LaMPE: Length-aware Multi-grained Positional Encoding for Adaptive Long-context Scaling Without Training

Sikui Zhang[1,2,3], **Guangze Gao**[1,2], **Ziyun Gan**[1,2], **Chunfeng Yuan**[1,2*],
Zefeng Lin[1,2], **Houwen Peng**[4], **Bing Li**[1,2], **Weiming Hu**[1,2]
[1]Beijing Key Laboratory of Super Intelligent Security of Multi-Modal Information, CASIA
[2]State Key Laboratory of Multimodal Artificial Intelligence Systems, CASIA
[3]School of Artificial Intelligence, University of Chinese Academy of Sciences [4]Tencent

Abstract

Large language models (LLMs) experience significant performance degradation when the input exceeds the pretraining context window, primarily due to the out-of-distribution (OOD) behavior of Rotary Position Embedding (RoPE). Recent studies mitigate this problem by remapping OOD positions into the in-distribution range with fixed mapping strategies, ignoring the dynamic relationship between input length and the model's effective context window. To this end, we propose **L**ength-**a**ware **M**ulti-grained **P**ositional **E**ncoding (LaMPE), a training-free method that fully utilizes the model's effective context window for adaptive long-context scaling in LLMs. Motivated by the left-skewed frequency distribution of relative positions, LaMPE establishes a dynamic relationship between mapping length and input length through a parametric scaled sigmoid function to adaptively allocate positional capacity across varying input lengths. Meanwhile, LaMPE devises a novel multi-grained attention mechanism that strategically allocates positional resolution across different sequence regions to capture both fine-grained locality and long-range dependencies. Our method can be seamlessly applied to a wide range of RoPE-based LLMs without training. Extensive experiments on three representative LLMs across five mainstream long-context benchmarks demonstrate that LaMPE achieves significant performance improvements compared to existing length extrapolation methods. Code is avaliable in https://github.com/scar-on/LaMPE.

1 Introduction

Rotary Position Embedding (RoPE) has become a cornerstone technique in large language models (LLMs) due to its excellent performance (Su et al., 2024; Touvron et al., 2023; Yang et al., 2024; Dubey et al., 2024). However, RoPE-based LLMs are typically limited by the context window formed during pretraining and post-training stages. When the input length exceeds the context window, RoPE encounters out-of-distribution (OOD) relative positions, as larger relative positions were not seen during the training phase (Han et al., 2024; Su, 2023a). Consequently, this often leads to significant degradation in model performance, limiting the model's applicability in real-world tasks, such as repository-level code completion (Guo et al., 2024), long-document question-answering (Li et al., 2024), and summarization (Pratapa and Mitamura, 2025).

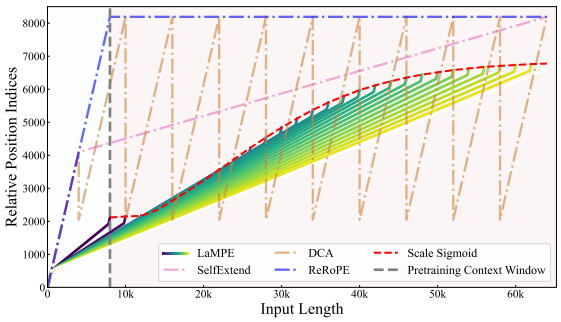


Figure 1: Comparison of different mapping strategies generated by LaMPE and other extrapolation methods, including ReRoPE (Su, 2023b), SelfExtend (Jin et al., 2024a), and DCA (An et al., 2024b).

Existing RoPE-based methods for LLM extrapolation can be broadly categorized into two types: base-modified and position indices modified. The first type focuses on modifying the base (bloc97, 2023; Peng et al., 2024; Chen et al., 2023; Rozière et al., 2024) to extend the context window. However, these methods typically require additional fine-tuning on longer texts and have explicit extrapolation upper bounds. In contrast, the second type that we focus on in this work achieves training-free length extrapolation, including ReRoPE (Su, 2023b), SelfExtend (Jin et al., 2024a), and DCA (An et al., 2024b). The core idea is to remap OOD relative positions into the pretraining context window with fixed mapping

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 32134–32149

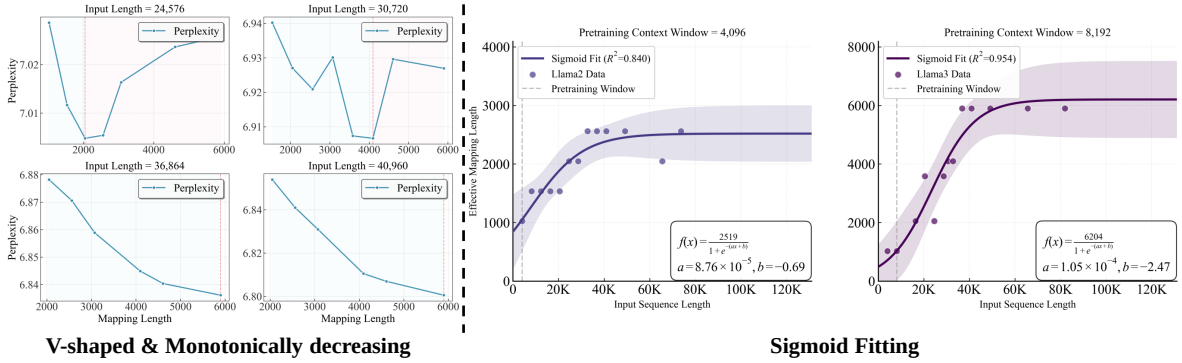


Figure 2: **Left:** Visualization of the variation in perplexity with respect to mapping length. **Right:** Sigmoid fit of the relationship between input length and optimal mapping length for Llama2 and Llama3.

strategies.

As shown in Figure 1, although these studies have shown promising results in mitigating the OOD issue compared to the first type of methods, they still face two major challenges corresponding to the peak point and slope of the curve respectively. (1) Overlooking Skewed Training Distribution: Such fixed mappings inherently overlook the left-skewed frequency distribution of relative positions (An et al., 2025) formed during the training phase. Specifically, they treat all positions within the context window as equally well-trained, while in reality, long-range positions are severely undertrained compared to short-range ones. (2) Inflexible Static Grouping: Fixed group sizes are used for position mapping, regardless of varying input lengths. This lack of adaptability prevents the optimal utilization of well-trained short-range positions, as the same grouping granularity cannot be optimal for both short and long sequences. These limitations highlight the need for a more flexible strategy that can adaptively set the optimal mapping length according to the input length. Thus, a key question arises: *How can we adaptively set the optimal mapping length and group size based on input length?*

To this end, we propose **L**ength-**a**ware **M**ulti-grained **P**ositional **E**ncoding (LaMPE), a training-free method that fully utilizes the model's effective context window for length extrapolation. LaMPE employs a parametric scaled sigmoid function to adjust the effective position mapping, enabling adaptive allocation of positional capacity across varying input lengths. This design is motivated by our empirical observation that the relationship between input length and optimal mapping length follows an S-shaped trend, as shown in Figure 2(Right). Specifically, the optimal mapping length increases slowly at first, accelerates in the middle, and then saturates. Moreover, drawing inspiration from the critical roles of neighboring tokens and initial tokens (Peysakhovich and Lerer, 2023; Liu et al., 2024; Xiao et al., 2024), we propose a multi-grained mechanism to ensure that current tokens maintain fine-grained positional information with both neighboring tokens and initial tokens. Owing to its training-free and plug-and-play nature, LaMPE can be seamlessly applied to any RoPE-based LLM.

We comprehensively evaluate LaMPE on a wide range of long-context benchmarks, including LongBench (Bai et al., 2024), L-Eval (An et al., 2024a), ∞Bench (Zhang et al., 2024), RULER (Hsieh et al., 2024) and PG-19 (Rae et al., 2020). Our experiments cover LLMs with context windows ranging from 4K to 128K, including Llama2-7B-Chat (Touvron et al., 2023), Llama3-8B-Instruct (Meta, 2024), and Llama3.1-8B-Instruct (Dubey et al., 2024). The experimental results demonstrate that LaMPE consistently outperforms existing extrapolation methods both at extrapolated lengths and within the pretraining context window. Our main contributions are summarized as follows:

- We propose LaMPE, a training-free method that incorporates a dynamic mapping strategy and multi-grained attention mechanism for adaptive long-context scaling.
- We explore perplexity variations across different mapping lengths, identifying V-shaped and monotonically decreasing patterns that guide position remapping.
- Comprehensive evaluation demonstrates that LaMPE outperforms existing extrapolation

methods on both real-world and synthetic long-context tasks.

