance loss from the appended structure, FLASHBACK introduces a learnable marking token. This token helps the model distinguish between the original input and the retrieved content, ensuring optimal utilization of both sources.

B. *Training-Free Based*

1) Grouped Attention: : For a pretrained LLM with relative position encodings, such as RoPE, the behavior of the LLMs becomes unpredictable during inference if the length of a sequence is longer than its pretraining context window length. This has been explored by (Han et al., 2023; Chen et al., 2023b) that with unseen relative positions, the attention distributions are very different compared to those within the pretraining context window length. We argue that such failure stems from the Out-of-Distribution (O.O.D.) relative distance in the same sense that neural networks are not robust to O.O.D. inputs (Shen et al., 2021).

One feasible and straightforward way to handle unseen relative positions is to map them to positions that were seen during pretraining. We can use the FLOOR operation to map the unseen positions to positions within the pretraining context window, as shown in Figure. The proposed method is identical to the original self-attention mechanism except that the FLOOR operation is applied to each token's original position before the inner product. We denote the self attention with the FLOOR operation applied as "grouped attention". In Python style, the "grouped attention" is denoted as:

$$P_g = P \quad // \quad G_s,$$

where $P \in \mathbb{R}^{B \times L}$ is the original position encoding, in which B is the batch size and L is the length of the input text sequence. Gs denotes the group size, which is the base of the FLOOR operation. Taking the floor of the position divided by the group size maps the original large position values to a smaller discrete set of values, avoiding the issue of out-of-distribution position values during inference.

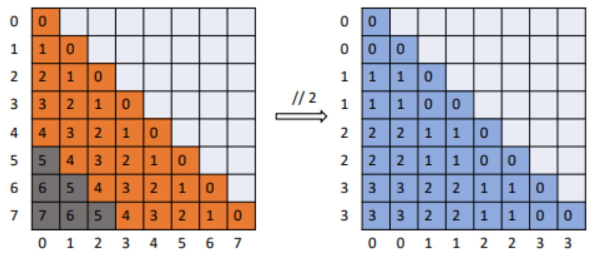

Fig. 2. Illustration of grouped attention. We suppose that the LLM's pretraining context window length is 5 and the length of the inference sequence is 8. On the left figure, we show the positional Out-of-Distribution (O.O.D.) issue while the input length is out of the pretraining context window size. The y-axis of this matrix represents the position of query tokens and the x-axis represents the position of key tokens. In this case, in the relative position matrix, only those in orange are seen during pretraining. Relative positions in gray are outside the pretraining context window. In the right figure, we show how the FLOOR operation is applied and the relative position matrix for grouped self attention. With the Gs set as 2, the positions of query tokens and key tokens are mapped from 0-7 to 0-3 by FLOOR (//). The new relative positions (in blue) are all within the range of the pretraining context window

We show the perplexity (PPL) on the PG-19 (Rae et al., 2019) dataset with the FLOOR operation applied to Llama-2-7b-chat across different sequence lengths, in Figure 2. As a comparison, we also show the PPL of the original model without the FLOOR operation as the dotted lines. From this figure, with the FLOOR operation, LLMs keep a relatively low and stable PPL on the sequences whose lengths exceed the pretraining context window. Meanwhile, with grouped attention, the PPL is a little higher than the original LLMs, which is expected. However, the model's PPL behavior is similar to the original model, as the PPL is nearly unchanged within the "context window" (for Llama-2: 2 - 8192, 4 - 16384, and 8 - 32768), demonstrating the effectiveness of group attention. Once the length of a sequence is longer than the new "context window" (e.g., sequences with 10k tokens as the input, with a group size of 2), the PPL explodes again due to the positional O.O.D issue.

