

# Mitigating Position Bias in Transformers via Layer-Specific Positional Embedding Scaling

Zhenghua Wang<sup>\*1,3</sup>, Yiran Ding<sup>\*2</sup>, Changze Lv<sup>\*1,3</sup>, Yixin Wu<sup>1,3</sup>, Tianlong Li<sup>1,3</sup>,  
Zhibo Xu<sup>1,3</sup>, Muling Wu<sup>1,3</sup>, Tianyuan Shi<sup>1,3</sup>, Shizheng Li<sup>1,3</sup>, Qi Qian<sup>1,3</sup>,  
Xuanjing Huang<sup>1,3</sup>, Xiaoqing Zheng<sup>†1,3</sup>

<sup>1</sup>Fudan University, <sup>2</sup>Westlake University

<sup>3</sup>Shanghai Key Laboratory of Intelligent Information Processing

{zhenghuawang23, czlv24}@m.fudan.edu.cn {yiran.ding}@hdu.edu.cn

{xjhuang, zhengxq}@fudan.edu.cn

## Abstract

Large Language Models (LLMs) still struggle with the “lost-in-the-middle” problem, where critical information located in the middle of long-context inputs is often underrepresented or lost. While existing methods attempt to address this by combining multi-scale rotary position embeddings (RoPE), they typically suffer from high latency or rely on suboptimal hand-crafted scaling strategies. To overcome these limitations, we introduce a layer-specific positional embedding scaling (LPES) method that assigns distinct scaling factors to each layer. LPES achieves a more balanced attention distribution without fine-tuning model parameters or increasing inference delay. A specially designed genetic algorithm is employed to efficiently select the optimal scaling factors for each layer by incorporating Bézier curves to significantly reduce the search space. Extensive experiments demonstrate that LPES effectively mitigates positional attention bias and delivers consistent improvements across multiple long-context benchmarks, yielding up to an 11.2% accuracy gain on the key-value retrieval dataset.

## 1 Introduction

Enabling Large Language Models (LLMs) to process long inputs is essential for supporting complex tasks such as long-text summarization (Feng et al., 2021; Zhang et al., 2021), code generation (Zheng et al., 2023; Liu et al., 2024a), and long-context question-answering (Li et al., 2024). Rotary position embeddings (RoPE) (Su et al., 2021), widely adopted in transformer-based LLMs, were designed to encode relative distances between input tokens, facilitating more effective processing of long-context inputs. However, as the context length increases, RoPE-based LLMs continue to

suffer from positional bias. A representative manifestation of this issue is the well-known lost-in-the-middle phenomenon (Liu et al., 2024b), where models tend to over-attend to tokens near the beginning and the end of the input, while relatively neglecting information located in the middle.

Several approaches have been proposed to address the position bias problem by combining multiple RoPEs with different bases or scaling factors (Chen et al., 2023b; Zhang et al., 2024; Lin et al., 2024). Chen et al. (2023b) observed that RoPE with different bases induces attention troughs at specific positions, which impairs the model’s ability to capture the corresponding content. To mitigate this, they introduced a method, named Attention Buckets, that combines multiple RoPEs with different bases to achieve a more balanced attention distribution. Similarly, Lin et al. (2024) proposed an MoICE method that assigns multiple RoPE bases to each attention head and aggregates the outputs through a weighted sum. However, these methods rely heavily on manually designed rules to determine scaling factors or base values, and require multiple forward passes during inference—one for each specific base or scaling factor—followed by ensembling the results. Although some operations can be parallelized, this procedure inevitably increases inference time and computational cost.

Varying RoPE bases across the entire model can be seen as model-level ensembling, while applying multiple bases to individual attention heads corresponds to module-level ensembling (Figure 1). Model-level ensembling requires multiple model inferences, incurring substantial computational overhead, whereas module-level scaling suffers from a large search space due to fine-grained granularity, limiting the applicability of automatic search algorithms. To balance efficiency and flexibility, we apply multiple scaled RoPEs at the layer level, achieving competitive or

\*Equal contribution.

†Corresponding Author.

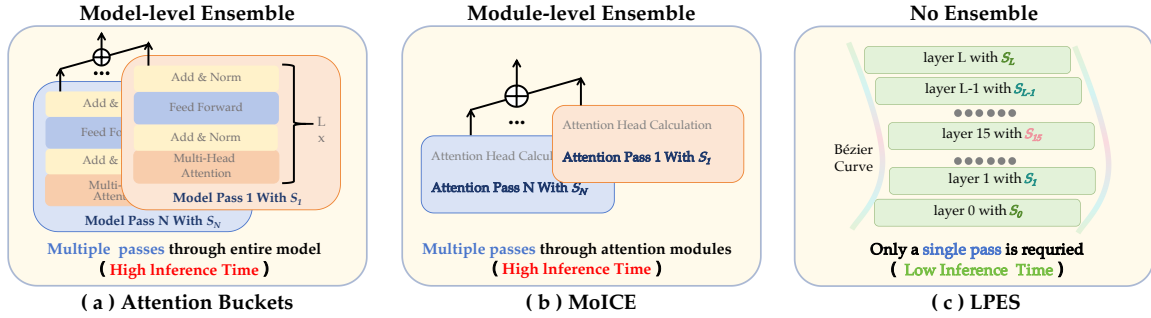


Figure 1: Comparison of the proposed LPES with two representative existing methods. (a) Attention Buckets combines multiple RoPEs with different bases through model parallels. (b) MoICE assigns multiple bases to each attention head. Unlike these existing methods which require multiple forward passes during inference, our LPES (c) achieves superior performance with a single forward pass, significantly reducing inference time.

superior performance with a single forward pass, thus avoiding the associated inference overhead.

Choosing an appropriate scaling factor for each layer is still a non-trivial problem. Let  $L$  denote the number of layers in a transformer-based network, and  $M$  the number of possible values for the scaling factors; the total number of combinations is  $M^L$ , which makes an exhaustive search computationally intractable. Determining optimal scaling factors is inherently a combinatorial optimization problem, and thus cannot be easily solved by gradient-based methods. To overcome this, we leverage the Bézier curve, which defines a smooth, continuous mapping between layer depth and scaling factors using a small set of discrete control points. Letting  $C$  denote the number of control points, the search space is reduced to  $(M \times L)^C$ . In addition to reducing the search space, we find that the smoothness of curve-based scaling preserves layer-wise representational structure and serves as a beneficial inductive bias. We further develop a curve-constrained genetic algorithm to solve this combinatorial optimization problem. By restricting the search space to Bézier curves, we can efficiently optimize layer-specific scaling factors, typically within 3 to 4 hours using only a few hundred examples (e.g., 200 instances) on four H100 GPUs. In long-text tasks, our method introduces no additional inference latency while delivering superior performance over existing approaches.

This study makes the following contributions:

- We propose a layer-specific positional embedding scaling method, termed LPES, which effectively mitigates the position bias without incurring additional inference latency. LPES achieves significant speedups,  $2.42 \times$  faster than MoICE (Lin et al., 2024) and  $1.45 \times$  faster than Ms-PoE

(Zhang et al., 2024), while also improving the model’s ability to handle long-context tasks.

- We introduce an efficient genetic search algorithm in which the search space is constrained by Bézier curves, enabling rapid optimization of layer-specific scaling factors using only a small set of examples.
- Extensive experiments on multiple benchmark datasets demonstrate that our method preserves the model’s general capabilities while producing a more balanced attention distribution without costly fine-tuning, making it broadly applicable across different models and tasks.