2 Background

2.1 Rotary Position Embedding (RoPE)

RoPE was first proposed in RoFormer (Su et al., 2024), where it incorporates relative positional information by applying a phase rotation to the Query and Key in attention computation. For the token embedding at position i, denoted as x_i, we use f to represent the operation that injects positional information into x_i:

$$f(x_i, i, \theta) = R_i^\theta W^T x_i, \tag{1}$$

where W denotes the learnable projection matrix of a linear layer, R_i^θ represents the rotary positional encoding matrix with the Ptolemy's identity $R_{j-i}^\theta = (R_i^\theta)^T R_j^\theta$, where θ represents frequency basis. So we define the dot product between the Query and Key at positions i and j as:

$$\begin{aligned} q_i^T k_j &= f_q(x_i, i, \theta)^T f_k(x_j, j, \theta), && (2)\\ &= (R_i^\theta W_q^T x_i)^T (R_j^\theta W_k^T x_j), && (3)\\ &= (W_q^T x_i)^T R_{j-i}^\theta (W_k^T x_j). && (4) \end{aligned}$$

During the attention computation, the positional information of the relative position $j - i$ can be incorporated by assigning the positional information of i and j to the Query and Key, respectively.

2.2 Extrapolation of RoPE

Previous work has focused on extending the context window by adjusting the rotary matrix θ in the function $f(x_i, i, \theta)$, including NTK-RoPE (bloc97, 2023) and YaRN (Peng et al., 2024). In contrast, another line of research focuses on adjusting position indices i to remap out-of-distribution relative positions into the in-distribution range. For example, SelfExtend, for each token, selects the w nearest tokens within a local window. Tokens outside this window are partitioned into multiple groups, each containing G tokens, where all tokens in a group share the same position index. Given a fixed group size G and local window size w, and a pretraining context window of size n, the model's context window is extended to $(n - w) \times G + w$.

3 Method

In this section, we present the details of Length-aware Multi-grained Positional Encoding (LaMPE). LaMPE comprises two key components: (1) a Length-aware Mapping Strategy that establishes the relationship between input length and optimal mapping length through a parametric scaled sigmoid function, and (2) a Multi-grained Attention Mechanism that partitions the input sequence into three distinct regions, each with tailored positional encoding granularity to jointly capture both locality and long-range dependencies.

3.1 Length-aware Dynamic Mapping Strategy

Existing methods typically adopt fixed mapping strategies that fail to generalize across inputs of varying lengths. To enable dynamic mapping, we first investigate the relationship between input length and optimal mapping length through empirical analysis.

Details. We adopt the widely used perplexity (PPL) as our evaluation metric and conduct experiments on the PG-19 dataset (Rae et al., 2020) using Llama2-7B-Chat and Llama3-8B-Instruct with pretraining context windows of 4K and 8K tokens respectively, where K denotes 1024. We systematically vary input length and mapping length to examine how perplexity changes under different configurations. Specifically, the input length is progressively increased from 4K to 64K. For each input length, the mapping length is varied from small values up to the pretraining context window.

Observations. Our analysis reveals two key observations about how perplexity varies with mapping length. As shown in Figure 2(Left): (1) **V-shaped** pattern for short inputs: When the input length is small, perplexity exhibits a V-shaped curve as mapping length increases. Starting from relatively high perplexity at small mapping lengths (e.g., 1/8 of the pretraining context window), perplexity gradually decreases to reach a minimum before rising again. This indicates a trade-off where both overly small and large mapping lengths lead to suboptimal performance. (2) **Monotonically decreasing** for long inputs: Once the input length exceeds a certain threshold, the V-shaped pattern disappears and perplexity consistently decreases with increasing mapping length, up to the pretraining context window size. These observations indicate that an optimal mapping length exists and varies depending on the input length.

Dynamic Mapping. Based on the above observations, we devise a length-aware dynamic mapping

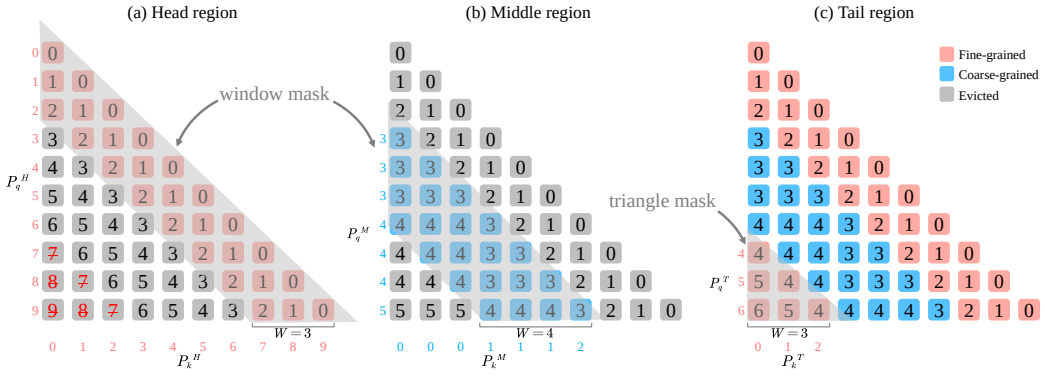


Figure 3: Illustration of LaMPE's Relative Position Matrix PE for a sequence of length $l = 10$ and pretraining context window $n = 7$. (a) head region with window size $s_1 = 3$. (b) middle region maps out-of-distribution relative positions ($[7, 8, 9]$) into the pretraining context window via the linear normalization. (c) tail region replaces the lower-left triangle part of the linear normalization with a window size of $s_2 = 3$.

strategy to adaptively determine the optimal mapping length using a scaled sigmoid function. As shown in Figure 2(Right), the optimal mapping length exhibits an S-shaped growth pattern that can be effectively modeled by a scaled sigmoid function. This scaled sigmoid function can be expressed as follows:

$$m = \frac{L}{1 + e^{-(al+b)}}, \tag{5}$$

where L denotes the maximum value of the curve, l represents the input length, and m corresponds to the optimal mapping length. This sigmoid function provides a smooth and continuous approximation of the optimal mapping length across a wide range of input lengths. The parameters a and b can be estimated via simple curve fitting based on a small number of points, making this method highly practical for real-world applications. Additional observation results and fitting details can be found in Appendix B.

3.2 Multi-grained Attention Mechanism

To address the limitations of fixed projected grouping strategies, we propose a multi-grained attention mechanism that strategically allocates positional resolution across different sequence regions based on the obtained optimal mapping length from the previous section. As illustrated in Figure 3, our method partitions the input sequence into three distinct regions, each with tailored positional encoding granularity. The following sections detail the adaptive grouping and its integration with attention computation.

Adaptive Grouping. Unlike fixed grouping methods, our method dynamically adapts the relative position matrix $PE \in \mathbb{Z}^{l \times l}$ based on input length, where each entry $PE[i][j]$ remaps the original relative position $(i - j)$ to a new relative position. Given input sequence length l and optimal mapping length m, the updated relative position matrix is defined as:

$$PE[i][j] = \left\lfloor \frac{m}{l}(i - j) \right\rfloor, \tag{6}$$

where $\lfloor \cdot \rfloor$ denotes the floor operation. As shown in Figure 3(b), our method dynamically adapts the relative position matrix based on the input length, where the compression ratio ($\frac{m}{l}$) determines how original relative positions are remapped. This linear transformation effectively maps the original position range $[0, l - 1]$ to a compressed range $[0, m - 1]$, enabling adaptive granularity control based on sequence length.

Restoring Locality and Long-Range Dependencies. In LLMs, key instructions or questions often appear at the beginning or end of the prompt. However, only applying Eq. 6 may disrupt the model's ability to capture local relationships and long-range dependencies because it alters the relative positions between neighboring and initial tokens (Jin et al., 2024a; Peysakhovich and Lerer, 2023; Xiao et al., 2024). Since these tokens are crucial for continuous generation, we introduce hyperparameters s_1 and s_2 to control the widths of fine-grained regions at the head and tail, respectively. The final relative position matrix is defined as:

$$PE[i][j] = \begin{cases} i-j, & i-j \le s_1 \\ \left\lfloor \frac{m-s_1-s_2}{l-s_1-s_2}(i-j-s_1)+s_1 \right\rfloor, & s_1 < i-j < l-s_2 \\ m-l+i-j, & i-j \ge l-s_2. \end{cases} \tag{7}$$

The design rationale for each region reflects different modeling priorities: The head region maintains fine-grained locality between neighboring tokens, which is crucial for continuous generation of next tokens. The middle region employs linear transformation with floor operation to preserve global relative ordering while supporting arbitrary input lengths without favoring specific positions. The tail region preserves fine-grained relationships for distant tokens, considering that key instructions or questions often appear at sequence boundaries.