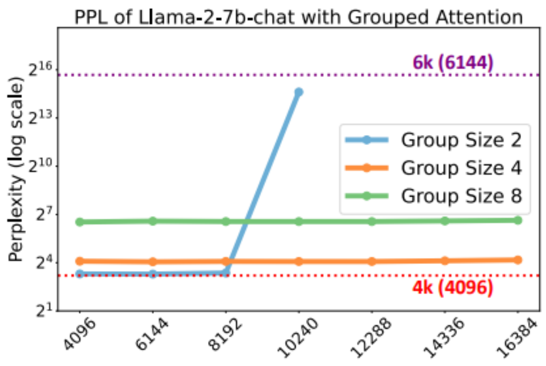

Fig. 3. Perplexity (PPL) using grouped attention with different group sizes under different sequence lengths on PG-19 dataset. The original Llama-2-7b-chat PPL is stable at 4k (4096) sequences (red dotted line) but explodes at 6k (6144) sequences (purple dotted line). The results show the LLMs keep a relatively low and stable PPL on long sequences with grouped attention.

Model	Evaluation Context Window Size						
Name	**4096**	**6144**	**8192**	**10240**	**12288**	**14336**	**16384**
Llama-2-7b-chat	9.181	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$
SelfExtend-Llama-2-7b-chat	8.885	8.828	9.220	8.956	9.217	9.413	9.274
Mistral-7b-instruct-0.1 w/ SWA	9.295	9.197	9.532	9.242	9.198	9.278	9.294
Mistral-7b-instruct-0.1 w/o SWA	9.295	9.205	10.20	55.35	$> 10^3$	$> 10^3$	$> 10^3$
SelfExtend-Mistral-7b-instruct-0.1	9.272	9.103	9.369	9.070	8.956	9.022	9.128

Fig. 4. . Perplexity on dataset PG19 with Llama-2-7b-chat and Mistral-7b-instruct-0.1. We report the PPL of withwithout Sliding Window Attention (SWA) for Mistral

2) *Dual Chunk Attention (DCA):* :

Intra-Chunk Attention: Intra-Chunk Attention is employed to calculate the inner product of queries and keys within the same chunk. For a long sequence of length l, we partition the sequence into $n = l/s$ chunks, ensuring that the position indices within each chunk will not exceed the chunk size s.

To aggregate information from other chunks, we introduce **Inter-Chunk Attention**. In Llama-based LLMs, the position indices for queries are greater than those of the keys to reflect the left-to-right information flow, i.e, we have $Pq[i] \geq Pk[j]$ whenever $i \geq j$.

Successive-Chunk Attention can be viewed as a special case for inter-chunk attention, proposed to maintain the locality of LLMs where locality means LLMs tend to heavily rely on the neighboring tokens to predict the next token (Xiao et al., 2023; Han et al., 2023). Simply using inter-chunk attention may no longer keep the precise relative position between neighboring tokens, leading to performance degradation. The inner product of q, k in DCA is consequently defined as:

$$\mathbf{q}_i^T\mathbf{k}_j = \begin{cases} f(\mathbf{q}, P_{\mathbf{q}}^{\text{Intra}}[i])^T f(\mathbf{k}, P_{\mathbf{k}}[j]), & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor = 0 \\ f(\mathbf{q}, P_{\mathbf{q}}^{\text{Succ}}[i])^T f(\mathbf{k}, P_{\mathbf{k}}[j]), & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor = 1 \\ f(\mathbf{q}, P_{\mathbf{q}}^{\text{Inter}}[i])^T f(\mathbf{k}, P_{\mathbf{k}}[j]), & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor > 1, \end{cases}$$

3) *Normalization:* :

Softmax layer The inner product calculations within the DCA are formalized as shown in Equation 8. Subsequently, a softmax function is applied to normalize the computed inner products:

$$\mathbf{p}_i = \text{softmax}\left(\left[\frac{\mathbf{q}_i^\top \mathbf{k}_0}{\sqrt{d}}, \frac{\mathbf{q}_i^\top \mathbf{k}_1}{\sqrt{d}}, \ldots, \frac{\mathbf{q}_i{}^\top \mathbf{k}_i}{\sqrt{d}}\right]\right).$$

where d denotes the dimension of hidden states

Perplexity (PPL) evaluation on PG19 (Rae et al., 2020) validation set. The results highlighted in red indicate the Perplexity has increased by more than 1.0 compared to its original value at the pretraining context length of 4096. ReRoPE (Su, 2023) encounters OOM (Out of Memory) problems with 16k input tokens as it is currently not compatible with Flash Attention. The scaling factors in PI and NTK are dynamically changed.