## 2 Related Work

Chen et al. (2023b) observed that RoPE with different bases can produce attention troughs at specific positions, which is called “Attention Waves”, thereby impairing the model’s ability to capture the relevant content. To address this, their “Attention Buckets” method integrates multiple RoPE bases through model-parallel inference to achieve a more uniform attention distribution. Zhang et al. (2024) suggested that the long-term decay in attention may contribute to the position bias, and proposed Ms-PoE that assigns distinct scaling factors to attention heads based on their relative sensitivity to positional information. MoICE (Lin et al., 2024), building on the work of Chen et al. (2023b), employs gradient descent to learn the weights for combining results from different bases at the level of individual attention heads. However, a major limitation of these approaches is their high computational cost and inference latency. Specifically, Attention Buckets requires multiple forward passes, while both Ms-PoE and MoICE require re-

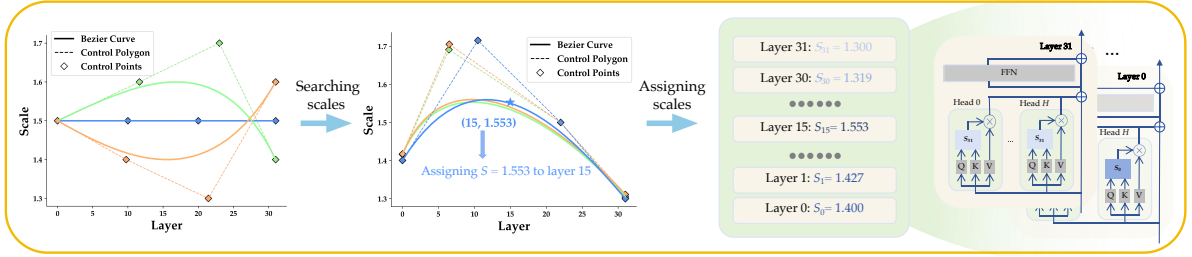


Figure 2: Illustration of the proposed layer-specific positional embedding scaling (LPES) method. Left: Bézier curves can represent a wide variety of shapes. Middle: An optimized Bézier curve found by our search algorithm, which defines a smooth, continuous curve using a limited set of discrete control points. Right: The relationship between the scaling factors and the optimized Bézier curve, and their application within the attention mechanism of a transformer-based network.

peated attention computations to integrate multi-scale RoPE information. They also rely on heuristic or hand-crafted rules to select bases or scaling factors. By contrast, our method achieves superior performance with a single forward pass and proposes an automatic search algorithm, which effectively determines optimal scaling factors using only a few hundred examples.

### 3 Method

#### 3.1 Problem Definition

In this study, we focus on RoPE, which is defined as follows:

$$\langle f(\mathbf{q}, i), f(\mathbf{k}, j) \rangle = \mathbf{q}^T R(i - j) \mathbf{k} \quad (1)$$

where  $f(\mathbf{x}, i)$  denotes a position-dependent rotation applied at position  $i$  to the query  $\mathbf{q}$ , and  $f(\mathbf{k}, j)$  represents the RoPE-rotated key at position  $j$ . The notation  $\langle \cdot, \cdot \rangle$  denotes the inner product between the two position-aware vectors, and  $R(\Delta)$  is the rotation corresponding to the relative offset  $\Delta = i - j$ . This equation shows that the inner product depends only on the vectors  $\mathbf{q}$ ,  $\mathbf{k}$ , and the relative distance between them. Chen et al. (2023a) showed that the context window can be extended by applying a scaling factor  $s$  to the position index as follows:

$$f'(\mathbf{x}, i) = f(\mathbf{x}, i/s) \quad (2)$$

We further show that the scaling factors can mitigate long-term decay and induce diverse attention patterns (Appendix A). Accordingly, our goal is to search for a unique scaling factor  $s$  for each layer to combine information from multiple scaled RoPEs, alleviating long-term decay and attention wave effects, and thereby reducing positional bias.

We model layer depth and scaling factors using Bézier curves, which drastically reduce the search space by determining all layer scales from a few control points. The details are analyzed in Appendix B. Furthermore, in Section 4.2, we show that the smooth and continuous nature of curve-based modeling preserves layer-wise representational structure. Brute-force search demonstrates that smooth scaling naturally emerges as a high-performing configuration, highlighting continuity across layers as a beneficial inductive bias.

As illustrated in Figure 2, a Bézier curve can be viewed as a smooth curve that connects all the scaling factors in a two-dimensional plane. The problem of selecting scaling factors for all layers can then be transformed into searching for an appropriate Bézier curve. Fortunately, Bézier curves can model a wide variety of shapes using only a small set of discrete control points, which significantly reduces the search space. A Bézier curve of degree  $d$ , with  $d + 1$  control points, is defined as follows (Mortenson, 1999):

$$B(t) = \sum_{k=0}^d b_k^d(t) P_k, \quad 0 \leq t \leq 1. \quad (3)$$

where  $t$  is the parametric coordinate controlling a point's position along the curve,  $P_k$  are the control points for the curve, and  $b_k^d$  are the Bernstein basis polynomials, which are defined as:

$$b_k^d(t) = \frac{d!}{k!(d-k)!} t^k (1-t)^{d-k}, \quad k = 0, \dots, d. \quad (4)$$

Once a Bézier curve is determined, the scaling factor  $s_h$  for layer  $h$  can be computed as follows:

$$s_h = \text{proj}_y [B(t(x_h))] \quad (5)$$

where the notation  $\text{proj}_y[\cdot]$  denotes the operation of extracting the  $y$ -coordinate of a two-dimensional point. The function  $t(\cdot)$  maps  $x_h$  to the corresponding parameter  $t$  (see Appendix D), where  $x_h$  represents the position of layer  $h$  within the evenly spaced  $x$ -coordinates defined by the minimum and maximum values of the control points. The value of  $x_h$  can be computed by:

$$x_h = P_0^x + \frac{P_d^x - P_0^x}{L - 1} \cdot h, \quad h = 0, \dots, L-1. \quad (6)$$

where  $L$  denotes the number of layers in a network, and  $P_t^x$  is the  $x$ -coordinates of the  $t$ -th control point for the Bézier curve.

Given a training dataset  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$  consisting of  $N$  examples, where  $x_i$  is an input to the large language model and  $y_i$  is the corresponding ground-truth output, our goal is to maximize the following function:

$$\mathcal{L}_{\mathcal{D}}(\theta) = \frac{1}{N} \sum_{i=1}^N \mathbb{I}\{\text{LLM}(x_i, \theta) \simeq y_i\} \quad (7)$$

where  $\theta = (P_0, \dots, P_d)$  denotes the set of control points defining a Bézier curve of degree  $d$  (each control point  $P_k$  is a two-dimensional point),  $\text{LLM}(x_i, \theta)$  denotes the output of a language model given input  $x_i$ , with all scaling factors determined according to Equation (5) based on the Bézier curve specified by  $\theta$ , and  $\mathbb{I}\{\cdot\}$  is an indicator function with binary output 0 or 1. We constructed the training dataset such that the content containing information useful for generating correct answers appears at varying positions within the input, thereby encouraging the model to distribute its attention more evenly across the input.

### 3.2 Optimization Algorithm

We can regard  $\theta = (P_0, \dots, P_d)$  as a set of newly introduced hyper-parameters that influence the behavior of an LLM. Each  $P_k$  is a two-dimensional vector whose  $x$ - and  $y$ -coordinates can take multiple values. Even though Bézier curves of degree  $d = 3$  which have  $d + 1 = 4$  control points, are capable of representing a wide variety of curves, selecting suitable control points is a combinatorial optimization problem that is difficult to solve using gradient-based methods (Appendix C). Due to the high complexity of the search space, a brute-force approach for determining the scaling factors across layers is intractable; instead, we employ a

specialized genetic algorithm to optimize the control points of the Bézier curves.

