Extending LLM Context Window with Adaptive Grouped Positional Encoding: A Training-Free Method

Xinhao Xu[1,2*] **Jiaxin Li**[1,2*] **Hui Chen**[2†] **Zijia Lin**[1]
Jungong Han[2,3] **Guiguang Ding**[1,2†]
[1]School of Software, Tsinghua University, Beijing, China
[2]BNRist, Tsinghua University, Beijing, China
[3]Department of Automation, Tsinghua University, Beijing, China
xxhthu18@gmail.com, thulijx@gmail.com
jichenhui2012@gmail.com, dinggg@tsinghua.edu.cn

Abstract

Processing long input remains a significant challenge for large language models (LLMs) due to the scarcity of large-scale long-context training data and the high computational cost of training models for extended context windows. In this paper, we propose **Ada**ptive **Gro**uped **P**ositional **E**ncoding (AdaGroPE), a training-free, plug-and-play method to enhance long-context understanding in existing LLMs. AdaGroPE progressively increases the reuse count of relative positions as the distance grows and dynamically adapts the positional encoding mapping to sequence length, thereby fully exploiting the range of pre-trained position embeddings. Its design is consistent with the principles of rotary position embedding (RoPE) and aligns with human perception of relative distance, enabling robust performance in real-world settings with variable-length inputs. Extensive experiments across various benchmarks demonstrate that our AdaGroPE consistently achieves state-of-the-art performance, surpassing baseline methods and even outperforming LLMs inherently designed for long-context processing on certain tasks.

1 Introduction

Processing long input is essential for large language models (LLMs) (OpenAI, 2023b; Touvron et al., 2023a; Huang et al., 2025), enabling them to comprehend complex content such as academic papers, technical reports, and long-form dialogues, thereby expanding their applications in domains like healthcare, finance, and education (Wei et al., 2024; Lee et al., 2023; Xu et al., 2024; Shen et al., 2025). To support long-context processing, several LLMs with extended context windows have been developed (Chen et al., 2024b; Ruoss et al., 2023; Rozière et al., 2023). These models typically require fine-tuning with long sequences. Despite exhibiting promising results, they rely on costly long-context dataset construction and require substantial GPU resources for training. Moreover, the scarcity of high-quality long-context data continues to limit their overall effectiveness (Gao et al., 2025).

To alleviate these constraints, recent training-free approaches have revealed that LLMs trained on short contexts can exhibit latent long-context processing capabilities (Xiao et al., 2024; Jin et al., 2024). For example, approaches such as StreamingLLM (Xiao et al., 2024) and LM-infinite (Han et al., 2024) manage the long-context challenge by restricting the number of neighbor tokens during inference to stay within the pre-trained attention window, improving perplexity on long-context tasks. However, these methods often discard significant context information and show limited effectiveness on real-world long-range dependent tasks. Other methods, such as SelfExtend (Jin et al., 2024) and An et al. (2024b), extend LLMs' context windows by reusing and remapping the position embedding from pre-training, achieving promising results on both language modeling and real-world long-context tasks without additional training.

In this paper, we propose a novel training-free framework, **Ada**ptive **Gro**uped **P**ositional **E**ncoding (AdaGroPE), to extrapolate LLM context windows. Building on the principles of rotary position embedding (RoPE) (Su et al., 2024), our approach dynamically adjusts position embedding according to sequence length and token distance, progressively increasing reuse count when calculating the position embedding of more distant tokens. This method draws inspiration from RoPE's long-term decay property, which allows the model to prioritize nearby tokens while paying less attention to those farther away. Furthermore, it aligns better with human perception of the long-context text, where the positions of nearby tokens are critical for

* Equal contribution
† Corresponding author

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 573–587
July 27 - August 1, 2025

maintaining coherence and understanding, while distant tokens are processed more for their semantic content rather than their positions (Ivgi et al., 2023). To this end, AdaGroPE preserves fine-grained relative positions within a preset local window and introduces grouped reuse for farther tokens, such that the reuse count increases with distance. This design not only aligns with RoPE's principle but also better reflects human's processing of relative positions in long-context understanding, thereby enabling effective training-free extensions of LLM context windows.

AdaGroPE is a plug-and-play, training-free method that can be integrated into various LLMs. As a position embedding extension strategy, it complements and can be combined with other methods that enhance long-context understanding, such as fine-tuning on long-context datasets. Our method also stands out by dynamically adapting the position embedding mapping strategy to the input length, ensuring optimal performance in real-world scenarios with variable input lengths. We evaluate our approach across different LLMs and datasets, including language modeling, synthetic long-context tasks, and real-world long-context tasks. Experimental results show that AdaGroPE effectively extends the long-context understanding of LLMs with short-context windows, achieving state-of-the-art performance and even surpassing models natively designed for long-context processing. This demonstrates the potential of our approach to reduce reliance on expensive long-context training datasets.

In summary, our contributions are as follows:

1. We introduce a novel positional encoding strategy, AdaGroPE, which extends the range of pre-trained position embedding by gradually increasing the reuse count based on the tokens' distance.

2. We implement AdaGroPE in a dynamic, adaptive adjusted manner, maximizing the use of pre-trained position embedding in real-world scenarios with variable input lengths.

3. We evaluate our method's effectiveness across various long-context benchmarks and LLMs. Results show that AdaGroPE achieves state-of-the-art performance, even surpassing models with inherent long-context capabilities on certain tasks.

2 Method

2.1 Preliminary

In this section, we provide a brief overview of RoPE (Su et al., 2024), which serves as the foundation for our AdaGroPE method. RoPE is a crucial positional encoding mechanism designed to capture the relative positional relationships between tokens, which enhances the attention mechanisms in transformers. It extends traditional absolute position embeddings by incorporating positional information directly into the query and key vectors used in the self-attention process, allowing for more flexible handling of long sequences.

Let $\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{L-1}\}$ represent the token embeddings, where L is the sequence length, and each $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional vector. The key idea behind RoPE is to rotate the query $\mathbf{q}_i$ and key $\mathbf{k}_j$ vectors based on their positional indices i and j, so that the dot product $\mathbf{q}_i^T\mathbf{k}_j$ inherently captures the relative positional information between tokens. This is achieved by applying a complex rotation to the vectors $\mathbf{q}_i$ and $\mathbf{k}_j$. Specifically, for tokens at positions i and j, their corresponding query and key vectors are transformed as:

$$\mathbf{q}_i = f_q(\mathbf{x}_i, i), \mathbf{k}_j = f_k(\mathbf{x}_j, j), \tag{1}$$

where f_q and f_k represent the RoPE functions that apply the positional rotations. The resulting dot product between the query at position i and the key at position j depends solely on their relative positional difference $i - j$, ensuring that the model focuses on relative distances rather than absolute positions.

Specifically, RoPE constructs a relative position matrix M during self-attention, where each element $M[i][j] = i - j$ reflects the relative positional information between the i-th query and the j-th key. This matrix is structured as a Toeplitz matrix, where the same relative positions exhibit consistent values across rows and columns. Consequently, RoPE enables transformers to maintain strong relative position awareness without the need for explicit absolute position embeddings.

2.2 Progressive Reuse of Relative Positions with Increasing Count

We assume that the relative position matrix M has the i-th row denoted as m_i, with values ranging as follows for j from 0 to i:

$$m_i = [i, i-1, ..., 0], \tag{2}$$

where the maximum relative position is i. Suppose the largest observed context window during pre-training is w. When $i \geq w$, the inner product between $\mathbf{q}_i$ and $\mathbf{k}_j$ is computed using an out-of-distribution relative position encoding, which leads to performance degradation in LLMs.