Model	Evaluation Context Window				
	4096	8192	16384	32768	65536
7B Finetuned Models					
Longlora-32k	**8.14**	**7.85**	**7.70**	7.80	91.79
Together-32k	8.21	7.95	7.76	**7.64**	$>10^2$
CodeLlama-16k	8.93	8.64	8.44	8.36	**8.65**
CLEX-16k	16.74	15.08	14.28	14.70	15.10
7B Training-free Models					
Llama2-RoPE	**7.87**	$>10^2$	$>10^2$	$>10^2$	$>10^2$
Llama2-ReRoPE	**7.94**	7.75	OOM	OOM	OOM
Llama2-PI	**7.87**	9.19	15.11	$>10^2$	$>10^2$
Llama2-PI-Yarn	**7.87**	8.80	11.75	42.42	$>10^2$
Llama2-NTK	**7.87**	11.98	26.12	58.91	$>10^2$
Llama2-NTK-Yarn	**7.87**	8.06	9.82	11.74	41.57
ChunkLlama2 (ours)	**7.87**	**7.67**	**7.64**	**7.89**	15.87
ChunkLlama2 13B	7.15	6.95	6.99	7.90	15.14
ChunkLlama2 70B	**5.24**	**5.18**	**5.21**	**5.30**	**5.59**

Fig. 5. Evaluation Metrics for Various LLMs

IV. Acknowledgment

We offer our heartfelt appreciation to Dr.Avantika Singh for her guidance and support during this effort. Her invaluable knowledge, insightful comments, and constructive input were critical in refining our grasp of the subject area and assisting us in overcoming the hurdles we encountered during the project. We are grateful to our institution faculty and staff for providing us with the resources and facilities to complete this project successfully. This project is a fantastic learning experience for us, and we are happy for the chance to work on it

References

[1] B. Peng, J. Quesnelle, H. Fan, and E. Shippole, *Yarn: Efficient context window extension of large language models*, 2023. arXiv: 2309.00071 [cs.CL].

[2] Z. He, Z. Qin, N. Prakriya, Y. Sun, and J. Cong, *Hmt: Hierarchical memory transformer for long context language processing*, 2024. arXiv: 2405.06067 [cs.CL].

[3] O. Press, N. A. Smith, and M. Lewis, *Train short, test long: Attention with linear biases enables input length extrapolation*, 2022. arXiv: 2108.12409 [cs.CL].

[4] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, *Roformer: Enhanced transformer with rotary position embedding*, 2023. arXiv: 2104.09864 [cs.CL].

[5] A. Ruoss, G. Delétang, T. Genewein, *et al.*, *Randomized positional encodings boost length generalization of transformers*, 2023. arXiv: 2305.16843 [cs.LG].

[6] Y. Sun, L. Dong, B. Patra, *et al.*, *A length-extrapolatable transformer*, 2022. arXiv: 2212.10554 [cs.CL].

[7] J. Ding, S. Ma, L. Dong, *et al.*, *Longnet: Scaling transformers to 1,000,000,000 tokens*, 2023. arXiv: 2307.02486 [cs.CL].

[8] H. Jin, X. Han, J. Yang, Z. Jiang, C.-Y. Chang, and X. Hu, *Growlength: Accelerating llms pretraining by progressively growing training length*, 2023. arXiv: 2310.00576 [cs.CL].

[9] A. Mohtashami and M. Jaggi, *Landmark attention: Random-access infinite context length for transformers*, 2023. arXiv: 2305.16300 [cs.CL].

[10] W. Wang, L. Dong, H. Cheng, *et al.*, *Augmenting language models with long-term memory*, 2023. arXiv: 2306.07174 [cs.CL].

[11] L. Liu, X. Yang, Y. Shen, *et al.*, *Think-in-memory: Recalling and post-thinking enable llms with long-term memory*, 2023. arXiv: 2311.08719 [cs.CL].

[12] S. Tworkowski, K. Staniszewski, M. Pacek, Y. Wu, H. Michalewski, and P. Miłoś, *Focused transformer: Contrastive training for context scaling*, 2023. arXiv: 2307.03170 [cs.CL].