Attention Computation with Dynamic Mapping. We implement our dynamic mapping strategy within the RoPE-based LLM by defining region-specific position indices $P_q[i]$ and $P_k[j]$ that determine rotation matrices during attention computation:

Head region ($i-j \le s_1$): Standard relative positions for fine-grained local interactions:

$$P_q^{(H)}[i] = i, \quad P_k^{(H)}[j] = j. \tag{8}$$

Middle region ($s_1 < i-j < l-s_2$): Compressed group position indices via linear transformation:

$$\begin{aligned} P_q^{(M)}[i] &= \left\lfloor \frac{m-s_1-s_2}{l-s_1-s_2} i + \frac{(l-m)s_1}{l-s_1-s_2} \right\rfloor, \\ P_k^{(M)}[j] &= \left\lfloor \frac{m-s_1-s_2}{l-s_1-s_2} j \right\rfloor. \end{aligned} \tag{9}$$

Tail region ($i-j \ge l-s_2$): Shifted mapping for boundary context preservation:

$$P_q^{(T)}[i] = m-l+i, \quad P_k^{(T)}[j] = j. \tag{10}$$

Importantly, our dynamic mapping strategy preserves continuity across region boundaries, ensuring that the monotonicity property of relative position encoding is maintained: tokens at greater distances consistently receive larger relative position values. In addition, LaMPE can seamlessly integrate with efficient attention implementations such as FlashAttention2 (Dao, 2024). The mathematical guarantee of the continuity property and the implementation pseudocode of LaMPE with FlashAttention2 are detailed in Appendix C and D, respectively.

4 Experiments

4.1 Experimental Setup

Baselines. We conduct experiments on Llama2-7B-Chat-4K (Touvron et al., 2023), Llama3-8B-Instruct-8K (Meta, 2024) and Llama3.1-8B-Instruct-128K (Dubey et al., 2024). In addition to comparing LaMPE with the original RoPE, we evaluate it against several commonly used length extrapolation baselines, including NTK-RoPE (bloc97, 2023), YaRN (Peng et al., 2024), SelfExtend (Jin et al., 2024b), DCA (An et al., 2024b), and STRING (An et al., 2025). Among these, NTK-RoPE and YaRN enable extrapolation by modifying the base. STRING is designed to enhance performance within the original context window by modifying position indices. We reproduced their results using scripts from their official repositories, with configurations aligned to those reported in their original papers. All experiments are performed on a single A800 GPU using BF16 precision and accelerated with FlashAttention2 (Dao, 2024). We apply LaMPE to Llama3.1-8B-Instruct with real-world inputs to measure the latency and GPU memory usage, and the results are shown in Appendix E (Figure 8).

Datasets. We conduct comprehensive evaluations of our proposed method on five widely recognized long-context benchmarks.

LongBench (Bai et al., 2024) & L-Eval (An et al., 2024a): These two benchmarks are widely used for evaluating long-context methods, covering both real-world tasks and synthetic tasks. LongBench consists of 16 subtasks. For L-Eval, we select five representative tasks from it for evaluation. The average input length for both datasets is below 32K tokens.

∞Bench (Zhang et al., 2024): Designed to assess ultra-long-context understanding, ∞Bench includes a diverse set of real-world tasks and synthetic tasks with an average input length of more than 128K tokens.

RULER (Hsieh et al., 2024): A synthetic benchmark for long-context evaluation with 13 complex tasks across 4 categories: 8 NIAH variants, long-context QA, variable tracking, and counting. A key advantage of RULER is its flexibility: task length can be customized based on the model's context

	LongBench							L-Eval					
	S-DOC	**M-DOC**	**Sum**	**ICL**	**Syn.**	**Code**	**Avg.**	**Coursera**	**GSM**	**QuALITY**	**TOEFL**	**Sfiction**	**Avg.**
Llama2-7B-Chat	24.90	22.53	24.63	60.00	5.95	48.15	31.52	29.21	19.00	37.62	51.67	60.15	39.53
+ SelfExtend(16K)	27.30	26.23	24.81	64.15	5.74	57.46	34.62	35.76	25.00	41.09	55.39	57.81	43.01
+ SelfExtend(25K)	27.56	27.14	24.89	62.86	3.70	57.01	34.30	36.19	32.00	42.07	57.99	53.12	44.27
+ DCA(16K)	27.04	20.91	24.76	64.19	4.08	54.76	33.02	32.12	32.00	35.14	57.62	61.72	43.72
+ DCA(25K)	26.29	19.90	24.85	63.18	4.36	54.16	32.48	41.27	29.00	41.08	57.24	59.37	45.59
+ YaRN(16K)	22.77	13.51	21.83	47.91	1.98	47.68	26.08	42.44	20.00	41.58	58.36	42.18	40.91
+ YaRN(25K)	22.72	25.92	25.56	60.26	2.24	46.85	31.35	43.16	14.00	38.11	53.53	56.25	41.01
+ NTK(16K)	17.20	10.59	17.40	38.27	0.69	40.43	20.79	37.06	16.00	33.66	61.71	40.62	37.81
+ NTK(25K)	19.58	13.90	21.63	47.56	3.89	42.40	25.03	39.53	11.00	29.70	50.92	48.43	35.91
+ LaMPE	29.70	27.45	25.82	63.93	4.41	55.72	**35.07**	42.58	35.00	42.57	57.24	63.28	**48.13**
Llama3-8B-Ins	36.83	34.91	26.83	69.08	33.50	54.11	42.38	52.76	79.00	59.40	80.95	66.40	67.07
+ SelfExtend(16K)	14.76	7.26	21.53	48.40	2.11	57.14	24.71	55.08	78.00	63.36	80.66	67.34	68.88
+ SelfExtend(32K)	38.57	33.98	28.55	68.59	46.06	37.15	42.22	56.39	79.00	64.35	80.66	66.56	69.39
+ DCA(16K)	39.10	38.77	27.44	63.54	37.75	64.95	44.50	56.10	75.00	61.38	81.41	73.43	69.46
+ DCA(32K)	40.16	38.86	27.83	63.53	37.50	64.59	44.70	56.10	77.00	59.40	81.41	75.78	69.93
+ YaRN(16K)	35.42	33.81	26.40	56.58	48.50	61.90	42.34	58.43	78.00	65.34	79.92	66.40	69.61
+ YaRN(32K)	38.74	42.71	28.91	59.23	47.75	65.04	45.90	56.54	76.00	67.32	79.92	74.21	70.79
+ NTK(16K)	32.59	21.81	22.17	46.50	26.12	57.67	33.55	51.45	76.00	63.36	80.66	53.12	64.91
+ NTK(32K)	35.85	40.39	27.48	56.62	51.50	61.93	44.24	53.34	79.00	64.85	82.52	64.06	68.75
+ LaMPE	41.94	44.48	28.70	61.00	46.00	65.73	**46.99**	60.02	80.00	66.33	79.93	72.65	**71.78**

Table 1: Performance comparison of different baselines on LongBench and L-Eval. The LongBench and L-Eval benchmarks consist of 16 and 5 tasks, respectively. The number (e.g. '16K') indicates the maximum input length.