To mitigate this, some methods have grouped relative positions by assigning the same relative position to neighboring tokens. These methods introduce a hyperparameter, the group size G_s, which controls the number of tokens in each group. As a result, the modified row m_i' becomes:

$$m_i' = [\lfloor \frac{i}{G_s} \rfloor, \lfloor \frac{i-1}{G_s} \rfloor, \ldots, 0]. \tag{3}$$

By ensuring that $\left\lfloor \frac{i}{G_s} \right\rfloor < w$, out-of-distribution issues can be avoided, improving LLM performance on long-context tasks. Notably, the selection of G_s depends on both w and the length of the input.

As noted by SelfExtend (Jin et al., 2024), such a direct approach fails to account for the varying sensitivity between nearby tokens and distant tokens during contextual understanding. Specifically, it applies uniform reuse of relative positions regardless of token distance. To overcome this limitation, we propose a method that progressively increases the reuse count based on the relative distance between tokens.

In particular, given a target extension length L and a maximum relative position limit P, where $L > i$ and $P \leq w$, similar to SelfExtend, our approach defines a neighbor window size w_n. For tokens with relative distances smaller than the neighbor window size, which are more sensitive to positional information during contextual understanding, we retain their original relative positions:

$$M_a[i][j] = i - j \quad \text{if } i - j < w_n, \tag{4}$$

where M_a is the relative position matrix modified by our AdaGroPE method.

For tokens with relative distances greater than the neighbor window size, the values in M_a depend on L, P, and the hyperparameter reuse ratio coefficient $r = \frac{P}{w_n}$, following three guiding principles: *minimizing reuse*, *prioritizing distant relative position reuse*, and *progressively increasing reuse count from close to distant*. These principles will be illustrated in detail based on the progressive increase of L in the following sections. Figure 1 presents an example of the expansion of the relative position matrix $M_a[i][j]$ as L increases, with $P = 16$, $w_n = 4$, and $r = 0.25$.

Minimizing Reuse and Prioritizing Distant Relative Position Reuse First, we define a sequence $\{L_n^{\max}\}_{n\in\mathbb{N}}$, representing the maximum allowable length L when the maximum reuse count required $G_s^m = n$. Naturally, $L_{n-1}^{\max} + 1$ denotes the minimum length L required for the maximum count $G_s^m = n$. Based on the definitions of $\{L_n^{\max}\}_{n\in\mathbb{N}}$ and w_n, we have:

$$\begin{aligned} L_1^{\max} &= P, \\ L_2^{\max} &= w_n + (P - w_n) \cdot 2 \\ &= 2P - w_n. \end{aligned} \tag{5}$$

This indicates that when $L_1^{\max} < L \leq L_2^{\max}$, the maximum reuse count required G_s^m is 2. Eq. (5) satisfies the principle of *minimizing reuse*, whereby relative position is reused only when L exceeds P, while simultaneously ensuring that the original relative position is employed for the nearest tokens, as shown in Eq. (4).

Furthermore, we define a sequence $\{d_n\}_{n\in\mathbb{N}}$, which represents the number of relative positions with a reuse count of $n+1$ when $G_s^m = n+1$:

$$d_n = L - L_n^{\max}. \tag{6}$$

It is evident that when $L_n^{\max} < L \leq L_{n+1}^{\max}$, at least d_n relative positions must be reused $n+1$ times to ensure all relative positions remain within the maximum relative position limit P. In this context, when $L_1^{\max} < L \leq L_2^{\max}$,

$$M_a[i][j] = \begin{cases} i - j & \text{if } i - j < L - 2d_1, \\ f_1^m(i-j) & \text{others,} \end{cases} \tag{7}$$

where

$$f_n^m(x) = L - (n+1)d_n + \lfloor \frac{x - (L - (n+1)d_n)}{n+1} \rfloor. \tag{8}$$

$f_n^m(x)$ represents the mapping from the original to the AdaGroPE-adjusted relative position corresponding to the maximum relative position reuse count $G_s^m = n+1$. This implies that the AdaGroPE-adjusted relative positions obtained through $f_n^m(x)$ all have a reuse count equal to the maximum required reuse count $n+1$.

It is important to note that Eq. (8) groups the farthest positions and assigns the same relative position to $G_s^m = n+1$ tokens within the group while preserving the relative positions of the $\mathbf{k}_j$ closer to $\mathbf{q}_i$. This *prioritization of distant relative position reuse* aligns with the notion that distant

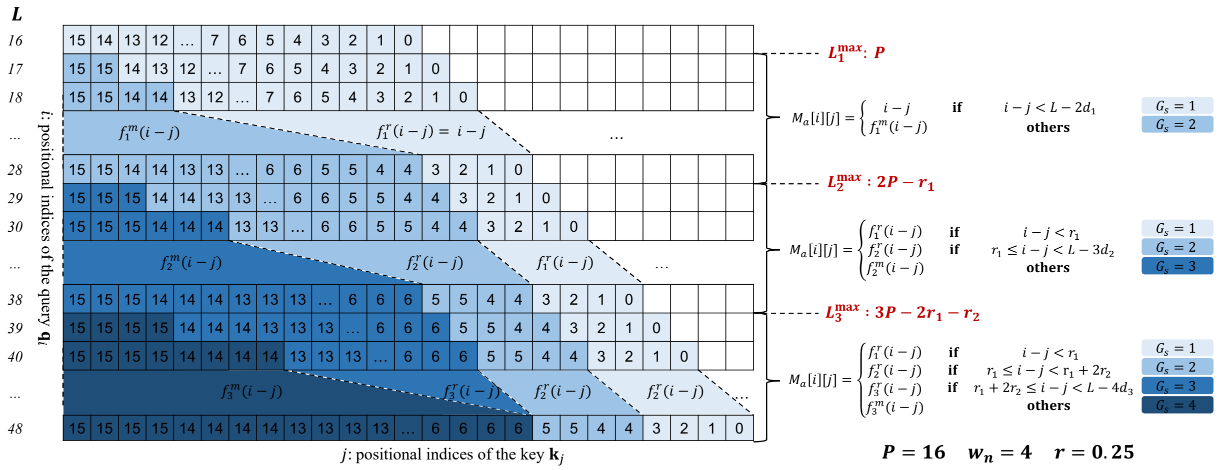

Figure 1: An example of the expansion of the computed relative positions in the AdaGroPE method. As L increases, the maximum reuse count of relative positions G_s^m increases from 1 to 4. The computed relative positions follow the principle of minimizing reuse, starting the reuse process from the farthest relative positions and progressing to the nearest. The reuse count increases with the distance from the current query.

tokens are less sensitive to positional encoding during attention calculation (Ivgi et al., 2023), and will continue to be reflected in the definition of M_a as L increases.

Progressive Increase in Reuse Count from Close to Distant The neighbor window size w_n preserves the original relative positions for keys $\mathbf{k}_j$ that are close to the query token $\mathbf{q}_i$. As L increases, we progressively increase the reuse count from close to distant tokens.

Specifically, we define a sequence $\{r_n\}_{n\in\mathbb{N}}$ as follows:

$$r_n = \begin{cases} \lfloor \frac{r}{n} \cdot P \rfloor & \text{if } \log_2 n \in \mathbb{N}, \\ 0 & \text{otherwise}, \end{cases} \tag{9}$$

where r_n denotes the minimum number of relative positions retained with reuse count n when the maximum reuse count $G_s^m > n$ as L increases. The reuse ratio coefficient r is set to 0.25 by default. Eq. (9) ensures the number of relative positions retained for each reuse count G_s as L increases. These values decrease approximately geometrically as $\lfloor rP \rfloor$, $\lfloor \frac{rP}{2} \rfloor$, $\lfloor \frac{rP}{4} \rfloor$, etc., with only the reuse count corresponding to powers of 2 being retained.