In our genetic algorithm, each individual is represented as  $(P_0^x, P_0^y, \dots, P_d^x, P_d^y)$ , where  $P_k^x$  and  $P_k^y$  denote the  $x$ - and  $y$ -coordinates of the  $k$ -th control point, and corresponds to a specific Bézier curve. The initial population is constructed as follows. First, we initialize an individual in which  $k$ -th control point is generated by:

$$(k(L - 1)/d, 1.5), \quad k \in \{0, \dots, d\} \quad (8)$$

where  $L$  is the number of layers in a network. Based on the empirical results reported by Zhang et al. (2024), we set the  $y$ -coordinate values of all control points to 1.5. Subsequently, the remaining individuals are generated by applying a mutation operator (described below) to this initial individual until the population reaches the predefined size.

The fitness of an individual  $\theta = (P_0^x, P_0^y, \dots, P_d^x, P_d^y)$  is evaluated by configuring the layer-wise scaling factors of an LLM according to  $\theta$ , running the LLM on a dataset  $\mathcal{D}$ , and calculating the resulting score  $\mathcal{L}_{\mathcal{D}}(\theta)$  as defined in Equation (7). When constructing the training dataset, we deliberately vary the position of relevant context within the input, which can generally be categorized into three types: the query-relevant content appears at the beginning, middle, or end of the input sequence. We denote these three corresponding sub-datasets as  $\mathcal{D}_B$ ,  $\mathcal{D}_M$ , and  $\mathcal{D}_E$ , respectively. Considering that original LLMs tend to allocate attention unevenly across different positions, we introduce three weights to reflect the relative importance of these sub-datasets when optimizing the model’s scaling factors. The final fitness of an individual is then computed as  $\lambda_B \mathcal{L}_{\mathcal{D}_B}(\theta) + \lambda_M \mathcal{L}_{\mathcal{D}_M}(\theta) + \lambda_E \mathcal{L}_{\mathcal{D}_E}(\theta)$ , where  $\lambda_B \geq 0$ ,  $\lambda_M \geq 0$ ,  $\lambda_E \geq 0$ , and  $\lambda_B + \lambda_M + \lambda_E = 1$ .

The crossover operator is performed by randomly selecting a pair of individuals with relatively high fitness scores as parents, choosing a single crossover point at random, and exchanging this point between the parents. This process produces two offspring, from which we retain only the one with the higher fitness.

The mutation operator modifies  $P^x$  and  $P^y$  within a specified range to prevent excessive variations in the resulting curve, as shown in Equation (9). Let  $M_x$  and  $M_y$  denote the maximum allowable change for the  $x$ - and  $y$ -coordinate, respectively. After mutation, the  $k$ -th control point

$(\hat{P}_k^x, \hat{P}_k^y)$  of an individual must remain within the following range:

$$\hat{P}_k^x \in \begin{cases} [\max(0, P_k^x - M_x), \min(P_{k+1}^x, P_k^x + M_x)], & k = 0, \\ [\max(P_{k-1}^x, P_k^x - M_x), \min(P_{k+1}^x, P_k^x + M_x)], & 0 < k < d, \\ [\max(P_{k-1}^x, P_k^x - M_x), \min(P_k^x + M_x, L - 1)], & k = d, \end{cases} \quad (9)$$

$$\hat{P}_k^y \in [\max(1, P_k^y - M_y), \min(P_k^y + M_y, 2)], 0 \leq k \leq d. \quad (10)$$

To ensure the smoothness of the curve and prevent undesirable abrupt changes in the scaling factor (Ding et al., 2024), the  $x$ -coordinate values of all control points must increase monotonically. The following condition should therefore be satisfied when performing either crossover or mutation operations. Let  $P_i^x$  and  $P_j^x$  denote the  $x$ -coordinates of the  $i$ -th and  $j$ -th control points, respectively. Their relationship is required to satisfy:

$$0 \leq P_i^x < P_j^x \leq L - 1 \quad \text{if } i < j \quad (11)$$

Offspring that fail to meet the above condition are discarded, and the crossover or mutation process is repeated until the condition is satisfied.

Starting with the initial population, individuals are selected based on their fitness, followed by the application of the crossover and mutation operators. This process is repeated iteratively until the maximum number of generations is reached. The complete process is summarized in Algorithm 1.

## 4 Experiments

The experiments are divided into three parts. First, we evaluate the impact of LPES on context utilization, inference latency, and general capabilities. The results show that LPES improves context utilization while preserving general capabilities and introducing no additional inference latency. Second, we analyze the effectiveness of curve-based modeling for layer-wise scaling factors from the perspectives of inter-layer representational structure and inductive bias. Third, we conduct ablation studies to examine the effects of curve types and the number of control points.

### 4.1 Boosting Context Utilization

**Base Models** We selected three RoPE-based LLMs for our experiments: Vicuna-7B-v1.5 (Chiang et al., 2023), and StableBeluga-7B (Mahan

et al., 2023), each with a 4k-token context window, as well as Qwen2.5-7B (Yang et al., 2024), which supports a 130k-token context window.

**Benchmarks** MDQA (Liu et al., 2024c) is a popular multi-document question answering dataset. The key-value retrieval dataset (Liu et al., 2024c) features unique UUID key-value pairs, ideal for evaluating relevant information extraction. ZeroSCROLLS (Shaham et al., 2023) includes multiple open-ended long-text tasks, with sub-datasets and metrics summarized in Table 14. For closed-ended tasks, L-Eval (An et al., 2023) is used, as outlined in Table 13 (Appendix G). Finally, MMLU (Hendrycks et al., 2020) and C-Eval (Huang et al., 2023) assess generalization ability across various tasks.

**Baselines** Positional Interpolation (PI) uses layer-agnostic scaling factors, which are the mean of the searched layer-wise scaling factors (Chen et al., 2023a). Attention Buckets performs multiple forward passes, each using a different RoPE base, and then aggregates the information from these passes (Chen et al., 2023b). Ms-PoE assigns scaling factors ranging from 1.2 to 1.8 to attention heads based on their sensitivity to relevant information (Zhang et al., 2024). Building on the work of Chen et al. (2023b), MoICE computes attention scores using seven different RoPE bases and then performs a weighted sum of these scores using learned weights (Lin et al., 2024).

**Experimental Setup** For LLMs with a 4k-token context window, we use 10 MDQA documents or 50 key-value pairs as context. To evaluate positional bias under longer contexts, Qwen2.5-7B is provided with 20 MDQA documents or 150 key-value pairs, and its accuracy is measured as the ground-truth information appears at different positions within the context. For ZeroSCROLLS and L-Eval, the context window is set to 3,584 tokens, with a maximum of 512 decoded tokens. We additionally report the performance of LPES under a 16K context window in Appendix H. To assess generalization, scaling factors learned on MDQA are transferred to ZeroSCROLLS and L-Eval, with additional evaluation on the MMLU and C-Eval benchmarks to measure generalization ability.