Models	**En.MC**	**En.QA**	**En.sum**	**Code.debug**	**Re.KV**	**Re.Number**	**Re.Passkey**	**Avg.**
Llama3-8B-Ins	50.66	10.54	16.23	25.13	6.60	6.78	20.12	19.43
+ SelfExtend (32K)	50.66	14.06	15.13	24.87	3.60	27.12	27.12	23.22
+ DCA (32K)	52.84	13.90	18.79	25.38	4.40	27.12	27.12	24.22
+ YaRN (32K)	39.30	10.42	15.84	26.14	0	26.10	27.12	20.70
+LaMPE (32K)	55.02	16.36	20.49	25.63	17.40	27.12	27.12	**27.02**
Llama3-8B-Ins	50.66	10.54	16.23	25.13	6.60	6.78	20.12	19.43
+ SelfExtend (64K)	53.71	15.10	15.22	21.57	2.80	54.24	54.24	30.98
+ DCA (64K)	50.66	14.35	18.98	24.11	2.00	52.88	54.24	31.03
+ YaRN (64K)	5.24	1.63	12.06	0.51	0	3.56	24.96	6.85
+ LaMPE (64K)	55.90	15.49	23.10	24.11	10.80	54.24	54.24	**33.98**
Llama3.1-8B-Ins	67.25	14.57	25.42	22.08	54.80	99.49	100.00	54.80
+ SelfExtend (128K)	68.12	15.58	26.26	24.62	57.20	100.00	100.00	55.96
+ DCA (128K)	67.79	13.67	26.75	24.37	70.40	100.00	100.00	57.55
+ YaRN (128K)	54.59	10.30	25.76	27.92	16.40	94.41	99.83	47.03
+ STRING (128K)	71.18	14.39	27.81	30.46	81.40	99.83	100.00	60.72
+ LaMPE (128K)	70.30	19.51	28.54	29.19	92.60	99.83	100.00	**62.85**

Table 2: Results of Llama3-8B-Instruct and Llama3.1-8B-Instruct on ∞Bench. Since all evaluation inputs exceed 64K tokens, the results on Llama3 fully reflect the extrapolation performance of LaMPE and other baseline methods.

window, allowing evaluations up to 128K tokens or beyond.

PG-19 (Rae et al., 2020): PG-19 is a traditional benchmark for long-context language modeling comprising over 28,000 books published before 1919, serves as the final dataset in our experiments. Evaluations on PG-19 can be found in Appendix F.1.

4.2 Performance on Long Context Benchmarks

LongBench & L-Eval. As shown in Table 1, LaMPE achieves the best overall performance on both LongBench and L-Eval with Llama2-7B-Chat and Llama3-8B-Instruct. Since baseline methods require manual tuning of extrapolation factors or group sizes, there exists an upper bound for extrapolation. However, as the distribution of datasets does not consist entirely of the specified length, we evaluate two different maximum input lengths under two different experimental settings. Specifically, we set maximum input lengths of 16K and 25K tokens for Llama2, and 16K and 32K tokens for Llama3. In contrast, our method does not require manual setting of maximum length and only needs one set of experiments.

In summary, our method applied to Llama2 and Llama3 respectively outperforms the best baseline by 0.45 and 1.09 points on average across 16 tasks on LongBench. On L-Eval, our method achieves improvements of 2.54 and 0.99 points, respectively. Specifically, on Llama2-7B-Chat, position indices modified methods (SelfExtend, DCA) significantly outperform base-modified methods (YaRN, NTK-

RoPE), though this performance gap narrows on Llama3-8B-Instruct due to the model's stronger capacity.

Method	RULER				
	8K	**16K**	**32K**	**64K**	**128K**
Llama3-8B-Instruct					
RoPE	88.76	-	-	-	-
SelfExtend	87.59	75.46	69.06	61.95	35.97
DCA	89.35	72.28	61.99	47.01	15.96
YaRN	88.76	62.93	58.36	5.02	12.17
LaMPE	**90.17**	**87.32**	**79.11**	**69.46**	**59.48**

Table 3: Performance of Llama3-8B-Instruct on the RULER benchmark across input lengths from 8K to 128K.

∞Bench. We evaluate the performance of various baselines on Llama3-8B-Instruct at extrapolation lengths of 4× (32K) and 8× (64K). Since all tasks in the ∞Bench greatly exceed 64K tokens, these results directly reflect the extrapolation capability of each method. The experimental results are shown in Table 2, LaMPE consistently achieves higher average scores than the baselines at both 32K and 64K. We observe that methods based on modifying position indices tend to be more stable than those modifying base. Notably, when the extrapolation length reaches 8×, YaRN exhibits a substantial performance degradation. In addition to evaluating LaMPE on Llama3-8B-Instruct with an 8K context window, we further apply it to Llama3.1-8B-Instruct, which supports a 128K context window. As shown in the third block of Table 2, when the maximum input length is set to 128K tokens, our method again achieves the best performance. In particular, for the KV retrieval task, our method outperforms the original RoPE by 37.8 points. It indicates that LaMPE can also effectively scale to long-context LLMs.

RULER. Table 3 presents the performance of LaMPE under stress testing on the RULER benchmark and detailed per-task scores are available in Appendix F.2. The evaluation spans from the pretraining context window of 8K up to 128K. Across this entire range, LaMPE consistently achieves the highest accuracy compared to all other baseline methods. Notably, we found similar patterns as observed on ∞Bench: when the extrapolation length reaches 64K, YaRN exhibits a dramatic performance drop. In contrast, modifying position indices methods (SelfExtend, DCA) are relatively stable, indicating that YaRN's extrapolation upper bound is lower than that of modifying position indices methods. We use mean-based decomposition to disentangle the position vectors to account for this phenomenon, the results and analysis can be found in Appendix G. Furthermore, besides achieving the best performance at extrapolation lengths, our method also achieves 90.57 within the pretraining context window, surpassing the original RoPE's 88.76. This demonstrates that utilizing the left-skewed frequency distribution of relative positions can achieve efficient extrapolation.

Method	Llama2-7B-Chat		Llama3-8B-Instruct	
	Below [0-4k]	*Beyond* [4K+]	*Below* [0-8k]	*Beyond* [8K+]
LongBench				
RoPE	42.86	31.83	45.03	41.40
SelfExtend	41.14	30.75	45.49	46.54
DCA	42.10	31.56	47.25	44.44
YaRN	38.70	30.46	**48.95**	44.82
NTK	43.43	20.58	48.48	38.91
LaMPE	**44.23**	**33.54**	48.68	**47.00**
L-Eval				
RoPE	53.43	42.49	68.22	53.58
SelfExtend	52.48	45.28	71.74	62.07
DCA	**58.91**	45.58	70.16	58.90
YaRN	50.11	43.56	71.75	**64.37**
NTK	37.25	38.31	71.95	53.77
LaMPE	57.72	**48.34**	**74.04**	62.93

Table 4: Performance statistics on LongBench and L-Eval, where *Below* denotes input lengths within the context window and *Beyond* denotes exceed.

4.3 Analysis

Performance Enhancement on Pretraining Context Window. To further assess LaMPE's performance within the pretraining context window, we categorize LongBench and L-Eval based on whether they fall inside or exceed the pretraining context window, and compute performance metrics for the two categories separately. As shown in Table 4, our method consistently achieves optimal or suboptimal performance both at extrapolation lengths and within the pretraining context window. This demonstrates that LaMPE can serve as a general technique for long-context scaling. Moreover, it can also be observed from the figure that DCA demonstrates strong stability across both extrapolation lengths and the pretraining context window, while SelfExtend's results on Llama3 in LongBench show that it impairs performance within the original window, which also highlights the limitations of manually setting group size.

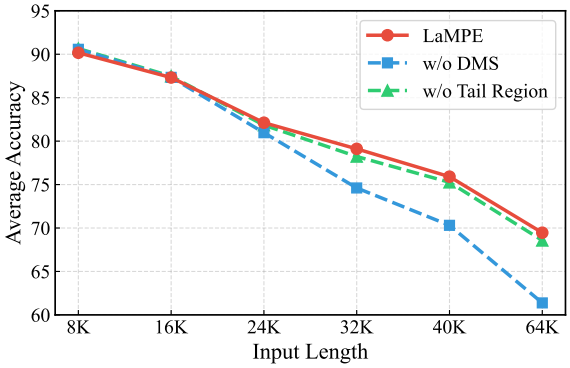


Figure 4: Ablation of DMS and Tail region on RULER benchmark using Llama3-8B-Instruct.

Method	FWE		CWE	
	32K	64K	32K	64K
LaMPE	**90.00**	**87.53**	**34.33**	**4.52**
Ablation on DMS				
w/ m=1536	76.13	78.66	12.74	1.20
w/ m=2048	66.33	74.33	11.22	0.98
w/ m=3072	88.66	84.20	22.61	1.10
w/ m=4096	89.93	83.53	24.44	1.30
w/ m=5120	89.53	84.46	22.26	0.04
w/ m=6144	89.00	84.33	29.24	0.14
Ablation on Tail region				
w/o Tail	89.46	84.26	30.73	0.76

Table 5: Ablation of DMS and Tail region on the FWE and CWE tasks.