Accordingly, we calculate $L_3^{\max}$:

$$\begin{aligned} L_3^{\max} &= r_1 + 2r_2 + (P - r_1 - r_2) \cdot 3 \\ &= 3P - 2r_1 - r_2. \end{aligned} \tag{10}$$

When $L_2^{\max} < L \leq L_3^{\max}$, the maximum reuse count G_s^m increases to 3. Consequently, the corresponding relative position matrix M_a is adjusted as follows:

$$M_a[i][j] = \begin{cases} f_1^r(i-j) & \text{if } i-j < r_1, \\ f_2^r(i-j) & \text{if } r_1 \leq i-j < L - 3d_2, \\ f_2^m(i-j) & \text{others}, \end{cases} \tag{11}$$

where

$$f_n^r(x) = \sum_{k=1}^{n-1} r_k \cdot k + \lfloor \frac{x - \sum_{k=1}^{n-1} r_n \cdot k}{n} \rfloor. \tag{12}$$

$f_n^r(x)$ inductively defines the mapping function from the original relative position to the AdaGroPE-adjusted relative position, corresponding to a reuse count less than or equal to n, under the condition that the maximum reuse count $G_s^m > n$. It is evident that $f_1^r(i-j) = i-j$ and the calculation formula for $M_a[i][j]$ in Eq. (7) when $i-j < L - 2d_1$ also satisfies this function's definition.

Eq. (11) ensures that neighbor tokens around $\mathbf{q}_i$ retain their original relative positions, while reuse is progressively introduced for more distant tokens. The reuse count G_s increases in a controlled manner as defined in Eq. (9). Similarly to Eq. (7), Eq. (11) adheres to the principles of *minimizing reuse* and *prioritizing reuse for distant relative positions*. Besides, the relative positions retained by M_a follow the principle of *progressively increasing reuse count* as the distance from the $\mathbf{q}_i$ grows.

As L increases, AdaGroPE ensures that the adjusted relative positions follow the three guiding principles mentioned above. Similarly, we calculate $L_4^{\max}$ as follows:

$$\begin{aligned} L_4^{\max} &= r_1 + 2r_2 + 3r_3 + (P - r_1 - r_2 - r_3) \cdot 4 \\ &= 4P - 3r_1 - 2r_2 - r_3. \end{aligned} \tag{13}$$

To this end, when $L_3^{\max} < L \le L_4^{\max}$,

$$M_a[i][j] = \begin{cases} f_1^r(i-j) & \text{if } i-j < r_1, \\ f_2^r(i-j) & \text{if } r_1 \le i-j < r_1 + 2r_2, \\ f_3^r(i-j) & \text{if } r_1 + 2r_2 \le i-j < L - 4d_3, \\ f_3^m(i-j) & \text{others.} \end{cases} \tag{14}$$

From Eq. (14), we deduce that when $L = L_4^{\max}$, reuse count of $G_s = 3$ are eliminated, leaving only $G_s = 1, 2, 4$, consistent with the rule in Eq. (9). Furthermore, when L increases to $L_4^{\max} + 1$, the maximum required reuse count G_s^m becomes 5, implying that the farthest relative position $P - 1$ is reused five times. We can iteratively obtain the subsequent adjusted relative positions by following the pattern established in Eq. (7), Eq. (11), and Eq. (14).

2.3 Adaptive Relative Position Adjustment Strategy

Based on the explanation above, extending L from values less than P up to $L_4^{\max}$, we have clarified the three fundamental principles of AdaGroPE's relative position reuse, along with the intuitive process (illustrated in Figure 1). In this section, we will summarize the observed patterns and derive a direct formula for computing the relative position matrix using predefined values of L and P, demonstrating that our method can adaptively scale to longer target lengths L.

From Eq. (7), Eq. (11), and Eq. (14), we can derive the general expression for $L_n^{\max}$ as follows:

$$L_n^{\max} = nP - \sum_{k=1}^{n-1} (n-k-1) r_k. \tag{15}$$

Furthermore, based on the definition of $\{L_n^{\max}\}_{n \in \mathbb{N}}$, we obtain the formula for calculating the maximum reuse count G_s^m for varying lengths L as follows:

$$G_s^m(L) = \begin{cases} 1 & \text{if } L \le L_1^{\max}, \\ 2 & \text{if } L_1^{\max} < L \le L_2^{\max}, \\ \vdots & \vdots \\ n+1 & \text{if } L_n^{\max} < L \le L_{n+1}^{\max}. \end{cases} \tag{16}$$

Notation	Explanation
i, j	Absolute position indices
$M_a[\cdot][\cdot]$	Relative position in AdaGroPE
L	Target context length after extension
w	Pre-trained context window length
P	Number of relative positions used; $P \le w$
r	Coefficient controlling minimum retained positions per usage count
w_n	Size of neighbor window preserving original relative positions; $w_n = rP$
G_s	Usage count of relative positions; increases with position
$G_s^m(\cdot)$	Maximum usage count as a function of L with positions reused up to $G_s^m(L)$ times
$L_n^{\max}$	Maximum L for which $G_s^m(L) = n$, *i.e.*, $G_s^m(L) = n$ holds iff $L \in (L_{n-1}^{\max}, L_n^{\max}]$
d_n	Number of positions used $n+1$ times when $G_s^m = n+1$; $L - L_n^{\max}$ for $L \in (L_n^{\max}, L_{n+1}^{\max}]$
r_n	Minimum retained positions used n times when $G_s^m > n$; $r_1 = w_n$

Table 1: Summary of notations and their corresponding explanations in AdaGroPE.

Finally, we derive the formula for calculating relative positions in AdaGroPE for any given target extension length L and a maximum relative position limit P:

$$M_a[i][j] = \begin{cases} f_1^r(i-j) & \text{if } i-j < r_1, \\ f_2^r(i-j) & \text{if } r_1 \le i-j < r_1 + 2r_2, \\ \vdots & \vdots \\ f_n^r(i-j) & \text{if } \sum_{k=1}^{n-1} k r_k \le i-j < \sum_{k=1}^{n} k r_k, \\ \vdots & \vdots \\ f_{g_l-1}^r(i-j) & \text{if } \sum_{k=1}^{g_l-2} k r_k \le i-j < \sum_{k=1}^{g_l-1} k r_k, \\ f_{g_l}^r(i-j) & \text{if } \sum_{k=1}^{g_l-1} k r_k \le i-j < L - (g_l+1) d_{g_l}, \\ f_{g_l}^m(i-j) & \text{if } L - (g_l+1) d_{g_l} \le i-j, \end{cases} \tag{17}$$

where $g_l = G_s^m(L) - 1$.

It is straightforward to verify that Eq. (7), Eq. (11), and Eq. (14) all satisfy the above equation. To this end, we finalize the construction of the AdaGroPE relative position matrix $M_a[i][j]$, which adheres to the three fundamental principles and can be directly calculated for any specified target extension length L and maximum relative position limit P. This provides a flexible and adaptive framework for configuring the positional encoding strategy. A detailed summary of the notations and their definitions, along with the pseudocode for computing relative positions during decoding, is provided in

Table 1 and Algorithm 1 in the Appendix, further clarifying the algorithmic process and implementation details of the proposed method.

3 Experimental Setup

Models and Baselines We evaluate our AdaGroPE on various LLMs: Llama-2 (7b and 13b) (Touvron et al., 2023b), Llama-3 (8b) (Dubey et al., 2024), Mistral (7b) (Jiang et al., 2023), SOLAR (10.7b) (Kim et al., 2024), and Phi-2 (Javaheripi et al., 2023). In addition, we compare AdaGropE's performance with the other two state-of-the-art training-free long-context extension methods, Dual Chunk Attention (DCA) (An et al., 2024b) and SelfExtend (Jin et al., 2024). Furthermore, several models fine-tuned to extend their context windows, *i.e.*, Longlora (Chen et al., 2024b), Together (Together, 2023), CodeLlama (Rozière et al., 2023), and CLEX (Chen et al., 2024a) are included for comparison to demonstrate the superiority of AdaGroPE. All usages of scientific artifacts in this paper obey the corresponding licenses stated in the original papers or websites.