During optimization,  $\lambda_B$ ,  $\lambda_M$ , and  $\lambda_E$  are set to 0.2, 0.3, and 0.5, respectively, with detailed analysis provided in Appendix E. Layer-wise scaling factors are learned by searching the control points

Models	Methods	0%	25%	50%	75%	100%	Avg.	0%	20%	40%	60%	80%	100%	Avg.
		MDQA						Key-Value Retrieval						
Vicuna-7B-v1.5	Baseline	70.4	58.0	55.4	55.4	60.4	59.9	95.2	71.6	81.0	79.0	77.4	73.4	80.9
	Positional Interpolation	71.2	59.6	58.8	56.4	56.2	60.4	98.6	92.8	83.8	90.0	85.8	83.0	89.0
	Attention Buckets	<b>72.6</b>	61.4	60.6	60.8	59.6	63.0	<b>100</b>	<b>94.6</b>	88.6	91.6	87.6	65.8	88.0
	Ms-PoE	<b>72.6</b>	61.4	61.8	<b>62.0</b>	59.0	63.5	95.2	63.2	84.8	91.6	87.4	77.8	83.3
	MoICE	71.6	61.2	60.6	60.8	<b>62.4</b>	63.3	<b>100</b>	93.2	<b>90.2</b>	87.4	89.4	70.0	88.4
	LPES (Ours)	<b>71.4</b>	<b>62.2</b>	<b>62.0</b>	61.0	61.6	<b>63.6</b>	99.4	92.8	<b>87.8</b>	<b>93.6</b>	<b>90.4</b>	<b>88.8</b>	<b>92.1</b>
StableBeluga-7B	Baseline	67.8	59.2	59.6	59.4	68.2	62.8	90.2	34.2	44.0	16.6	59.8	79.4	54.0
	Positional Interpolation	<b>69.6</b>	58.6	58.2	60.0	65.4	62.4	95.2	53.6	31.8	28.6	61.6	83.6	59.1
	Attention Buckets	69.2	59.0	59.8	59.2	67.4	63.0	<b>100</b>	79.8	54.4	58.2	68.4	89.2	75.6
	Ms-PoE	68.4	57.0	60.2	<b>61.0</b>	68.4	63.0	90.2	27.2	27.6	<b>59.4</b>	70.4	89.0	60.6
	MoICE	67.4	60.0	60.2	60.0	<b>68.6</b>	63.2	99.8	71.2	52.2	54.8	<b>74.4</b>	91.4	74.0
	LPES (Ours)	<b>68.8</b>	<b>60.0</b>	<b>60.8</b>	<b>61.0</b>	68.2	<b>64.5</b>	99.2	<b>82.4</b>	<b>57.2</b>	56.2	70.4	<b>89.6</b>	<b>75.8</b>
Qwen2.5-7B	Baseline	69.4	61.0	62.6	58.6	63.6	63.0	99.8	88.6	92.6	90.6	99.0	99.2	95.0
	Positional Interpolation	68.6	62.0	62.2	58.4	64.0	63.0	<b>100</b>	93.2	91.2	88.6	98.6	99.0	95.1
	Attention Buckets	69.6	62.2	63.0	60.2	62.0	63.4	<b>100</b>	89.2	91.4	91.6	98.2	99.2	94.9
	Ms-PoE	69.4	61.8	63.4	60.2	61.4	63.2	<b>100</b>	94.2	91.2	93.6	98.0	99.2	96.0
	MoICE	68.4	61.2	63.0	61.0	63.8	63.5	99.8	88.0	92.6	91.6	99.0	<b>99.4</b>	95.1
	LPES (Ours)	<b>69.6</b>	<b>64.8</b>	<b>69.2</b>	<b>63.0</b>	<b>65.4</b>	<b>66.4</b>	99.8	<b>97.4</b>	<b>93.2</b>	<b>94.0</b>	<b>99.2</b>	99.2	<b>97.1</b>

Table 1: Comparison of accuracy across varying positions of relevant information (e.g., 50% denotes the middle) against established baselines. LPES consistently exceeds all baseline performance, validating its efficacy in neutralizing positional bias.

of cubic Bézier curves using 200 samples from the MDQA or key-value retrieval datasets, and are evaluated on 500 held-out samples per dataset.

**Result Analysis** *Layer-specific positional embedding scaling greatly mitigates position bias.* Table 1 shows that LPES consistently outperforms the baselines in average performance across different positions, notably boosting Vicuna’s average accuracy by 11.2% in key-value retrieval. LPES demonstrates strong transferability when applying MDQA-optimized scaling factors to Zero-SCROLLS and L-Eval (Table 2). The results confirm that these factors generalize robustly across diverse models and long-text tasks. Furthermore, the results on longer context windows and larger model scales (detailed in Appendix H) further validate the applicability of LPES. Additionally, LPES preserves the model’s general capabilities, as shown in Table 3. Notably, the scaling factors are treated as hyperparameters rather than trainable model parameters. Their adjustment is therefore considered optimization rather than training. In machine learning, training typically refers to updating model weights and biases using gradient-based methods. In contrast, our approach does not modify model parameters and can thus be regarded as a training-free method. This property avoids catastrophic forgetting (De Lange et al., 2021) caused by large-scale parameter updates and makes the method particularly suitable for already deployed models.

*LPES yields a more balanced attention distribution without additional inference cost.* Ms-PoE and MoICE are sample-dependent, their scaling factors cannot be precomputed and must be determined for each input. Specifically, Ms-PoE entails an additional attention pass to assess head sensitivity, whereas MoICE requires parallel computations across seven RoPE bases alongside serial routing weight calculations.

To demonstrate the advantage in inference efficiency, we sample 500 examples from the MDQA dataset and report the average inference time of Vicuna on a single H100 GPU. For a fair comparison, FlashAttention-2 (Dao, 2023) was used as the attention backend for all methods. As shown in Table 4, LPES is roughly  $1.45\times$  faster than Ms-PoE and  $2.42\times$  faster than MoICE.

## 4.2 Motivation for Curve-Based Modeling

**Preserved Representational Structure** The smooth and continuous nature of the Bézier curve enforces gradual variations in scaling across layers, which helps preserve the coherence of the model’s layer-wise representational structure. We compare against several intuitive baselines: *uniform scaling*, where all layers share a single scaling factor which is the mean of the searched layer-wise scaling factors; *noisy Bézier scaling*, which adds independent uniform noise sampled from  $\mathcal{U}(-0.1, 0.1)$  to each layer’s Bézier-derived scale; *shuffled scaling*, which randomly permutes the layer-wise scaling factors while preserving

Model	Method	Open-ended Long-Text Tasks								Closed-ended Long-Text Tasks				
		GovRpt	Qasper	SumScrFd	Qmsum	NarrQA	Squality	SpeDgst	Avg.	Coursera	QUALITY	TOEFL	SFiction	Avg.
Vicuna-7B-v1.5	Baseline	18.44	22.82	<b>18.42</b>	14.50	10.98	16.56	21.39	16.91	37.21	38.12	38.00	57.90	42.81
	MoICE	<b>22.29</b>	32.34	13.31	14.79	<b>13.61</b>	16.22	<b>22.60</b>	19.30	<b>42.35</b>	<b>43.71</b>	39.33	57.20	<b>45.65</b>
	LPES (Ours)	21.47	<b>33.37</b>	14.39	<b>15.53</b>	11.52	<b>16.91</b>	22.24	<b>19.35</b>	40.41	42.57	<b>40.67</b>	<b>58.20</b>	45.46
Qwen2.5-7B	Baseline	24.76	22.92	14.69	16.25	9.78	14.85	53.66	22.42	45.47	62.43	66.00	60.87	58.69
	MoICE	25.56	23.51	15.12	<b>23.19</b>	10.64	<b>16.92</b>	<b>53.81</b>	24.11	48.13	64.28	67.33	66.00	61.44
	LPES (Ours)	<b>27.56</b>	<b>23.91</b>	<b>16.18</b>	<b>23.19</b>	<b>11.97</b>	14.92	<b>53.81</b>	<b>25.51</b>	<b>48.51</b>	<b>66.43</b>	<b>69.28</b>	<b>66.42</b>	<b>62.66</b>

Table 2: Performance comparison on **open-ended** and **closed-ended** long-text benchmarks. Open-ended tasks are reported on the left, while closed-ended tasks are shown on the right.

Model	Method	MMLU	C-Eval
Vicuna-7B-v1.5	Baseline	49.90	49.42
	LPES (Ours)	49.00	49.33
StableBeluga-7B	Baseline	51.50	34.78
	LPES (Ours)	51.30	34.63

Table 3: General capability of models equipped with LPES on MMLU and C-Eval datasets.