[13] C. Packer, S. Wooders, K. Lin, *et al.*, *Memgpt: Towards llms as operating systems*, 2024. arXiv: 2310.08560 [cs.AI].

[14] C. Han, Q. Wang, H. Peng, *et al.*, *Lm-infinite: Zero-shot extreme length generalization for large language models*, 2024. arXiv: 2308.16137 [cs.CL].

[15] H. Jiang, Q. Wu, X. Luo, *et al.*, *Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression*, 2023. arXiv: 2310.06839 [cs.CL].

[16] S. Chen, S. Wong, L. Chen, and Y. Tian, *Extending context window of large language models via positional interpolation*, 2023. arXiv: 2306.15595 [cs.CL].

[17] D. Zhu, N. Yang, L. Wang, *et al.*, *Pose: Efficient context window extension of llms via positional skip-wise training*, 2024. arXiv: 2309.10400 [cs.CL].

[18] R. Liu, X. Xiao, H. Huang, Z. Chi, and Z. Wu, *Flashback:efficient retrieval-augmented language modeling for long context inference*, 2024. arXiv: 2405.04065 [cs.CL].

Year	Title	Technique	Model	Tasks	Remarks
2021	Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation [3]	Positional Encoding	Customized Transformer Model	Language modeling, text generation	Enables models trained on shorter sequences to perform well on longer sequences during inference, Achieves lower memory usage compared to the commonly used sinusoidal position embeddings.
2021	RoFormer: Enhanced transformer with Rotary Position Embedding [4]	Positional Encoding	Customized Transformer Model (RoFormer), BERT, WoBERT, NEZHA	Machine translation, Semantic text matching	The paper explores position encoding in transformers, introducing Rotary Position Embedding (RoPE) for improved language modeling with flexible sequence lengths and enhanced attention mechanisms.
2023	Randomized Positional Encodings Boost Length Generalization of Transformers [5]	Positional Encoding	Encoder-only Customized Transformer Model	Various algorithmic reasoning tasks	Addresses a key weakness: the inability of transformers to handle unseen sequence lengths, Doesn't discuss the specifics of the randomized encoding scheme or potential downsides (e.g., computational overhead).
2023	A Length Extrapolatable Transformer[6]	Specialized attention mechanism	Customized Transformer Model (LeX)	Details not provided	tackles position modeling in Transformers for handling short text training and long text evaluation. Introduces relative position embedding and blockwise causal attention, achieving strong performance in various settings.
2023	LongNet: Scaling Transformers to 1,000,000,000 Tokens [7]	Specialized attention mechanism	Customized Transformer Model (LongNet)	Longsequence modeling	Proposes dilated attention, a novel mechanism with linear computational complexity and efficient handling of long-range dependencies, Dilated attention might limit the model's ability to capture long-range dependencies effectively, potentially sacrificing some information in very long sequences.
2023	GrowLength: Accelerating LLMs Pretraining by Progressively Growing Training Length [8]	Window based approaches	70M, 160M and 410M LLM (specific model not mentioned)	Details not provided	introduces "GrowLength," a method for accelerating Large Language Models' pretraining by progressively increasing sequence length. It enhances efficiency and performance without extra engineering.
2023	Landmark Attention: Random-Access Infinite context length for Transformers [9]	Memory/Retrieval augmented approaches	TransformerXL, LLaMA7B	Language modeling, Next word prediction, Information retrieval over long contexts	Offers random access to the entire context while maintaining efficient memory usage, Overcomes the limitations of prior approaches that either sacrificed flexibility or relied on separate retrieval mechanisms.
2023	Augmenting Language Models with Long-Term Memory [10]	Memory/Retrieval augmented approaches	GPT-2-407M	Long-context language modeling, long-context understanding, memory augmented in-context learning	LONGMEM augments Large Language Models (LLMs) with long-term memory, enabling them to utilize rich long-context information. It employs a decoupled network architecture for memory retrieval, achieving significant performance improvements in long-context modeling and in-context learning tasks.
2023	Think-in-Memory: Recalling and Post-thinking Enable LLMs with Long-Term Memory [11]	Memory/Retrieval augmented approaches	ChatGLM-6B, Baichuan2- 13B	Response generation for long-term conversations	TiM (Think-in-Memory) enhances Memory-augmented Large Language Models (LLMs) by maintaining evolved memories and eliminating repeated reasoning, improving response quality in long-term interactions through dynamic memory updates and efficient retrieval mechanisms.
2023	Focused Transformer: Contrastive Training for Context Scaling [12]	Memory/Retrieval augmented approaches	Decoder-only Customized Transformer Model (LongLLaMA), OpenLLaMA 3B, OpenLLaMA 7B	Passkey retrieval, QA, Long-context language modeling	Tackles a key challenge in extending context length by addressing the issue of irrelevant information overwhelming the model, Allows for effectively using a larger context, improving task performance requiring long-range dependencies.
2023	MemGPT: Towards LLMs as Operating Systems [13]	Memory/Retrieval augmented approaches	GPT-4	Deep memory retrieval task, Conversation opener task, Multi-document QA, Nested key-value retrieval requiring multi-hop lookups	MemGPT uses virtual context management inspired by hierarchical memory systems to extend Large Language Models' context windows, enhancing performance in document analysis and multi-session chat by managing different storage tiers effectively.