Datasets Following An et al. (2024b) and Jin et al. (2024), we present our main results on language modeling tasks, synthetic long-context tasks, and real-world long-context tasks. For language modeling, we use the PG19 (Rae et al., 2020) dataset, with context lengths ranging from 4k to 32k tokens. In synthetic long-context tasks, we include the passkey retrieval task, as defined in Landmark Attention (Mohtashami and Jaggi, 2023), where a language model must retrieve an n-digit passkey embedded within a long, meaningless text sequence. The passkey is placed at different depths within the document and tested across context lengths from 8k to 64k tokens. For real-world long-context tasks, we evaluate AdaGroPE on the LongBench (Bai et al., 2024) benchmark and four closed-ended tasks from L-Eval (An et al., 2024a): TOFEL, QuALITY (cleaned from Pang et al. (2022)), Coursera, and SFiction, following the setup of An et al. (2024b).

4 Main Results

Performance on Language Modeling Tasks We compute perplexity (PPL) for different models on the test data, where a lower PPL indicates better performance of LLMs. Table 2 shows that AdaGroPE achieves state-of-the-art performance across nearly all context lengths. Notably, we compare training-free methods with fine-tuned models designed to extend their context windows, marked with an asterisk (*). The results further demonstrate that the training-free AdaGroPE surpasses these training-dependent methods, underscoring the effectiveness of the proposed approach.

Model	Evaluation Context Window			
	4096	8192	16384	32768
Llama-2-7b	**7.87**	>100	>100	>100
ChunkLlama-2-7b	**7.87**	7.67	7.64	7.89
SE-Llama-2-7b	**7.87**	7.67	7.58	7.71
AdaGroPE-Llama-2-7b	**7.87**	**7.65**	**7.56**	7.75
Longlora-7b-32k*	8.14	7.85	7.70	7.80
Together-7b-32k*	8.21	7.95	7.76	**7.64**
CodeLlama-7b-16k*	8.93	8.64	8.44	8.36
CLEX-7b-16k*	16.74	15.08	14.28	14.70
Llama-3-8b	**9.04**	**8.71**	78.88	>100
Chunk-Llama-3-8b	**9.04**	**8.71**	8.61	8.62
SE-Llama-3-8b	9.13	8.80	8.59	**8.52**
AdaGroPE-Llama-3-8b	**9.04**	**8.71**	**8.57**	**8.52**

Table 2: Perplexity (PPL) ↓ evaluation on PG19 (Rae et al., 2020) validation set. We highlight the best results for each model size in bold. Models marked with * indicate those fine-tuned to extend their context windows.

Performance on Synthetic Long Context Tasks Figure 2 displays the evaluation results for various methods on the passkey benchmark (Mohtashami and Jaggi, 2023). In our experiments, the passkey consists of 36 digits, and we conduct multiple retrieval tests for each combination of context length and depth. The passkey is randomly placed within a 400-token span. For example, with a context length of 8k and a depth of 0.1, the passkey appears between tokens 800 and 1600. Each span is evaluated over 10 iterations, yielding 20 iterations in this setting.

As shown in Figure 2, AdaGroPE, without any fine-tuning, achieves nearly 100% passkey retrieval accuracy across all tested depths and context lengths. In comparison, the original Mistral-7b-instruct-0.1 with SWA sees a drastic performance drop to 0 at smaller depths, while ChunkMistral-7b-ins-0.1 displays significant accuracy fluctuations as the token limit increases. Although SelfExtend achieves results similar to AdaGroPE, its performance degrades at larger token limits, such as 65,536, where AdaGroPE consistently maintains superior accuracy.

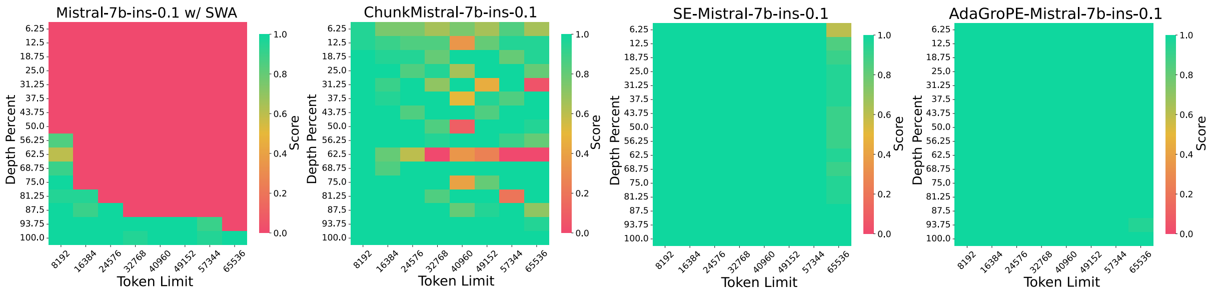

Figure 2: Passkey retrieval accuracy for Mistral-7b-instruct-0.1 with SWA, DCA, SelfExtend, or AdaGroPE. The number of passkey digits is set to 36. Mistral with AdaGroPE obtains nearly 100% passkey retrieval accuracy for all sequence lengths (token limits) and all depths.