Method	Inference time per sample (s)
Baseline	0.71
Attention Buckets	3.38
Ms-PoE	1.03
MoICE	1.72
LPES (Ours)	<b>0.71</b>

Table 4: Comparison of inference efficiency between LPES and baseline methods.

their overall distribution; and *fully random scaling*, where each layer independently samples its scale from  $\mathcal{U}(1, 2)$ .

Inspired by RSA (Kriegeskorte et al., 2008), which measures representational structure across input samples at a fixed layer using representational similarity matrices (RSM), we focus instead on the *layer dimension*. Specifically, we construct a layer-wise RSM by computing pairwise dot-product similarities between the last-token hidden representations of different layers. Given the set of hidden states  $\{\mathbf{H}_l\}_{l=1}^L$ , where  $\mathbf{H}_l \in \mathbb{R}^d$  denotes the representation at layer  $l$ , the entries of the RSM are defined as:

$$\mathbf{RSM}_{ij} = \mathbf{H}_i^\top \mathbf{H}_j, \quad 1 \leq i, j \leq L, \quad (12)$$

where  $\mathbf{RSM}_{ij}$  quantifies the similarity between the  $i$ -th and  $j$ -th layers. This matrix captures the global structural organization of representations across the model’s depth. We quantify representational stability via the *representational structure deviation*  $\mathcal{D}$ , which measures the average absolute difference between the RSM of a perturbed model ( $\mathbf{RSM}^p$ ) and that of a vanilla configuration with-

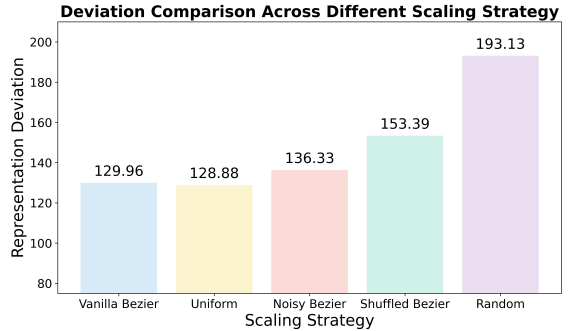


Figure 3: Comparison of representational structure deviation under different scaling strategies. Vanilla B’ezier curve achieves smaller representational deviation while effectively integrating information from multiple base RoPE configurations.

out scaling ( $\mathbf{RSM}^v$ ) as follows:

$$\mathcal{D} = \frac{1}{L^2} \sum_{i=1}^L \sum_{j=1}^L \left| \mathbf{RSM}_{i,j}^p - \mathbf{RSM}_{i,j}^v \right|, \quad (13)$$

This metric captures global changes in inter-layer relationships across layers, rather than at individual layers. Experiments are conducted using Vicuna-v1.5-7B on 500 randomly sampled MDQA examples, with results averaged over 16 random seeds. As shown in Figure 3, the smooth Bézier-based scaling consistently yields smaller structural deviations, indicating minimal disruption to the model’s internal representations.

**Empirical Convergence Behavior** We conduct a brute-force genetic algorithm search using the fitness function defined above over 200 randomly sampled MDQA inputs, where each individual represents a set of layer-wise scaling factors, initialized uniformly from  $\mathcal{U}(1, 2)$  to avoid introducing any prior smoothness bias. During the search, we track both performance—measured by the average fitness of the best 1, 2, 4, and 8 individuals—and smoothness, quantified by a second-

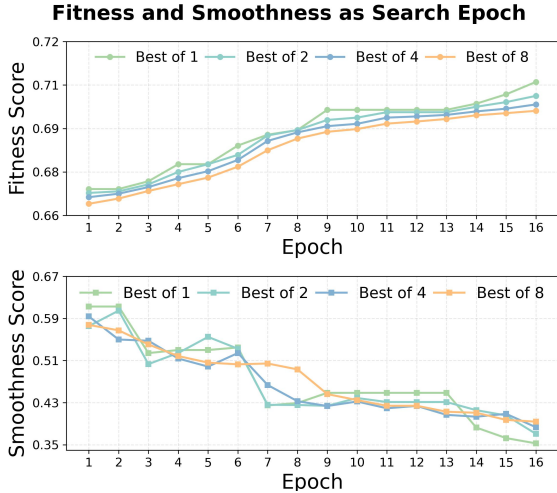


Figure 4: Trend of fitness and smoothness over brute-force search epochs. The population progressively evolves toward smoother scaling factors across layers, indicating that the smoothness of curves serves as an effective inductive bias.

order metric:

$$\mathcal{S} = \frac{1}{L-2} \sum_{l=2}^{L-1} \|s_{l+1} - 2s_l + s_{l-1}\|_2, \quad (14)$$

where  $L$  is the total number of layers, and  $s_l$  denotes the scaling factor at layer  $l$ . Here,  $\mathcal{S}$  quantifies the local curvature, with smaller values indicating smoother transitions. As shown in Figure 4, higher-performing configurations consistently exhibit lower  $\mathcal{S}$ , indicating that smooth variation emerges during the search. This finding suggests that smoothness constitutes a beneficial inductive bias, motivating the use of smooth curve modeling—such as Bézier curves—to efficiently parameterize high-performing scaling configurations.

### 4.3 Ablation Studies

In this section, we present three ablation studies using Vicuna-v1.5-7B on MDQA. First, we demonstrate that Bézier curves outperform alternative curves in determining layer-specific scaling factors. Next, we examine the impact of control point counts on convergence quality and speed. We further provide an ablation study on the hyperparameter  $\lambda$  in Appendix E.

**Curve Type** Bézier curves provide a compact, low-dimensional parameterization capable of approximating a wide variety of curve shapes (Nuntawisuttiwong and Dejdumrong, 2021). To demonstrate the advantages of Bézier curve modeling, we consider two alternative approaches with

Method	0%	25%	50%	75%	100%	Average
Baseline	70.4	58.0	55.4	55.4	60.4	59.92
LPES (Linear Curve)	<b>71.8</b>	61.0	<b>62.2</b>	60.0	60.6	63.12
LPES (Step Curve)	71.6	60.2	59.4	59.2	60.4	62.16
LPES (Bézier curve)	71.4	<b>62.2</b>	62.0	<b>61.0</b>	<b>61.6</b>	<b>63.64</b>

Table 5: Performance comparison of different curve types for determining layer-wise scaling factors. Bézier curves achieve superior performance.

the same number of control points: linear interpolation between control points and step-function modeling based on these control points. Although these alternatives differ in their curve formulations, they also serve as layer-specific scaling strategies within our framework. While linear interpolation offers slightly higher computational efficiency, which can be neglected in the search procedure (Appendix E), we ultimately adopt Bézier curves due to their superior performance. As shown in Table 5, Bézier curves outperform other curve-fitting methods, and the minor additional cost required to determine the scaling factors is fully offset by the inference-time performance gains.

**Number of Control Points** Setting the maximum iterations to 20, we vary the number of control points to evaluate performance—measured by the mean and variance of accuracy across positions—and convergence speed. While more control points improve Bézier curve fitting precision and the likelihood of finding optimal scaling factors, they also expand the search space, slowing convergence. As shown in Table 6, using four control points provides a favorable trade-off between performance and convergence speed. In contrast, brute-force search shows little tendency to converge within the limited number of iterations, further demonstrating the efficiency of our curve-constrained genetic algorithm.