Year	**Title**	**Technique**	**Model**	**Tasks**	**Remarks**
2023	LM-Infinite: Simple On-the-Fly Length Generalization for Large Language Models [14]	Specialized attention mechanism	MPT-7B, LLaMA, GPT-J-6B, LLaMA-2	Long text generation	LM-Infinite boosts LLMs' ability to handle long contexts (up to 200M tokens) without retraining, but might neglect mid-sequence information due to its attention mask focusing on beginnings and ends.
2023	LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression [15]	Prompt compression based approaches	GPT-3.5- Turbo, LongChat-13B	Single and Multi Doc QA, Summarization, Code completion, Sentiment classification, Information reordering	LongLLMLingua compresses input prompts to enhance Large Language Models' (LLMs) performance, reduce costs, and decrease latency across various long-context scenarios, achieving notable gains in performance and efficiency.
2023	Extending Context Window of Large Language Models via Positional Interpolation [16]	RoPE based approaches	LLaMA-7B, 13B, 33B, and 65B	Language modeling, Passkey retrieval, Long document summarization	Extends the context window size of RoPE-based LLMs (like LLaMA) to up to 32768 tokens with minimal fine-tuning, Doesn't mention how it might affect performance on tasks within the original context size.
2023	YaRN: Efficient Context Window Extension of Large Language Models [1]	RoPE based approaches	LLaMA-2 7B, LLaMA-2 13B, GPT-NeoX and Mistral 7B v0.1,MistralLite 7B, PaLM	Passkey retrieval	YaRN extends transformer-based language models' context windows efficiently, enabling effective utilization and extrapolation to longer contexts while surpassing previous state-of-the-art methods in context window extension and generalization.
2023	PoSE: Efficient Context Window Extension of LLMs via Positional Skip-wise Training [17]	RoPE based approaches	LLaMA-7B	Language modeling, Passkey retrieval	PoSE extends LLM context windows (up to 128k tokens) efficiently by simulating long inputs within a fixed window during fine-tuning, Manipulates positions within chunks, potentially limiting the model's ability to learn absolute positional information in very long contexts.
2024	HMT: Hierarchical Memory Transformer for Long Context Language Processing [2]	Hierarchical Memory Transformer (HMT)	Model-agnostic	General language modeling (Wikitext-103, PG-19), Question answering (PubMedQA)	Leverages recurrent memory within segments to achieve better performance than context-limited and long-context models, Lack of detail on the specific memory filtering mechanism and potential computational cost increase due to additional parameters.
2024	FLASHBACK: Efficient Retrieval-Augmented Language Modeling for Long Context Inference [18]	FLASHBACK - a modular RALM with appending context pattern and Low-Rank Adaption	Llama 2-7B	Testing generation quality	FLASHBACK improves inference efficiency in Retrieval-Augmented Language Modeling (RALM) by appending retrieved documents instead of prepending them leveraging the Key-Value cache and achieves up to 4x faster inference speed. Doesn't mention details on how Marking Tokens (introduced for appending context) are fine-tuned