Ablation Study. To evaluate the contributions of two components in LaMPE, we introduce two variants for the ablation study: (1) *w/o DMS*, which uses a fixed mapping length that remains constant across inputs of varying lengths; (2) *w/o Tail region*, which does not recover long-range dependencies between current tokens and initial tokens. As shown in Table 5, we first evaluate these variants on two representative tasks from RULER: Frequent Words Extraction (FWE), and Common Words Extraction (CWE). The results show that removing DMS and replacing it with a fixed mapping length m leads to substantial performance degradation. Removing the tail region (*w/o Tail*) causes notable drops, especially on 64K CWE (4.52% $\rightarrow$ 0.76%). Similar results can also be observed in Figure 4. These results demonstrate the necessity of both components, with the dynamic mapping strategy being particularly critical for maintaining performance across varying input lengths.

Hyperparameter Analysis. We investigate the effects of head region size s_1 and tail region size

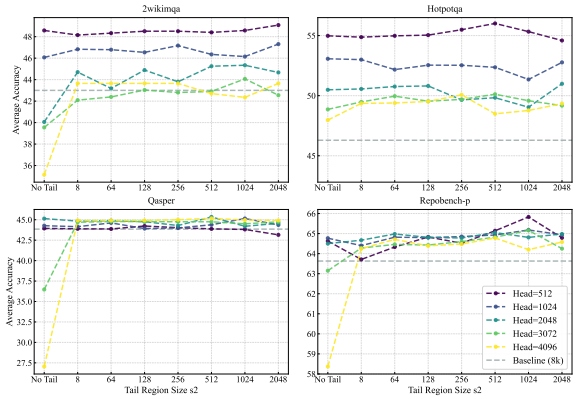


Figure 5: Hyperparameter analysis results on LongBench Using different head and tail region sizes.

w/o Tail region	Tail region = 8
Ozalj, present day Croatia.	Ozalj.
No. The to the. The to the. ...	No.
Alt \n_REF. The to the. The to the. The to the. ...	Altuğ Çelikbilek

Table 6: Examples of outputs with and without the tail region. Gray highlights indicate the major differences.

s_2 on performance, adjusting the mapping length accordingly for larger head regions. As shown in Figure 5, we observe a surprising phenomenon: when the head region size s_1 exceeds 1/4 context window and no tail region is included ($s_2 = 0$), the model experiences significant performance degradation. Remarkably, introducing even a very small tail region (e.g., $s_2 = 8$) almost fully restores performance. Table 6 summarizes the key differences between models with and without the tail region, showing that the absence of the tail region often leads to redundant and repetitive outputs. This finding underscores the compensatory role of the tail region in recovering long-range dependencies and provides strong evidence for the effectiveness of our Multi-grained Attention mechanism. In practice, we recommend setting s_1 to approximately 1/16 of the pretraining context window, and s_2 to a small but non-zero value ranging from 8 to 1024 tokens.

5 Conclusion

We have proposed LaMPE, a training-free method for adaptive long-context scaling of RoPE-based large language models. LaMPE models the dynamic relationship between mapping length and input length using a parametric scaled sigmoid function, and incorporates a multi-grained attention mechanism to jointly capture fine-grained lo-

cality and long-range dependencies. Extensive experiments across long-context benchmarks demonstrate that LAMPE significantly improves extrapolation performance and enhances performance within the original context window.

Limitations

One limitation of our work is the focus on Multi-Head Attention (MHA), Grouped-Query Attention (GQA), and Multi-Head Latent Attention (MLA), without extending to modern linear attention mechanisms. Future work could expand to encompass additional architectures. Additionally, due to computational resource constraints, we do not investigate whether the two observed patterns (the V-shaped pattern and the monotonically decreasing pattern) emerge in larger-scale models.

Acknowledgments

This work is supported by the Beijing Natural Science Foundation (Grants L243015, L223003), the Natural Science Foundation of China (No. 62192782, 62532015), and the Beijing Major Science and Technology Project under Contract no. Z251100008425008.

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)*, 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*.

Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. 2025. Why does the effective context length of LLMs fall short? In *The Thirteenth International Conference on Learning Representations*.

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)*, Bangkok, Thailand. Association for Computational Linguistics.

bloc97. 2023. NTK-Aware Scaled RoPE allows LLaMA models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

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

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. LongRoPE: Extending LLM context window beyond 2 million tokens. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11091–11104. PMLR.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. *Preprint*, arXiv:2401.14196.

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)*, pages 3991–4008. Association for Computational Linguistics.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*.

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. *Preprint*, arXiv:2310.06825.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024a. LLM maybe LongLM: Self-Extend LLM context window without tuning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22099–22114. PMLR.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024b. LLM maybe LongLM: Self-Extend LLM context window without tuning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22099–22114. PMLR.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context llms struggle with long in-context learning. *Preprint*, arXiv:2404.02060.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

AI Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

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

Adithya Pratapa and Teruko Mitamura. 2025. Scaling multi-document event summarization: Evaluating compression vs. full-text approaches. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, Albuquerque, New Mexico. Association for Computational Linguistics.

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

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, and 7 others. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Ning Shang, Li Lyna Zhang, Siyuan Wang, Gaokai Zhang, Gilsinia Lopez, Fan Yang, Weizhu Chen, and Mao Yang. 2025. LongroPE2: Near-lossless LLM context window scaling. In *Forty-second International Conference on Machine Learning*.

Jianlin Su. 2023a. The evolution of transformers: 8. length generalization and positional robustness.

Jianlin Su. 2023b. The road to transformer upgrades: 12. rerope for unlimited extrapolation?

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

MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2023-05-05.

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, and 49 others. 2023. 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, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian

Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand. Association for Computational Linguistics.

zican Dong, Junyi Li, Xin Men, Xin Zhao, Bingning Wang, Zhen Tian, weipeng chen, and Ji-Rong Wen. 2024. Exploring context window of large language models via decomposed positional vectors. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

A Related Work

Positional Encoding. Positional encoding is a fundamental technique for incorporating position information into transformers and can be broadly categorized into two types: absolute and relative positional encoding. Absolute positional encoding associates each position with either learnable parameters (Devlin et al., 2019; Lan et al., 2020) or fixed sinusoidal embeddings as employed in the original transformer (Vaswani et al., 2017). In contrast, relative positional encoding encodes the relative distances between tokens rather than their absolute positions. Among relative positional encoding methods, ALiBi (Press et al., 2022) introduces a position-biased attention mechanism that applies linear decay to attention weights based on token distance, successfully implemented in MPT (Team, 2023). Another notable method is Rotary Position Embedding (RoPE) (Su et al., 2024), which has achieved widespread adoption due to its excellent performance. RoPE has been integrated into several state-of-the-art language models (Rozière et al., 2024; Dubey et al., 2024; Jiang et al., 2023), demonstrating its effectiveness across diverse architectures and applications. Our work focuses on RoPE-based LLMs, aiming to enhance their capability for long-context scaling.

Length Extrapolation based RoPE. We divide length extrapolation methods based RoPE on into two types: base-modified and position indices modified. Inspired by neural tangent kernel theory, NTK-aware (bloc97, 2023), Dynamic-NTK (emozilla, 2023) and YaRN (Peng et al., 2024) perform extrapolation on high-frequency components and interpolation on low-frequency components. Subsequently, LongRoPE(Ding et al., 2024; Shang et al., 2025) uses an evolutionary search to exploit two forms of non-uniformities in RoPE Frequency. These methods can work without training, but a small number of training steps can lead to better performance. In contrast, ReRoPE(Su, 2023b), SelfExtend(Jin et al., 2024b) and DCA(An et al., 2024b) achieve length extrapolation by modifying position indices that exceed a certain range to the limiting value. The first type methods are orthogonal to our method and could be integrated with our techniques. Our method belongs to the second type and employs a dynamic mapping strategy to fully leverage effective positions for length extrapolation.

B Additional Observations and Fitting Details

Additional Observations. The V-shaped and monotonically decreasing patterns for Llama2 and Llama3 are illustrated in Figures 6 and 7. As the input length increases, the optimal mapping length increases correspondingly. We set the maximum mapping length to 3/4 of the pretraining context window, as beyond this point perplexity increases substantially. This observation aligns closely with findings from STRING (An et al., 2025), where needle-in-a-haystack tasks consistently fail in the last 1/3 of sequences, occurring predominantly in the tail of the position frequency distribution. These results demonstrate that mechanically using the entire pretraining context window for extrapolation is inadequate and that one should instead fully leverage well-trained positions.