Control Points	Accuracy (Std)	Epochs to Convergence
Baseline	59.9 ( $\pm 5.56$ )	--
Brute-Force	60.2 ( $\pm 4.69$ )	20
2	60.6 ( $\pm 4.55$ )	3
3	62.2 ( $\pm 3.97$ )	5
4	63.6 ( $\pm 3.90$ )	9
5	<b>63.8 (<math>\pm 3.87</math>)</b>	16

Table 6: Performance comparison of different numbers of control points. Additional points improve accuracy and reduce positional bias, but slow convergence due to the increased computational cost of optimization. Overall, experimental results indicate a performance insensitivity to the number of control points used.

## 5 Conclusion

We present layer-specific positional embedding scaling (LPES), a method that mitigates position bias in transformer-based LLMs by assigning distinct scaling factors to each layer, achieving balanced attention over input without fine-tuning or extra latency. Optimal scaling factors are efficiently identified via a Bézier-constrained genetic algorithm, reducing the search space and converging with only a few hundred examples. Experiments show that LPES consistently improves long-context performance, preserves general capabilities, and requires only a single forward pass, achieving up to  $2.42\times$  speedup over MoICE and  $1.45\times$  over Ms-PoE, which makes LPES a broadly applicable and efficient solution.

## Limitations

In this work, we adopt a training-free strategy that assigns different scaling factors across layers to encourage a more balanced attention distribution. While this design enables straightforward and efficient deployment, we do not explore the behavior of our method in training-based settings. Extending the proposed approach to training or fine-tuning pipelines could potentially yield further gains, which we leave for future work. Nevertheless, this limitation does not diminish the practical effectiveness or applicability of our method in real-world scenarios.

## References

- Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *arXiv preprint arXiv:2307.11088*.
- Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.
- Yuhan Chen, Ang Lv, Ting-En Lin, Changyu Chen, Yuchuan Wu, Fei Huang, Yongbin Li, and Rui Yan. 2023b. Fortify the shortest stave in attention: Enhancing context awareness of large language models for effective tool use. *arXiv preprint arXiv:2312.04455*.
- Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhonghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%\* chatgpt quality. See <https://vicuna.lmsys.org> (accessed 14 April 2023), 2(3):6.
- Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.
- Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.
- Yiran Ding, Li Lina Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*.
- Xiachong Feng, Xiaocheng Feng, and Bing Qin. 2021. A survey on dialogue summarization: Recent advances and new frontiers. *arXiv preprint arXiv:2107.03175*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.
- Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, and 1 others. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in Neural Information Processing Systems*, 36:62991–63010.
- Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bannettini. 2008. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:249.
- Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*.
- Hongzhan Lin, Ang Lv, Yang Song, Hengshu Zhu, Rui Yan, and 1 others. 2024. Mixture of in-context experts enhance llms’ long context awareness. *Advances in Neural Information Processing Systems*, 37:79573–79596.
- Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2024a. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024b. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024c. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Dakota Mahan, Ryan Carlow, Louis Castricato, Nathan Cooper, and Christian Laforte. 2023. [Stable beluga models](#).
- M.E. Mortenson. 1999. *Mathematics for Computer Graphics Applications*. G - Reference, Information and Interdisciplinary Subjects Series. Industrial Press.
- Taweechai Nuntawisuttiwong and Natasha Dejdumrong. 2021. An approximation of bézier curves by a sequence of circular arcs. *Information Technology and Control*, 50(2):213–223.
- Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. Zeroscrolls: A zero-shot benchmark for long text understanding. *arXiv preprint arXiv:2305.14196*.
- Ning Shang, Li Lina Zhang, Siyuan Wang, Gaokai Zhang, Gilsinia Lopez, Fan Yang, Weizhu Chen, and Mao Yang. 2025. Longrope2: Near-lossless llm context window scaling. *arXiv preprint arXiv:2502.20082*.
- Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *Cornell University - arXiv, Cornell University - arXiv*.
- Qi Sun, Edoardo Cetin, and Yujin Tang. 2025. Transformer<sup>2</sup>: Self-adaptive llms. *arXiv preprint arXiv:2501.06252*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed H Awadallah, Dragomir Radev, and Rui Zhang. 2021. Summ<sup>n</sup>: A multi-stage summarization framework for long input dialogues and documents. *arXiv preprint arXiv:2110.10150*.
- Zhenyu Zhang, Runjin Chen, Shiwei Liu, Zhewei Yao, Olatunji Ruwase, Beidi Chen, Xiaoxia Wu, and Zhangyang Wang. 2024. Found in the middle: How language models use long contexts better via plug-and-play positional encoding. *arXiv preprint arXiv:2403.04797*.
- Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, and 1 others. 2023. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x. corr abs/2303.17568 (2023). *arXiv preprint arXiv:2303.17568*, 10.

## A Long-Term Decay and Attention Wave in RoPE

Zhang et al. (2024) observed that the long-term decay of RoPE causes the model to focus more on the end of a sequence. As the relative distance grows, attention scores drop rapidly, leading the model to overemphasize nearby tokens during autoregressive decoding while neglecting distant ones. To mitigate this issue, they scale RoPE by a factor  $s \geq 1$  (Figure 5), which effectively reduces the relative distance to  $1/s$  of its original value (Figure 6). This adjustment slows the decay rate, enabling the model to attend not only to nearby tokens but also to more distant ones, particularly those in the middle of the sequence.

To demonstrate that scaling RoPE can indeed enhance the model’s attention to middle positions, we use the **Vicuna-7B-v1.5** (Chiang et al., 2023) and **LLaMA-2-7B-hf** (Touvron et al., 2023) which both consist of 32 transformer layers to conduct experiments on the validation dataset of **QM-Sum** (Shaham et al., 2023). We split the context into three parts and calculate the attention scores to the middle-part tokens at different scales. In Figure 7, an increase in the scale factor leads to higher attention scores, demonstrating that scaling RoPE allows the model to focus more on middle-part content during autoregressive decoding.

Chen et al. (2023b) analyze the phenomenon of oscillatory “attention waves” in Transformer models, where attention fluctuates across tokens instead of being smoothly distributed. These oscillations, mainly induced by the mechanisms of RoPE, can cause the model to under-attend to important information located at attention troughs, limiting long-context utilization and potentially introducing instability. To address this issue, the authors propose the *Attention Buckets* approach, which runs multiple model parallels with different bases in RoPE and combines the decoded logits across these bases, producing complementary attention wave patterns. The method enhances the model’s sensitivity to context across all positions.

## B Search Space and Time Complexity Analysis

We follow Ding et al. (2024), discretizing the continuous search space to enable more efficient searching. Assume the control points of the Bézier curve are  $(P^x, P^y)$ , where  $P^x \in [0, L - 1]$  ( $L$  is the number of scaled layers) and  $P^y \in [1, 2]$ .

The values of  $P^x$  are discretized with a step size of 1, and the values of  $P^y$  are discretized with a step size of 0.1. Given that the model consists of 32 layers, there are 32 possible selections in  $P^X$ , while the scaling factor chosen from the  $P^Y$  set offers 11 options as shown in Table 7. The total number of choices for the brute-force search is  $11^{32}$ . If a Cubic Bézier curve is used, each control point has  $32 \times 11$  possible combinations. With four control points, the total search space is  $352^4$  which approximately narrows the search space by a significant factor  $10^{20}$  compared to the brute-force search.

Coordinate	Search Space
$P^x$	{0, 1, 2, 3, 4, 5, 6, 7, 8, ..., n - 4, n - 3, n - 2, n - 1}
$P^y$	{1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0}

Table 7: Search space for the control point of Bézier curves.

In our method, the dominant cost of the genetic algorithm arises from evaluating the fitness function, which requires running model inference to assess the effectiveness of different scaling factors. In contrast, the computational overhead of other GA operations—such as assignment, mutation, and crossover—is negligible. Using  $4 \times \text{H100}$  GPUs, we measured the per-epoch time cost of each operation as follows:

Operation Type	Time (s)
Assigning scaling factor from curve	5.2
Mutation	4.5
Crossover	2.3
Computing fitness via model inference	1167.4

Table 8: Measured runtime per epoch of each operation in the genetic algorithm when using  $4 \times \text{H100}$  GPUs. Model inference dominates the total cost.