Model	Tokens	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
		NarrativeQA	Qasper	MultiField-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PassageCount	PassageRe	Lcc	RepoBench-P	
GPT-3.5-Turbo-16k	16k	23.6	43.3	52.3	51.6	37.7	26.9	29.5	23.4	26.7	68.0	91.4	41.7	4.5	71.0	54.7	53.6	43.74
ChatGLM2-6B-32k	32k	21.1	31.5	46.2	45.1	34.0	21.9	32.4	24.0	26.5	62.5	78.7	36.3	1.5	77.0	55.6	49.9	40.26
Baichuan-13B-4k	16k	0.07	17.55	17.28	3.29	15	0.1	6.8	1.71	23.1	20.05	20.06	5.77	0.06	0.5	47.98	16.58	10.49
ALiBi-7B-4k	16k	0.04	8.13	17.87	2.73	8	1.33	5.31	1.64	25.55	9.25	8.83	4.67	0	1.27	46.69	18.54	9.48
Llama-2-7b-chat	4k	18.7	19.2	36.8	25.4	**32.8**	9.4	27.3	20.8	25.8	61.5	77.8	40.7	2.1	**9.8**	52.4	43.8	31.52
ChunkLlama-2-7b-chat	25k	20.27	25.80	34.87	23.18	28.42	10.15	27.03	21.27	**26.47**	68.50	74.86	41.51	1.48	4.75	58.05	50.76	32.34
SE-Llama-2-7b-chat	25k	**21.37**	26.68	34.63	35.47	30.46	15.51	27.51	21.30	25.87	68.50	78.79	41.29	**3.90**	3.50	**59.69**	**53.83**	34.26
AdaGroPE-Llama-2-7b-chat	25k	18.72	**29.61**	**40.20**	**37.33**	30.86	**15.73**	**28.39**	**21.45**	26.10	**69.50**	**83.23**	**42.16**	3.48	6.00	59.38	52.24	**35.27**
Llama-3-8b-ins	8k	21.63	**44.11**	44.35	46.84	35.84	21.53	29.98	22.66	27.75	75.50	90.58	42.67	6.50	66.50	56.81	51.24	42.78
ChunkLlama-3-8b-ins	25k	**26.54**	42.36	47.82	47.54	35.27	25.19	31.86	23.02	27.39	**77.00**	90.31	42.81	7.00	75.50	**58.49**	**55.15**	44.58
SE-Llama-3-8b-ins	25k	23.88	43.82	50.64	50.71	36.58	30.24	**32.90**	23.90	27.79	76.00	**91.68**	**42.93**	4.40	98.00	56.72	47.53	46.11
AdaGroPE-Llama-3-8b-ins	25k	25.72	44.09	**51.80**	**52.10**	**38.80**	**31.82**	32.81	**24.11**	**27.90**	76.50	91.13	42.30	**7.62**	**99.00**	56.49	51.46	**47.10**
Mistral-7b-ins-0.1 w/o SWA	8k	20.46	35.36	39.39	34.81	29.91	11.21	24.70	21.67	26.67	68.00	86.66	41.28	0.18	24.00	56.94	**55.85**	36.07
Mistral-7b-ins-0.1 w/ SWA	16k	19.40	34.53	37.06	42.29	32.49	14.87	27.38	22.75	26.82	65.00	**87.77**	42.34	1.41	28.50	**57.28**	53.44	37.08
ChunkMistral-7b-ins-0.1	16k	20.86	36.56	42.40	35.89	31.25	12.47	28.08	22.87	**27.09**	69.50	86.52	42.94	2.14	21.50	54.92	52.70	36.73
SE-Mistral-7b-ins-0.1	16k	23.56	**39.33**	49.50	45.28	34.92	23.14	30.71	**24.87**	26.83	69.50	86.47	**44.28**	1.18	29.50	55.32	53.44	39.86
AdaGroPE-Mistral-7b-ins-0.1	16k	**25.02**	39.00	**53.38**	**47.88**	**35.26**	**25.47**	**31.26**	23.84	26.67	**70.50**	86.66	43.86	**3.41**	**33.50**	55.05	51.50	**40.77**
SOLAR-10.7b-ins	4k	16.50	24.06	46.76	44.03	36.05	22.76	31.39	19.81	26.36	70.00	87.91	42.49	4.50	26.50	41.04	54.36	37.16
ChunkSOLAR-10.7b-ins	16k	22.48	29.77	**48.84**	51.62	34.80	27.35	31.59	21.75	26.22	**74.50**	87.41	42.69	**7.50**	20.00	48.98	54.94	39.40
SE-SOLAR-10.7b-ins	16k	22.63	32.49	47.88	46.19	34.32	27.88	30.75	22.10	25.62	**74.50**	89.04	42.79	4.00	28.00	53.73	56.47	39.90
AdaGroPE-SOLAR-10.7b-ins	16k	**24.35**	**34.42**	48.81	**53.31**	**43.30**	**33.93**	**32.38**	**22.29**	**26.51**	**74.50**	**89.62**	**43.11**	6.50	**36.00**	**54.32**	**58.64**	**42.62**
Phi-2	2k	4.46	7.01	19.98	9.43	8.55	4.62	25.64	14.32	24.03	50.50	74.55	**1.71**	**2.83**	**4.17**	**58.96**	54.14	22.81
SE-Phi-2	8k	12.04	**12.10**	20.15	8.22	9.68	3.89	27.90	14.58	22.13	**61.00**	**82.82**	1.40	2.37	2.83	57.87	**56.42**	24.71
AdaGroPE-Phi-2	8k	**14.14**	11.90	**26.80**	**9.96**	**11.37**	**5.09**	**29.68**	**20.04**	**25.19**	60.00	82.69	1.29	2.37	4.73	58.10	55.07	**26.15**

Table 3: Performance comparison of different LLMs on LongBench (Bai et al., 2024). Best and second-best results in each group are highlighted with bold and underline, respectively. The same applies below.

Performance on Real-World Long Context Tasks The evaluation results on LongBench and L-Eval are shown in Table 3 and Table 4, respectively. Following Jin et al. (2024), we present the performance of representative large language models (OpenAI, 2023a; Zeng et al., 2024; Baichuan, 2023) for reference, including those employing the ALiBi position encoding scheme (Press et al., 2022), on the LongBench benchmark. These results are reported by the LongBench (Bai et al., 2024) and CLEX (Chen et al., 2024a). As illustrated in Table 3, AdaGroPE significantly enhances the performance of the original models and outperforms other training-free extension methods, such as SelfExtend (Jin et al., 2024) and DCA (An et al., 2024b), achieving the best overall average performance. Table 4 further highlights AdaGroPE's superior performance and broad applicability. Notably, it demonstrates that AdaGroPE enables models with smaller initial context windows to exceed the performance of models with inherently larger context windows, which are pre-trained or fine-tuned for long-text understanding. For instance, AdaGroPE-Llama-2-7b-chat and AdaGroPE-Vicuna-2-1.5-7b, both based on models with 4k context windows, achieve better average performance than Longchat-1.5-7b-32k and Vicuna-1.5-7b-16k, respectively. Without relying on fine-tuning or additional training, AdaGroPE achieves competitive performance during inference, highlighting its potential as a resource-efficient approach for extending the context windows of existing LLMs.

5 Analysis

Performance as the Context Length Increases Figure 3 presents the performance of three different training-free methods for extending long texts

Model	TOFEL (3k∼5k)	QuALITY (4k∼9k)	Coursera (5k∼17k)	SFiction (6k∼27k)	Avg.
Llama-2-7b-chat	51.67	37.62	29.21	60.15	44.66
Longchat-1.5-7b-32k	39.77	37.62	32.99	57.02	41.85
ChunkLlama-2-7b-chat	57.62	35.14	32.12	61.72	46.65
SE-Llama-2-7b-chat	55.39	**41.09**	**35.76**	57.81	47.51
AdaGroPE-Llama-2-7b-chat	**61.34**	38.12	35.47	**64.06**	**49.28**
Llama-2-13b-chat	60.96	42.57	35.75	54.68	48.49
ChunkLlama-2-13b-chat	66.54	**43.06**	41.56	57.03	52.05
SE-Llama-2-13b-chat	66.17	41.09	38.95	**63.28**	52.37
AdaGroPE-Llama-2-13b-chat	**68.77**	40.59	**46.66**	57.81	**53.46**
Vicuna-1.5-7b-16k	55.39	39.60	38.66	60.15	48.45
SE-Vicuna-1.5-7b	55.39	**41.58**	37.21	**63.28**	49.37
AdaGroPE-Vicuna-1.5-7b	**56.51**	**41.58**	**42.01**	60.94	**50.26**
SOLAR-10.7b-ins	77.32	59.90	48.84	69.53	63.90
SE-SOLAR-10.7b-ins	79.18	**70.30**	50.44	**73.44**	68.34
AdaGroPE-SOLAR-10.7b-ins	**81.78**	68.81	**56.83**	71.88	**69.83**
Phi-2	55.76	**42.08**	38.37	52.34	47.14
SE-Phi-2	62.83	41.08	**42.44**	52.34	49.67
AdaGroPE-Phi-2	**68.40**	41.58	41.28	**55.47**	**51.68**

Table 4: Comparison with open-source chat models and proprietary models on 4 closed-ended tasks with various input lengths from L-Eval (An et al., 2024a).