Fitting Details. Given the optimal mapping lengths across various input lengths, we apply nonlinear least squares fitting to these points. Let $f(x)$ denote the scaled sigmoid function parameterized by a, b, and L. The fitting process can be formulated as:

$$\min_{a,b,L} \sum_i (y_i - f(x_i; a, b, L))^2. \tag{11}$$

Here, L controls the saturation stage of the curve, representing the upper limit of the mapping length. Due to the introduction of the parameters a, b, and L, LaMPE requires a separate dynamic mapping configuration for each model. In practice, we can

easily scale to other models with different pretraining context windows by simply adjusting L, and still achieve strong performance. As shown in Table 7, we validate this generalization ability on Mistral-7B-Instruct-v0.1 and DeepSeek-V2-16B by setting L to three-quarters of their pretraining context windows (6K and 12K, respectively).

Model	LongBench						
	S-DOC	M-DOC	Sum	ICL	Syn.	Code	Avg.
DeepSeek-V2	38.61	32.15	28.02	68.29	27.13	61.85	42.45
+ LaMPE	41.72	35.86	27.38	69.28	26.50	58.29	**43.27**
Mistral-7B	30.74	29.88	25.58	65.15	14.38	54.45	36.98
+ LaMPE	35.58	32.94	27.28	66.61	20.77	52.16	**39.57**

Table 7: Scaling LaMPE to different models on LongBench tasks.

C Mathematical Guarantee of the Continuity Property

LaMPE achieves length extrapolation by mapping larger position indices to smaller ones through region-specific transformations. This section provides a rigorous mathematical proof that the dynamic mapping strategy preserves the monotonicity property essential for maintaining coherent relative position relationships in attention mechanisms.

We first formalize the monotonicity property for relative position encodings. Let $PE[i][j]$ denote the relative position value between token i and token j, where $PE[i][j] = P_q[i] - P_k[j]$ represents the position difference that determines the rotation angles in RoPE-based attention computation.

Definition (Monotonicity Property): For any token position i , the relative position encodings satisfy monotonicity if for all $0 \leq j_1 < j_2 \leq i$, we have $PE[i][j_1] \geq PE[i][j_2]$.

This property ensures that tokens closer to the current position maintain smaller relative distances, preventing discontinuous jumps in the relative position sequence that could disrupt attention patterns.

To establish our proof, we introduce the following notation from the middle region transformation:

$$k = \frac{m - s_1 - s_2}{l - s_1 - s_2}, \quad b = \frac{(l-m)s_1}{l - s_1 - s_2}, \tag{12}$$

where $0 < k \leq 1$ since $0 < m \leq l$ in our method. These parameters satisfy two properties:

$$ks_1 + b = s_1, \quad k(l - s_2) + b = m - s_2. \tag{13}$$

Our proof relies on the floor function inequality:

$$\lfloor x \rfloor - \lfloor y \rfloor - 1 \leq \lfloor x - y \rfloor \leq \lfloor x \rfloor - \lfloor y \rfloor. \tag{14}$$

Since LaMPE partitions the sequence into three regions with distinct mapping functions, we establish monotonicity by proving continuity at the boundaries between adjacent regions, as monotonicity within each region follows directly from the mapping definitions.

Boundary between the Head and Middle Regions: Consider any token position at i and boundary positions j_1 and j_2 where $i - j_1 = s_1$ (head region) and $i - j_2 = s_1 + 1$ (middle region). We need to show $PE[i][j_2] \geq PE[i][j_1]$. Note that $PE[i][j_1] = i - j_1 = s_1$ by the head region mapping, while $PE[i][j_2]$ follows the middle region transformation.

$$\begin{aligned} PE[i][j_2] &= \lfloor ki + b \rfloor - \lfloor kj_2 \rfloor, && (15)\\ &\geq \lfloor k(i - j_2) + b \rfloor, && (16)\\ &= \lfloor (ks_1 + b) + k \rfloor = s_1 + \lfloor k \rfloor, && (17)\\ &\geq s_1 = PE[i][j_1]. && (18) \end{aligned}$$

Boundary between the Middle and Tail Regions: Consider boundary positions where $i - j_2 = l - s_2 - 1$ (middle region) and $i - j_3 = l - s_2$ (tail region). We need to show $PE[i][j_2] \leq PE[i][j_3]$. Note that $PE[i][j_3] = (m - l + i) - j_3 = m - s_2$ by the tail region mapping.

$$\begin{aligned} PE[i][j_2] &= \lfloor ki + b \rfloor - \lfloor kj_2 \rfloor, && (19)\\ &\leq \lfloor k(i - j_2) + b \rfloor + 1, && (20)\\ &= \lfloor k(l - s_2) + b + 1 - k \rfloor, && (21)\\ &= m - s_2 + \lfloor 1 - k \rfloor, && (22)\\ &= m - s_2 = PE[i][j_3]. && (23) \end{aligned}$$

These boundary conditions, combined with the inherent monotonicity within each region, establish that our dynamic mapping strategy preserves the monotonicity property across the entire sequence. This mathematical guarantee ensures that LaMPE maintains coherent relative position relationships during length extrapolation, preventing the attention mechanism from encountering discontinuous position patterns that could degrade model performance.

D Pseudocode of LaMPE

As shown in Algorithm 1, we provide a Python-style pseudocode implementation of our method integrated with FlashAttention2. The implementation consists of three distinct regions, each with specific attention mechanisms:

Head Region. As shown in Algorithm 1 (`line 5-8`), there is no need to modify the query or key.

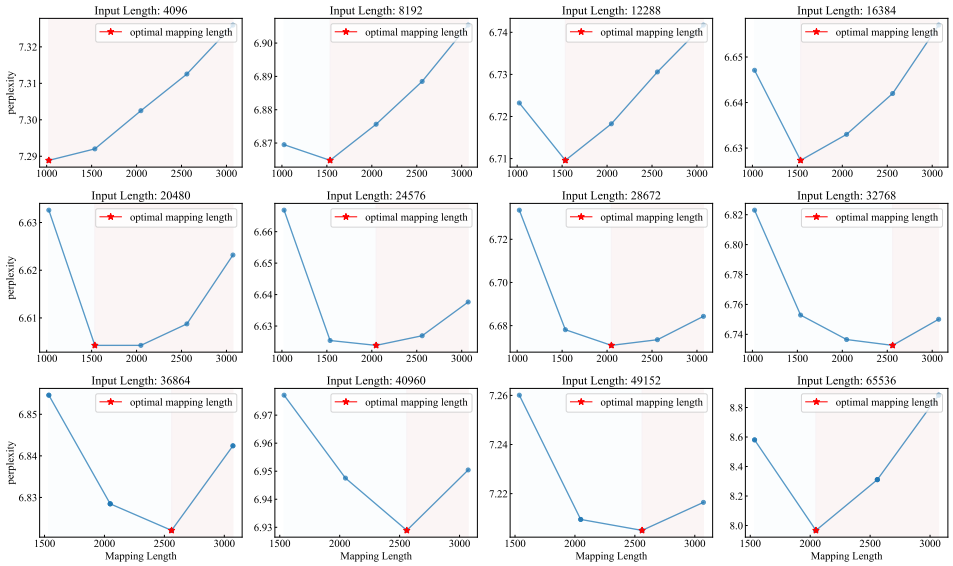


Figure 6: Visualization of two patterns of PPL variation with mapping length in Llama2-7B-Chat.

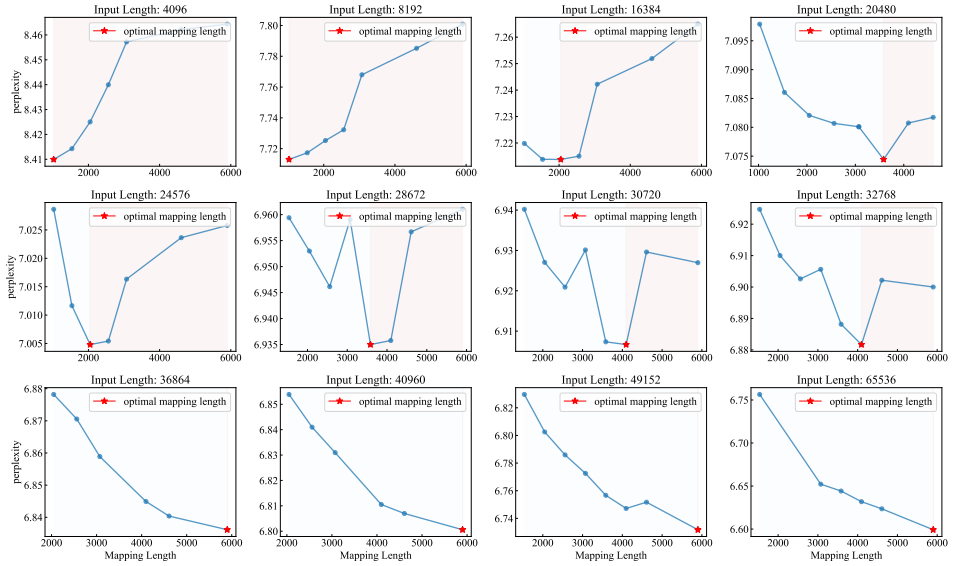


Figure 7: Visualization of two patterns of PPL variation with mapping length in Llama3-8B-Instruct.