Assume the algorithm runs for at most  $M$  epochs and generates  $N$  new individuals per epoch, and the search uses  $S$  samples. Each individual requires three inference runs (placing the correct document at different positions). Thus, the total number of inference calls is  $3NMS$ . In practice, we perform data-parallel inference using  $N_{\text{card}}$  GPUs with batch size  $B$ , which reduces the effective runtime to  $O((3MNS)/(N_{\text{card}} \cdot B))$ .

## C Limitations of Gradient-Based Methods

We also attempted to determine the layer-specific scaling factors using gradient descent, but ob-

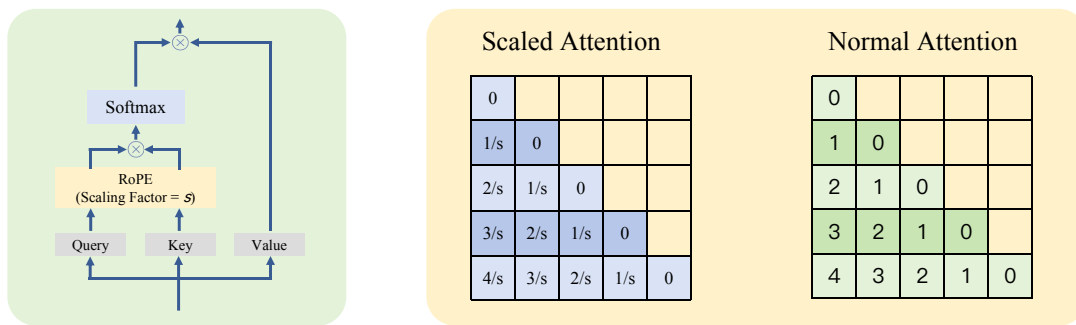


Figure 5: We obtain multi-scale RoPE by scaling the positional indices.

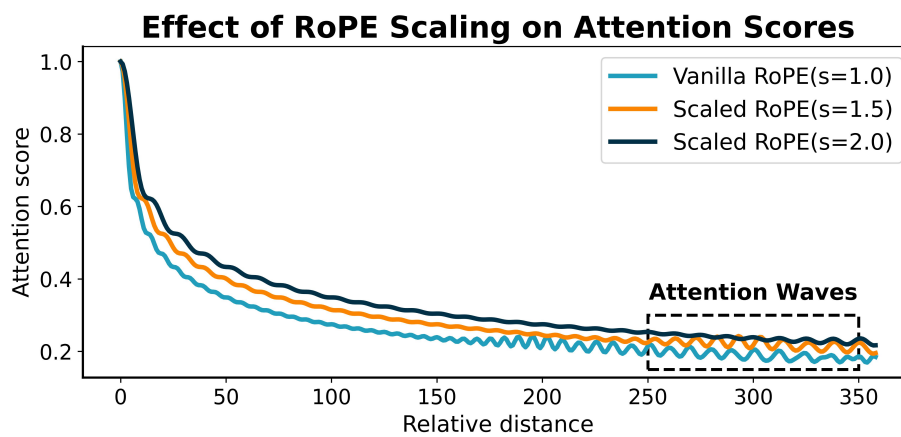


Figure 6: The rapid decay of RoPE prioritizes local focus, and the attention waves may cause the model to overlook crucial information at attention troughs, whereas the scaling operation can slow this decay and generate diverse wave patterns.

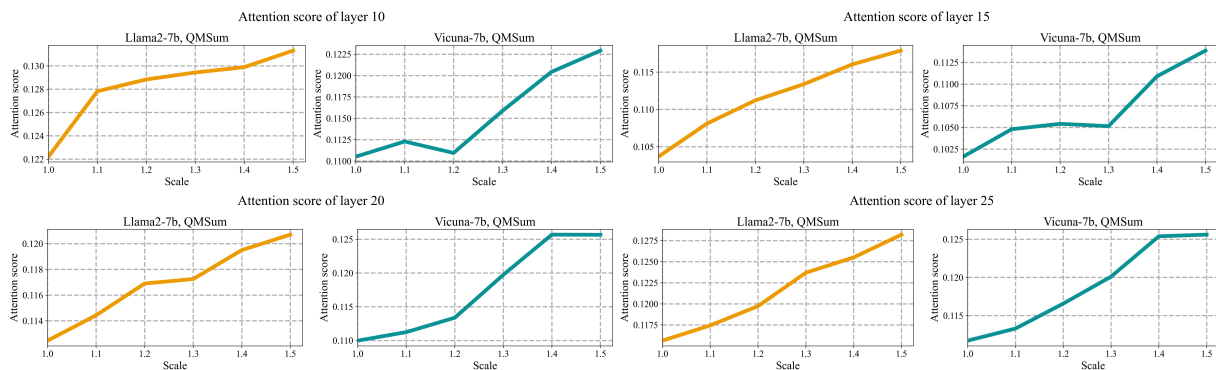


Figure 7: The attention score to the middle part across some layers. The scaling operation can enhance the model's attention to middle positions.

Hyperparameter	Value	Description
Population_size $N_{ps}$	32	Number of individuals in initial population generation.
Parents_size $N_{pa}$	12	Number of individuals selected as parents.
Max_epoch $T$	20	Maximum number of generatios.
Mutation_numbers $N_{mu}$	16	Number of offspring generated through mutation.
Crossover_numbers $N_{cr}$	4	Number of offspring generated through crossover.
Max_crossover_try $N_{ct}$	4	Maximum attempts allowed to produce valid offspring during crossover.
$M_x$	2	Perturbation magnitude of the control point’s $x$ -coordinate ( $P^x$ ).
$M_y$	0.2	Perturbation magnitude of the control point’s $y$ -coordinate ( $P^y$ ).

Table 9: Hyperparameter settings of the constrained genetic algorithm

served poor convergence behavior. This may also shed light on why LongRoPE (Ding et al., 2024) and LongRoPE2 (Shang et al., 2025) employ genetic algorithms rather than backpropagation to determine the scaling factors across RoPE dimensions. Although the genetic algorithm incurs higher computational overhead compared to directly optimizing hyperparameters via backpropagation, it consistently converges to a more favorable set of scaling parameters. Furthermore, incorporating Bézier curves significantly accelerates the convergence process.

In the **gradient-based setting**, we construct three datasets from the MDQA, each containing 2,000 samples in which the correct document is placed at a different position (i.e., first, middle, or last). In each epoch, a total of 2,000 samples are drawn from these datasets based on the value of  $\lambda$  as specified in Section §4.1, where a larger  $\lambda$  indicates a higher probability of sampling from the corresponding dataset. For stable training, we use a batch size of 32, a learning rate of  $1e - 5$ , and train the model for a total of 30 epochs.

For the gradient-based method, we observed that even with a large batch size and a small learning rate, the optimization of scaling factors via backpropagation failed to converge. A possible reason is the limited number of trainable parameters (Sun et al., 2025). We evaluated the model at the 30th epoch and found a significant degradation in performance, as shown in Table 10.