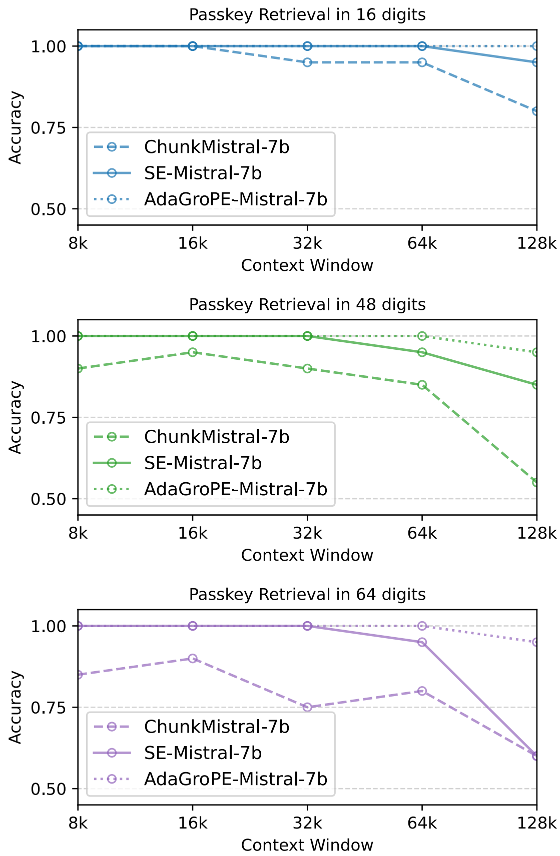

Figure 3: The performance of different training-free context window extension methods as the context length increases. AdaGroPE demonstrates robust performance in passkey retrieval as the input length increases, particularly when the number of digits in the passkey increases.

as the input length varies, specifically for passkey lengths of 16 digits, 48 digits, and 64 digits. Notably, AdaGroPE maintains an accuracy of over 90% as the context window lengthens, in contrast to ChunkMistral and SE-Mistral, whose performance exhibits significant degradation with increasing context window sizes. This degradation is particularly pronounced when the passkey is set to 64 digits, where the accuracy of ChunkMistral and SE-Mistral declines from approximately 100% and 80% at an 8k context window to around 60%. The comparison results suggest that AdaGroPE demonstrates more robust performance when handling longer context windows, especially in more challenging tasks.

Ablation Studies on the Selection of P and r Figure 4 presents the results of the ablation study on the selection of P and r. Following An et al. (2024b), we conduct experiments on two real-world datasets from Longbench: Qasper and Lcc.

As shown in Figure 4, the selection of P has a more significant impact on the performance of AdaGroPE, with the optimal P varying across different tasks. This reflects a trade-off between leveraging more comprehensive positional encodings and the degree of pre-training on these encodings. On one hand, larger P values allow for better utilization of the relative positional encodings learned during

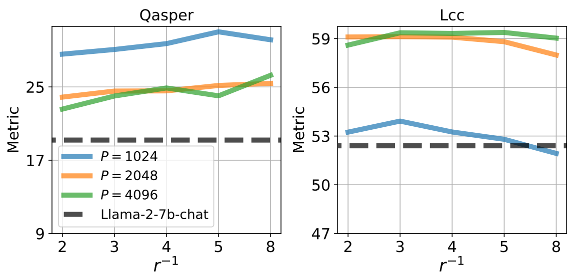

Figure 4: Ablation study results on the selection of P and r. The selection of P has a more significant impact on the method's performance compared to that of r.

pre-training, resulting in more accurate relative position representation. On the other hand, as the relative position range extends, the lack of sufficient pre-training for larger positions can lead to degradation in performance. Notably, Lcc, a code-based dataset, is sensitive to the relative positions of nearby tokens and tends to benefit from more precise encodings. This may explain why larger

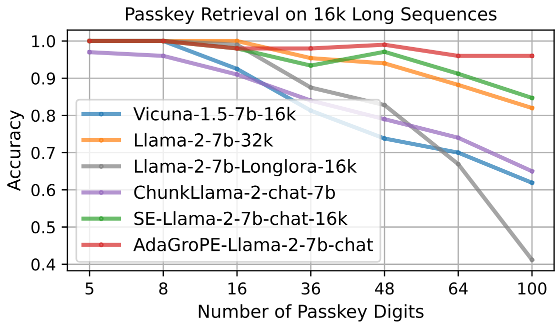

Figure 5: Passkey retrieval accuracy for fine-tuned long-context models and training-free context extension methods on Llama-2-chat-7b.

values of P and w_n generally lead to improved performance, as reflected by the green and orange lines outperforming the blue line in the right figure. Additionally, we observe that the optimal r varies across datasets. We find that our default setting $r = 0.25$ consistently yields strong average performance and provides reliable improvements over the Llama-2-7b-chat baseline.

Varying-Length Passkey Retrieval Task We validate AdaGroPE's capability to capture information in long contexts by conducting experiments on the passkey retrieval task, with passkey digit lengths set to 5, 8, 16, 36, 48, 64, and 100, respectively. As the number of digits increases, the task complexity also increases accordingly.

As shown in Figure 5, the performance of ChunkLlama, Longlora, and Vicuna declines significantly as the number of passkey digits increases, especially beyond 8 digits. Despite Vicuna and Longlora being fine-tuned for long-context windows, they still struggle with more difficult passkey retrieval tasks that demand higher precision. In contrast, while all methods exhibit some performance degradation, AdaGroPE shows a notably milder decline and maintains relatively robust overall results. These findings suggest the potential of AdaGroPE as an effective training-free alternative for long-context modeling, while also highlighting the challenges that fine-tuning-based methods may face in accurately capturing information across extended sequences. The performance of the varying-length passkey retrieval task on longer sequences is provided in the Appendix.

6 Conclusion

In this paper, we propose a novel long-context window extension method, AdaGroPE, which can be applied to existing LLMs with short-context windows in a training-free, plug-and-play manner. AdaGroPE employs a progressively reused relative position encoding strategy, adhering to three key principles when constructing the relative position matrix: minimizing reuse, prioritizing reuse of distant relative positions, and progressively increasing reuse count from nearby to distant positions. This adaptive approach allows the relative position matrix to be tailored to the target context window length. We demonstrate the effectiveness and superiority of AdaGroPE across language modeling tasks, synthetic long-context tasks, and real-world long-context tasks, and further validate its robustness under increasing context lengths and task complexity.

Limitations

The proposed AdaGroPE is an empirically validated method for improving long-context processing in large models. While its effectiveness has been demonstrated through extensive experiments, a more thorough theoretical analysis of the underlying principles behind positional encoding in large language model attention mechanisms is not included. We believe future work should delve deeper into the intrinsic mechanisms of transformer positional encodings to develop novel approaches for enhancing long-text understanding in large language models. Additionally, our current exploration focuses solely on extending long-sequence capabilities in single-modal settings. Further investigation and validation are needed for multimodal approaches that integrate modalities such as images, videos, and audio.

Acknowledgment

This work was supported by the Beijing Natural Science Foundation (No. L247026) and the National Natural Science Foundation of China (Nos. 62021002, 62525103, 62441235, 62271281). It was also sponsored by the CAAI-CANN Open Fund, developed on OpenI Community.

References

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and

Xipeng Qiu. 2024a. L-eval: Instituting standardized evaluation for long context language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 14388–14411. Association for Computational Linguistics.

Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. 2024b. Training-free long-context scaling of large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

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. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3119–3137. Association for Computational Linguistics.

Baichuan. 2023. A 13b large language model developed by baichuan intelligent technology.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Guanzheng Chen, Xin Li, Zaiqiao Meng, Shangsong Liang, and Lidong Bing. 2024a. CLEX: continuous length extrapolation for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *CoRR*, abs/2306.15595.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024b. Longlora: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jishnu Ray Chowdhury and Cornelia Caragea. 2023. Monotonic location attention for length generalization. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 28792–28808. PMLR.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Chaochen Gao, Xing Wu, Zijia Lin, Debing Zhang, and Songlin Hu. 2025. Nextlong: Toward effective long-context training without long documents. *CoRR*, abs/2501.12766.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3991–4008. Association for Computational Linguistics.

Wei Huang, Yizhe Xiong, Xin Ye, Zhijie Deng, Hui Chen, Zijia Lin, and Guiguang Ding. 2025. Fast quiet-star: Thinking without thought tokens. *CoRR*, abs/2505.17746.