This is a standard sliding window attention with a window size of s_1.

Middle Region. `Line 10-14` corresponds to the implementation of the linear normalization. It is necessary to modify the position indices of query and key to the normalized query (`line 12`) and key (`line 13`). We matmul the last query vector `Q_m[s1-`l`:]` and key vector `K_m[:`l`-s1]` with sliding window size $l - s_1 - s_2$.

Tail Region. The tail region implements full attention using a lower triangular mask, where the query vectors are modified with position encoding (`line 18`) while the key vectors remain unchanged (`line 19`). This step computes the attention for the last s_2 tokens (`line 20`).

In the decoding stage, for each generated token, we refrain from extending the head and middle regions and instead increase the length of the tail region to better preserve long-range dependencies.

E Efficiency analysis

Figure 8 shows the latency and GPU memory usage of LaMPE across different input lengths on a single A800. We take Llama 3.1 8B as the base model, using the summarization task from InfiniteBench, setting the decode length to 10, and the model is evaluated 30 times. It can be seen that as input length increases, LaMPE introduces only approximately 0.5s of average latency overhead compared to the standard FlashAttention implementation. Mean-

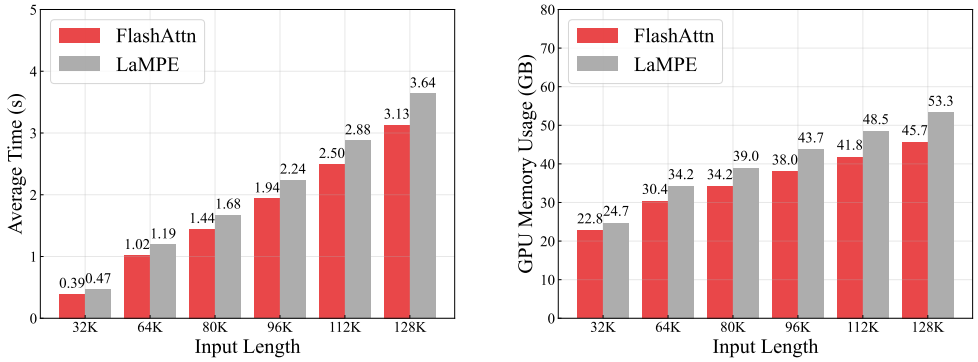


Figure 8: Average time per token and GPU memory usage across different input lengths based on Llama3-8B-Instruct. FlashAttn refers to the official Transformers implementation.

	16K				24K				32K			
	SelfExtend	**DCA**	**YaRN**	**LaMPE**	**SelfExtend**	**DCA**	**YaRN**	**LaMPE**	**SelfExtend**	**DCA**	**YaRN**	**LaMPE**
NIAH_S1	100.00	100.00	100.00	100.00	100.00	100.00	92.60	100.00	100.00	100.00	91.20	100.00
NIAH_S2	99.80	83.40	99.40	100.00	98.40	74.60	94.80	100.00	99.80	56.50	95.40	100.00
NIAH_S3	99.80	92.60	98.40	99.80	99.80	84.60	92.60	100.00	100.00	71.80	90.20	99.80
NIAH_M1	86.00	56.80	83.00	96.60	83.00	52.60	48.20	95.00	89.60	46.40	48.60	92.80
NIAH_M2	48.80	47.80	24.00	94.80	36.80	42.60	0.6	83.80	42.00	37.00	12.80	81.40
NIAH_M3	37.80	33.40	10.40	73.40	21.20	29.20	22.40	59.80	25.80	15.80	0.20	56.80
NIAH_MQ	99.25	92.10	97.85	98.85	97.75	83.45	72.00	98.10	98.40	81.80	94.80	98.05
NIAH_MV	93.10	89.20	97.95	98.05	94.60	84.25	89.85	98.40	94.95	77.30	91.35	97.20
VT	68.64	64.40	2.52	83.00	52.99	64.32	0	84.20	60.12	65.44	76.28	77.12
CWE	54.28	87.41	14.02	77.63	46.11	80.95	11.00	56.72	19.40	68.31	0.04	34.33
FWE	91.13	92.20	87.26	91.93	81.33	85.00	67.93	82.86	89.00	92.66	84.60	90.00
QA_1	59.40	59.00	71.00	73.20	51.40	54.60	38.80	64.20	47.80	55.80	42.20	57.00
QA_2	42.80	41.40	32.40	48.00	34.90	41.80	31.20	44.60	38.00	36.80	31.00	44.00
AVg.	75.44	72.28	62.93	**87.32**	69.44	67.53	50.92	**82.12**	69.06	61.99	69.06	**79.12**
	40K				**64K**				**128K**			
	SelfExtend	**DCA**	**YaRN**	**LaMPE**	**SelfExtend**	**DCA**	**YaRN**	**LaMPE**	**SelfExtend**	**DCA**	**YaRN**	**LaMPE**
NIAH_S1	100.00	99.60	99.00	100.00	100.00	99.00	0	100.00	100.00	85.20	75.40	100.00
NIAH_S2	98.20	50.00	93.20	99.60	99.20	43.20	0	98.80	50.40	31.20	11.80	85.00
NIAH_S3	91.60	61.40	98.40	99.80	98.40	44.00	0	100.00	8.20	20.20	0	99.00
NIAH_M1	72.40	45.00	60.00	91.00	75.80	37.80	0	89.80	29.60	21.20	11.20	48.40
NIAH_M2	19.40	33.40	18.40	67.40	17.20	25.00	0.6	46.00	4.60	1.60	0	16.40
NIAH_M3	9.80	14.00	1.20	45.80	12.60	3.20	0	14.60	4.00	0	0	1.20
NIAH_MQ	93.55	75.30	84.70	97.85	95.30	59.40	0	96.35	48.85	6.05	0	89.60
NIAH_MV	91.15	74.95	81.55	97.05	95.30	64.10	0	96.60	35.90	6.70	0.90	89.90
VT	45.79	65.40	79.44	76.12	43.80	63.80	0	79.24	57.35	2.79	00	62.04
CWE	34.78	51.59	7.56	34.89	0.26	21.11	0	4.52	0.02	1.20	0.02	0.02
FWE	84.53	91.06	90.06	88.86	84.60	88.40	47.30	87.53	85.00	0.73	56.53	88.33
QA_1	40.60	39.20	47.00	50.20	49.00	31.80	9.40	55.00	25.20	14.00	1.80	57.40
QA_2	33.60	33.00	34.20	38.20	34.00	30.40	8.00	34.60	22.20	16.60	0.60	36.00
AVg.	62.72	56.45	61.13	**75.90**	61.95	47.01	5.02	**69.46**	35.97	15.96	12.17	**59.48**

Table 8: Comparison of per-task performance across different baselines on RULER at extrapolated lengths.

while, the GPU memory increase remains below 10GB.