Maor Ivgi, Uri Shaham, and Jonathan Berant. 2023. Efficient long-text understanding with short-text models. *Trans. Assoc. Comput. Linguistics*, 11:284–299.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, Suriya Gunasekar, Mojan Javaheripi, Piero Kauffmann, Yin Tat Lee, Yuanzhi Li, Anh Nguyen, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Michael Santacroce, Harkirat Singh Behl, Adam Taumann Kalai, Xin Wang, Rachel Ward, Philipp Witte, Cyril Zhang, and Yi Zhang. 2023. Phi-2: The surprising power of small language models.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. LLM maybe longlm: Selfextend LLM context window without tuning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Sanghoon Kim, Dahyun Kim, Chanjun Park, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, Changbae Ahn, Seonghoon Yang, Sukyung Lee, Hyunbyung Park, Gyoungjin Gim, Mikyoung Cha, Hwalsuk Lee, and Sunghun Kim. 2024. SOLAR 10.7b: Scaling large language models with simple yet effective depth upscaling. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track, NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 23–35. Association for Computational Linguistics.

Gibbeum Lee, Volker Hartmann, Jongho Park, Dimitris Papailiopoulos, and Kangwook Lee. 2023. Prompted llms as chatbot modules for long open-domain conversation. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4536–4554. Association for Computational Linguistics.

Haoran Lian, Junmin Chen, Wei Huang, Yizhe Xiong, Wenping Hu, Guiguang Ding, Hui Chen, Jianwei Niu, Zijia Lin, Fuzheng Zhang, and Di Zhang. 2025. Breaking the stage barrier: A novel single-stage approach to long context extension for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 4897–4909. Association for Computational Linguistics.

LocalLLaMA. 2023. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. *CoRR*, abs/2305.16300.

OpenAI. 2023a. Gpt-3.5-turbo. `https://platform.openai.com/docs/models/gpt-3-5`. Accessed: 2025-05-31.

OpenAI. 2023b. GPT-4 technical report. *CoRR*, abs/2303.08774.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. 2022. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5336–5358. Association for Computational Linguistics.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon LLM: outperforming curated corpora with web data only. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950.

Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. 2023. Randomized positional encodings boost length generalization of transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1889–1903. Association for Computational Linguistics.

Leqi Shen, Tao He, Guoqiang Gong, Fan Yang, Yifeng Zhang, Pengzhang Liu, Sicheng Zhao, and Guiguang Ding. 2025. Llava-mlb: Mitigating and leveraging attention bias for training-free video llms. *CoRR*, abs/2503.11205.

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

Together. 2023. Llama-2-7b-32k-instruct — and fine-tuning for llama-2 models with together api.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

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, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Ao Wang, Hui Chen, Jianchao Tan, Kefeng Zhang, Xunliang Cai, Zijia Lin, Jungong Han, and Guiguang Ding. 2024. Prefixkv: Adaptive prefix KV cache is what vision instruction-following models need for efficient generation. *CoRR*, abs/2412.03409.

Jing Wei, Sungdong Kim, Hyunhoon Jung, and Young-Ho Kim. 2024. Leveraging large language models to power chatbots for collecting user self-reported data. *Proc. ACM Hum. Comput. Interact.*, 8(CSCW1):1–35.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yizhe Xiong, Wei Huang, Xin Ye, Hui Chen, Zijia Lin, Haoran Lian, Zhenpeng Su, Jungong Han, and Guiguang Ding. 2025. Uniattn: Reducing inference costs via softmax unification for post-training llms. *CoRR*, abs/2502.00439.

Xinhao Xu, Hui Chen, Zijia Lin, Jungong Han, Lixing Gong, Guoxin Wang, Yongjun Bao, and Guiguang Ding. 2024. Tad: A plug-and-play task-aware decoding method to better adapt llms on downstream tasks. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 6587–6596. ijcai.org.

Xinhao Xu, Hui Chen, Mengyao Lyu, Sicheng Zhao, Yizhe Xiong, Zijia Lin, Jungong Han, and Guiguang Ding. 2025. Mitigating hallucinations in multi-modal large language models via image token attention-guided decoding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 1571–1590. Association for Computational Linguistics.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from GLM-130B to GLM-4 all tools. *CoRR*, abs/2406.12793.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

A Related Work

A.1 Positional Encoding

Position information is crucial for transformer models (Vaswani et al., 2017; Xiong et al., 2025; Wang et al., 2024; Xu et al., 2025) and is commonly represented using either absolute or relative embeddings. Absolute position embeddings assign a vector based on a token's position within the sequence, as seen in both sinusoidal and learned embeddings, such as those used in GPT-3 (Brown et al., 2020) and OPT (Zhang et al., 2022). In contrast, relative positional encodings, which have become widely adopted, capture token distances relative to one another, enhancing contextual understanding, particularly in long-context scenarios. Notable approaches include RoPE (Su et al., 2024) and ALiBi (Press et al., 2022), which have been incorporated into prominent models like Llama (Touvron et al., 2023a) and Falcon (Penedo et al., 2023). Our method builds upon the RoPE framework, aiming to optimize positional encoding for more effective long-context modeling.

A.2 Extrapolation of RoPE

Research has shown that directly extrapolating RoPE leads to significant performance degradation in long-context tasks (Chen et al., 2023; Chowdhury and Caragea, 2023; Chen et al., 2024a), primarily due to the model encountering unseen relative positions during pre-training (Jin et al., 2024). To address this, recent approaches have focused on training techniques that enhance the long-context understanding of LLMs after extrapolation (Rozière et al., 2023; Together, 2023; Lian et al., 2025). In addition, some studies explore training-free methods that ensure relative positions remain within the scope of the observed context length (An et al., 2024b; Jin et al., 2024; Chen et al., 2023; LocalLLaMA, 2023), thus reusing position embeddings and mitigating extrapolation-related degradation. However, these methods often fail to account for variations in relative positions and struggle to adapt dynamically to changing input lengths. In contrast, AdaGroPE builds on these prior techniques, offering a training-free, plug-and-play solution that distinguishes itself through the progressive reuse of relative positions and its dynamic adaptability to input length variations.

Model	P	TOFEL (3k∼5k)	QuALITY (4k∼9k)	Coursera (5k∼17k)	SFiction (6k∼27k)
Llama-2-7b-chat	N/A	51.67	37.62	29.21	60.15
	1k	**61.34**	36.63	**37.50**	60.15
+AdaGroPE	2k	56.13	**38.12**	35.47	**64.06**
	4k	53.53	37.62	36.05	61.03

Table 5: Impact of P on the effectiveness of AdaGroPE across tasks with different context lengths.

B Implementation Details

All the experiments in the paper are conducted on a single NVIDIA H800 GPU. Unless otherwise specified, all experimental results in the paper are based on the default setting of $r = 0.25$. We find that this default setting is generally applicable across different tasks. Furthermore, as noted in Section 2.2, w_n can be parameterized by P and r, with w_n set to $0.25P$ in all experiments by default.

For the selection of P, we set $P = w$ on language modeling tasks, while for the other tasks, we generally set $P = w/2$, where w is the pre-training context window size of the model. As highlighted in prior work (Jin et al., 2024), positional encodings with smaller relative distances are more effectively trained. Therefore, extrapolation based on smaller relative positions tends to yield better performance. Besides, for tasks where the input context length is relatively short (*e.g.*, close to or less than the pre-training window size), we observe that limiting P to a smaller range, such as $w/4$, leads to better results. Taking the experiments on L-Eval in Table 4 as an example, we set $P = w/4$ for the TOEFL (3k–5k) task and $P = w/2$ for the other three tasks: QuALITY (4k-9k), Coursera (5k-17k), and SFiction (6k-27k). Table 5 illustrates the effects of applying AdaGroPE with different P values on Llama-2-7b-chat (pre-training window size $w =$ 4k).

We observe that the proposed guideline for setting P achieves optimal average performance and we report only the results obtained using the P values selected according to this guideline. Although a more refined selection of P for specific tasks may yield better results in some cases (*e.g.*, in the Coursera task, AdaGroPE performs better when $P = w/4 =$ 1k), we believe that following the proposed guideline ensures the effectiveness of AdaGroPE.

Algorithm 1 provides the pseudocode for relative position computation in AdaGroPE during decod-

Algorithm 1 Pseudocode for relative position computation in AdaGroPE during decoding

Input: L ▷ Target context length after extension
P ▷ Number of relative positions used
r ▷ Reuse ratio coefficient
Output: m_a ▷ Relative position in AdaGroPE

function ADAGROPERELPOS(L, P, r)
  if $L \leq P$ **then**
    $m_a \leftarrow [L{-}1, L{-}2, \ldots, 0]$
    return m_a
  end if
  $G_s^m \leftarrow 1$
  $L_n^{\max} \leftarrow P$
  $r_{\text{sum}}, l_{\text{sum}} \leftarrow 0$
  $m_a \leftarrow [\,]$
  while $L_n^{\max} < L$ **do**
    if isPowerOfTwo(G_s^m) **then**
      $r_n \leftarrow \lfloor r \times P / G_s^m \rfloor$
      for $p = r_{\text{sum}}$ to $r_{\text{sum}} + r_n - 1$ **do**
        prepend p to m_a for G_s^m times
      end for
      $r_{\text{sum}} \leftarrow r_{\text{sum}} + r_n$
      $l_{\text{sum}} \leftarrow l_{\text{sum}} + r_n \times G_s^m$
    end if
    $G_s^m \leftarrow G_s^m + 1$
    $L_n^{\max} \leftarrow (P - r_{\text{sum}}) \times G_s^m + l_{\text{sum}}$
  end while
  $d_{n-1} \leftarrow P - (L_n^{\max} - L) - r_{\text{sum}}$
  for $p = r_{\text{sum}}$ to $P - d_{n-1} - 1$ **do**
    prepend p to m_a for $G_s^m - 1$ times
  end for
  for $p = P - d_{n-1}$ to $P - 1$ **do**
    prepend p to m_a for G_s^m times
  end for
  return m_a
end function

Model	Latency (s/token)			
	32k	64k	96k	128k
Llama-2-7b-chat	1.81	5.30	10.62	17.77
AdaGroPE-Llama-2-7b-chat	2.10	5.80	11.24	18.48

Model	Memory (MB)			
	32k	64k	96k	128k
Llama-2-7b-chat	28787	43925	59063	74193
AdaGroPE-Llama-2-7b-chat	29543	45433	61323	77201

Table 6: Comparison of latency and memory consumption across context lengths between the original model and our AdaGroPE.

ing. Here, m_a denotes the relative position with respect to the query token, while l_{sum} and r_{sum} denote, respectively, the total number of true relative positions that have been mapped to computed relative positions in AdaGroPE, and the total number of relative positions used that correspond to these mapped true relative positions. Specifically, r_{sum} corresponds to the summation of the terms defined by Eq. (9). The implementation proceeds by first iteratively determining the maximum reuse count G_s^m, during which relative positions with reuse counts less than G_s^m are retained according to Eq. (9). Finally, based on the target extension length L, the most distant relative positions reused $G_s^m{-}1$ and G_s^m times are added to the resulting relative position sequence.

C Latency and Memory Analysis

We also conduct comparisons on the passkey retrieval task across different context length settings, evaluating the token generation latency (s/token) and GPU memory consumption (MB) for the original model and our proposed method. The results are shown in Table 6.

We observe that AdaGroPE increases inference latency and memory usage by no more than 10% on average compared to the original model. This demonstrates the practical applicability of the proposed method in real-world scenarios.

D Limitations in Challenging Tasks

Despite the overall effectiveness of our method, we observe certain failure cases that reveal its current limitations. As shown in Table 3, AdaGroPE performs suboptimally on code tasks on average. We believe this is due to the distinct characteristics of code compared to natural language. Specifically, the assumption that distant relative positions can be less precise than closer ones, which generally holds for natural language, does not strictly apply to code. Code understanding requires a higher degree of accuracy in relative distances between tokens, as it is significantly influenced by the structural semantic relationships and hierarchical organization between tokens. These factors cannot be inferred solely from relative distance.

Beyond this, by examining the outputs of QA tasks in the LongBench benchmark, we observe that AdaGroPE's performance tends to decline on questions requiring reasoning, inference, or complex information integration across long contexts.

Model	Position Emb	Training context	Context Window			
			32k	64k	96k	128k
CodeLlama 7b	NTK	16k	8.36	8.65	9.14	9.87
+AdaGroPE	NTK	16k	**8.34**	**8.32**	**8.38**	**8.48**
Together 7b	PI	32k	**7.64**	>100	>100	>100
+AdaGroPE	PI	32k	**7.64**	**7.58**	**7.58**	**7.60**

Table 7: Integration of AdaGroPE with other long context window extension methods.

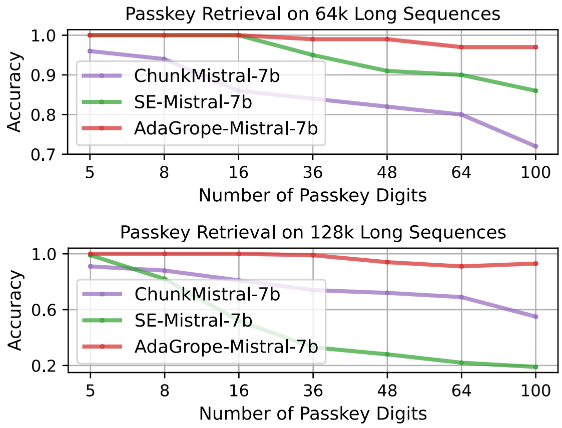

Figure 6: Passkey retrieval accuracy with longer sequence lengths on Mistral-7b.

Specifically, we find that the model's responses to "Why" questions are often less satisfactory compared to "What" or "Where" questions, which are easier to locate answers directly. Although AdaGroPE enables the model to handle and understand contexts beyond the pre-training window, it seems to not fully resolve the original model's limitation in precise reasoning and analysis of long-distance information.

These failure cases reveal opportunities for improving the method, such as enabling finer-grained control over position reuse or integrating auxiliary mechanisms to better handle extremely long contexts. We leave these directions for future investigation.

E Additional Experimental Results

E.1 Performance of Integrating AdaGroPE with Other Long-Context Window Extension Methods

Table 7 presents the performance of AdaGroPE applied to Codellama (Rozière et al., 2023) and Together (Together, 2023). Codellama and Together expand their context windows to 16k and 32k, respectively, using NTK (LocalLLaMA, 2023) and PI (Chen et al., 2023) strategies. The table shows the PPL results (Rae et al., 2020), demonstrating that AdaGroPE can effectively integrate with existing long-context expansion methods, further enhancing the language modeling capabilities of models with already large context windows.

Model	MultiField-en	2WikiMQA	GovReport
SE-Phi-2	26.33	11.33	27.99
AdaGroPE-Phi-2	**27.79**	**12.49**	**29.68**
Model	**TrivialQA**	**PassageCount**	**RepoBench-P**
SE-Phi-2	83.14	2.12	52.82
AdaGroPE-Phi-2	**84.71**	**3.09**	**53.63**

Table 8: Evaluation results on LongBench (Bai et al., 2024) conducted on Ascend 910 GPUs.

E.2 Performance of AdaGroPE on Varying-Length Passkey Retrieval Task with Longer Sequence Lengths

To further evaluate the performance of AdaGroPE and baseline methods, we extended the input sequence length to 64k and 128k. As shown in Figure 6, AdaGroPE exhibits a noticeably slower degradation with increasing numbers of passkey digits at the longer 64k and 128k input sequence lengths, outperforming baseline methods. This highlights the dual advantages of the proposed method in handling both longer input sequences and larger numbers of passkey digits.

E.3 Evaluation on Ascend 910 GPUs

We validate the effectiveness of the AdaGroPE on Ascend 910 GPUs, as presented in Table 8. AdaGroPE outperforms the baseline on datasets spanning diverse sub-tasks.