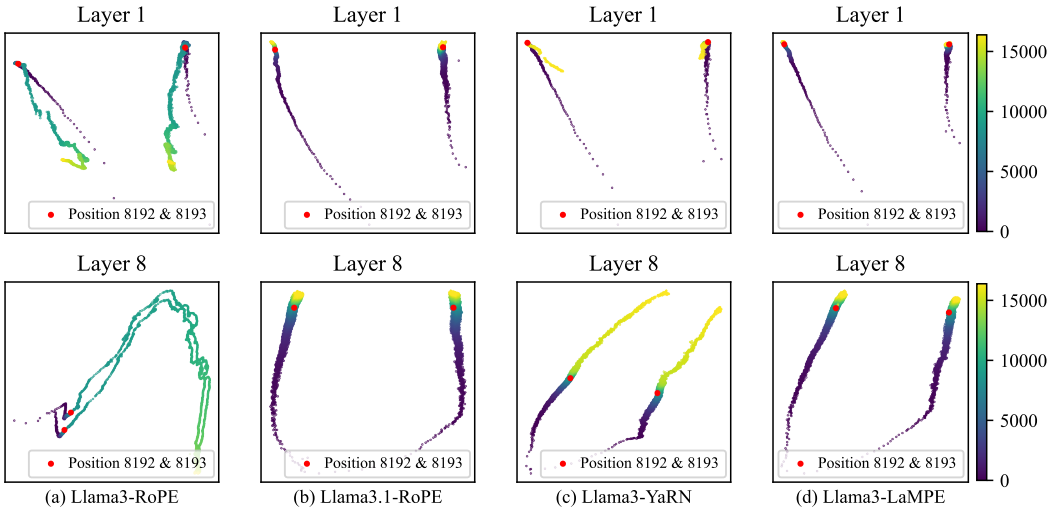
F Additional Experiment

F.1 PG19

Due to limited computational resources, we sample 200 paragraphs from 100 books to construct our test set, while keeping all other experimental settings consistent with prior work. This setup significantly reduces resource requirements while providing a reliable assessment of LLMs' perplexity. Table 9 presents the perplexity scores on the PG19, LaMPE maintains relatively low PPL growth even as input length increases. Compared to other methods, LaMPE consistently achieves either the best or second-best results. For Llama2-7B-Chat, LaMPE keeps PPL below 7 across the 8K–40K

Method	Evaluation Context Window Size						
	4K	**8K**	**16K**	**24K**	**32K**	**40K**	**64K**
Llama2-7B-Chat	7.13	$> 10^2$	$> 10^2$	$> 10^2$	$> 10^2$	$> 10^2$	$> 10^2$
+ SelfExtend	7.13	6.85	7.44	22.72	74.38	$> 10^2$	$> 10^2$
+ DCA	7.12	6.87	6.79	6.87	7.07	7.36	8.53
+ YaRN	7.13	6.96	7.68	7.74	8.40	10.68	30.90
+ NTK	7.13	7.05	14.79	$> 10^2$	$> 10^2$	$> 10^2$	$> 10^2$
+ Ours	7.29	6.86	6.62	6.62	6.73	6.97	7.96
Llama3-8B-Ins	8.81	8.14	39.67	$> 10^2$	$> 10^2$	$> 10^2$	$> 10^2$
+ SelfExtend	8.81	8.10	7.56	7.34	7.26	7.16	7.00
+ DCA	8.81	8.11	7.42	7.01	6.94	6.86	6.84
+ YaRN	8.81	8.14	7.28	7.02	6.87	6.81	6.93
+ NTK	8.81	8.14	7.99	7.83	7.99	8.29	9.76
+ Ours	8.41	7.70	7.21	7.00	6.90	6.80	6.59

Table 9: Perplexity (PPL) evaluation comparison of different baselines on PG-19.



(a) Llama3-RoPE (b) Llama3.1-RoPE (c) Llama3-YaRN (d) Llama3-LaMPE

Figure 9: PCA visualization of positional vectors from the 1st and 8th layers. (a) Original RoPE of Llama3-8B. (b) Original RoPE of Llama3.1-8B. (c) YaRN-extended Llama3-8B. (d) LaMPE-extended Llama3-8B.

Algorithm 1 Pseudocode of LaMPE with FlashAttention2

```
#  Q, K, V: Queries, Keys, Values
#  s1, s2: head region size, tail region size
#  m, l: mapping length ,input  length
#  Head Region
pos = [0,1,2,...l-1]
Q_h = apply_rotary_pos_emb(Q, pos)
K_h = apply_rotary_pos_emb(K, pos)
O_h, lse_h = flash_attn_func(Q, K, window_size=(
    s1, 0))
#  Middle Region
q_m_pos = floor((m-s1-s2)/(l-s1-s2) * pos + (l-m)s1/(l-s1-s2))
k_m_pos = floor((m-s1-s2)/(l-s1-s2) * pos)
Q_m = apply_rotary_pos_emb(Q, q_m_pos)
K_m = apply_rotary_pos_emb(K, k_m_pos)
O_m, lse_m = flash_attn_func(Q_m[s1-l:], K_m[:l-s1
    ], window_size=(l-s1-s2, 0))
#  Tail Region
q_t_pos = pos +m - l
k_t_pos = pos
Q_t = apply_rotary_pos_emb(Q, q_t_pos)
K_t = apply_rotary_pos_emb(K, k_t_pos)
O_t, lse_t = flash_attn_func(Q_t[-s2:], K_t[:s2],
     window_size=(-1, -1))
# Merge hidden_state O_h, O_m, O_t
# Step 1: Split outputs
head_h = O_h[:, :s1]
midd_h = O_h[:, s1:l-s2-s1]
tail_h = O_h[:, l-s2:]
midd_m = O_m[:, :l-s2-s1]
tail_m = O_m[:, l-s2-s1:]
# Step 2: Merge middle parts
gate1 = sigmoid(lse_m[:l-s2-s1] - lse_h[s1:l-s2])
gate2 = 1 - gate1
midd = midd_h * gate2 + midd_m * gate1
# Step 3: Merge tail parts
gate1 = 1 / (1 + exp(lse_h[l-s2:] - lse_m[l-s2-s1
    :]) + exp(lse_t - lse_m[l-s2-s1:]))
gate2 = 1 / (1 + exp(lse_m[l-s2-s1:] - lse_t) +
    exp(lse_h[l-s2:] - lse_t))
gate3 = 1 / (1 + exp(lse_t - lse_h[l-s2:]) + exp(
    lse_m[l-s2-s1:] - lse_h[l-s2:]))
tail = tail_m * gate1 + O_t * gate2 + tail_h *
    gate3
# Step 4: Concatenate
output = concat(head_h, midd, tail)
```

input length, while other methods like SelfExtend, YaRN, and NTK-RoPE all exceed 10. For Llama3-8B-Instruct, LaMPE consistently achieves the lowest PPL across all input lengths.

	Pretraining Context Window (8K)			
	RoPE	**SelfExtend**	**DCA**	**LaMPE**
NIAH_S1	100.00	100.00	100.00	100.00
NIAH_S2	99.80	99.80	100.00	100.00
NIAH_S3	99.80	100.00	99.40	100.00
NIAH_M1	95.00	93.60	96.80	98.60
NIAH_M2	84.60	89.80	95.60	98.80
NIAH_M3	95.00	92.60	94.60	93.80
NIAH_MQ	99.45	99.40	99.35	99.35
NIAH_MV	99.20	94.60	98.75	99.75
VT	91.08	82.40	88.84	90.07
CWE	92.59	92.13	91.19	86.19
FWE	83.40	81.60	83.33	83.06
QA_1	66.40	65.80	67.20	75.20
QA_2	47.60	47.00	46.60	52.60
AVg.	88.76	87.59	89.35	**90.57**

Table 10: Comparison of per-task performance across different baselines on RULER within the pretraining context window.

F.2 Per-task performance on RULER

Table 10 presents RULER's per-task performance within the pretraining context window, and Table 8 shows the performance at extrapolated lengths.

G Visualization of Positional Vectors

Following (zican Dong et al., 2024), we adopt a mean-based decomposition method to disentangle positional vectors from the hidden states of LLMs, and visualize them with PCA. Specifically, we test Llama3-8B with an 8K context window.

When evaluated at a 16K input length, the positional vectors in the 8K–16K range become out-of-distribution (OOD), exhibiting abnormal patterns as shown in Figure 9(a). As a comparison, we also visualize Llama3.1-8B with a 128K context window, where the same 8K–16K region remains in-distribution and thus provides a reliable baseline in Figure 9(b). We further extend the context window of Llama3-8B to 16K using YaRN and LaMPE, and compare their behaviors against both the original Llama3-8B and Llama3.1-8B. The results reveal that YaRN produces positional distributions beyond 8K that diverge substantially from Llama3.1, whereas LaMPE exhibits much higher similarity to Llama3.1. Meanwhile, we also observe that the distributions become sparser toward the left positions, indicating a stronger capacity to encode positional information. In contrast, the right (larger) positions exhibit more concentrated patterns, suggesting that models can better capture positional information by using the left-side positions.