#### D Cubic Bézier Curve Parameterization for Layer Assignment

Consider a cubic Bézier curve with four control points:

$$\begin{aligned} P_0 &= (x_0, y_0), & P_1 &= (x_1, y_1), \\ P_2 &= (x_2, y_2), & P_3 &= (x_3, y_3). \end{aligned} \quad (15)$$

Model	Method	0%	25%	50%	75%	100%
Vicuna-7B-v1.5	Baseline	70.4	58.0	55.4	55.4	60.4
	Gradient-Based	67.4	54.0	51.2	52.8	55.8
Qwen2.5-7B	Baseline	69.4	61.0	62.6	58.6	63.6
	Gradient-Based	68.7	56.6	57.6	55.8	57.8

Table 10: Gradient-based methods lead to accuracy degradation in the MDQA dataset.

where the  $x$ -coordinates are strictly increasing since Equation 11:

$$x_0 < x_1 < x_2 < x_3. \quad (16)$$

The parametric form of the cubic Bézier curve is

$$\begin{aligned} x(t) &= (1-t)^3 x_0 + 3(1-t)^2 t x_1 + 3(1-t)t^2 x_2 + t^3 x_3, \\ y(t) &= (1-t)^3 y_0 + 3(1-t)^2 t y_1 + 3(1-t)t^2 y_2 + t^3 y_3, \end{aligned} \quad (17)$$

where  $t \in [0, 1]$ .

Since the  $x_i$  are strictly increasing, the function  $x(t)$  is typically monotonic. This property allows the use of a binary search over the interval  $[0, 1]$  to efficiently find the parameter  $t$  corresponding to any given target value  $x$ , which defines the function  $t(x)$ .

#### E Hyperparameters of the constrained genetic algorithm

In our experiments, we observed that when scaling RoPE, the model tends to improve performance at early positions while neglecting performance at later positions. Consequently, when setting  $\lambda$ , we favor assigning larger weights to later positions. Here, we define  $\langle \lambda_B, \lambda_M, \lambda_E \rangle$  as the weights assigned to the accuracy of the beginning, middle, and end positions, respectively, in the genetic algorithm’s fitness function. In this study, we compare three weighting

schemes:  $\langle 0.333, 0.333, 0.333 \rangle$ ,  $\langle 0.1, 0.3, 0.6 \rangle$ , and  $\langle 0.2, 0.3, 0.5 \rangle$ .

Method	0%	25%	50%	75%	100%	Average
Baseline	70.4	58.0	55.4	55.4	60.4	59.9
$\langle 0.333, 0.333, 0.333 \rangle$	<b>73.2</b>	<b>62.4</b>	60.2	58.8	58.2	62.6
$\langle 0.1, 0.3, 0.6 \rangle$	70.6	60.2	60.8	<b>61.0</b>	<b>62.0</b>	63.0
$\langle 0.2, 0.3, 0.5 \rangle$	71.4	62.2	<b>62.0</b>	<b>61.0</b>	61.6	<b>63.2</b>

Table 11: The impact of hyper-parameters  $\lambda$  on the optimized layer-wise scaling factors, showing that performance is largely insensitive to their choice.

## F Search Algorithm Robustness

In this section, we evaluate the robustness of the scaling factors under variations in the search dataset. On the MDQA dataset, we use Vicuna-1.5-7B and randomly sample 200 training instances to form the search set for each run. Across five independent runs with different search sets, the method achieves an average performance of 63.68 with a sample variance of only 0.027, demonstrating that our approach is highly stable across different search sets. Overall, our method consistently outperforms prior approaches, highlighting the robustness of the proposed search algorithm.

Method	0%	25%	50%	75%	100%	Average
Baseline	70.4	58.0	55.4	55.4	60.4	59.9
Attention Buckets	<b>72.6</b>	61.4	60.6	60.8	59.6	63.0
Ms-PoE	<b>72.6</b>	61.4	61.8	62.0	59.0	63.5
MoICE	71.6	61.2	60.6	60.8	<b>62.4</b>	63.3
LPES (run 1)	71.4	62.2	62.0	61.0	61.6	63.6
LPES (run 2)	71.6	62.4	<b>62.2</b>	60.8	61.8	63.8
LPES (run 3)	71.6	61.8	61.8	62.0	61.0	63.6
LPES (run 4)	<b>72.6</b>	61.0	62.0	<b>63.2</b>	61.0	<b>63.9</b>
LPES (run 5)	72.2	<b>62.8</b>	61.0	61.0	60.4	63.5

Table 12: Performance of LPES across five runs compared with baseline methods. Percentages indicate the relative position of relevant documents in the context.

## G Dataset Details

Dataset	Question Style	Domain	Metric
Coursera	Multiple Choice	Advanced Courses	Accuracy
QuALITY	Multiple Choice	Gutenberg	Accuracy
TOEFL	Multiple Choice	English Test	Accuracy
SFiction	True/False Questions	Scientific Fiction	Accuracy

Table 13: Overview and evaluation metrics of the sub-datasets in L-Eval.

Write a high-quality answer for the given question using only the provided search results (some of which might be irrelevant).

{search\_results}

Question: {question}  
Answer:

(a) MDQA prompt

Extract the value corresponding to the specified key in the JSON object below.

JSON data:  
{formatted\_kv\_records}

Key: "[key]"  
Corresponding value:

(b) KV prompt

Figure 8: Prompt templates used in MDQA and Key-Value Retrieval datasets.

Dataset	Description	Metric
GovReport	Summarization of long reports	ROUGE-1/2/L
SummScreenFD	Summarization of TV show episode scripts	ROUGE-1/2/L
QMSum	Query-based summarization over meeting transcripts	ROUGE-1/2/L
SQuALITY	Question-focused summarization over stories	ROUGE-1/2/L
Qasper	Question answering over research papers	F1
NarrativeQA	Question answering about entire books and movie scripts	F1
SpaceDigest	Aggregated sentiment classification over 50 hotel reviews from Space	Exp_similarity

Table 14: Overview and evaluation metrics of the sub-datasets in ZeroSCROLLS.

## H Effectiveness of LPES on Longer Contexts

We conduct experiments on Vicuna-1.5-13B and Qwen-2.5-7B under a 16k-token context setting on L-Eval to verify the effectiveness of LPES in long-context scenarios. The decoding length is set to 512 tokens, so the maximum usable context window is limited to 15,872 tokens. As shown in Table 15 and 16, the results demonstrate that our method remains effective on larger models and extended context lengths, highlighting its strong scalability and robustness.

Model	Method	Coursera	QuALITY	TOEFL	SFiction	Average
Vicuna-13B-v1.5-16k	Baseline	69.6	51.4	33.3	57.1	52.9
	MoICE	67.4	<b>55.6</b>	35.7	52.6	52.8
	LPES (Ours)	<b>70.6</b>	54.4	<b>36.0</b>	<b>59.4</b>	<b>55.1</b>
Qwen2.5-7B	Baseline	59.8	66.3	76.6	71.8	68.6
	MoICE	59.8	66.3	78.7	<b>73.0</b>	69.5
	LPES (Ours)	<b>63.8</b>	<b>69.1</b>	<b>80.9</b>	72.9	<b>71.7</b>

Table 15: Results under longer-context settings (16k tokens) on the L-Eval benchmark. LPES consistently improves performance over the baseline and MoICE on both Vicuna-13B-v1.5-16k and Qwen2.5-7B, demonstrating strong scalability to larger models and longer context windows.

Model	Method	GovRpt	Qasper	SumScrFd	QMSum	NarrQA	SQuality	SpcDgst	Average
Qwen2.5-7B	Baseline	25.17	23.33	14.23	17.36	6.00	17.60	51.12	22.12
	MoICE	<b>38.77</b>	18.83	15.21	17.84	6.14	18.06	53.63	24.07
	LPES (Ours)	35.77	<b>24.63</b>	<b>15.91</b>	<b>20.84</b>	<b>8.02</b>	<b>19.06</b>	<b>53.77</b>	<b>25.43</b>

Table 16: Performance comparison on ZeroSCROLLS benchmarks with a 16k context length using Qwen2.5-7B. LPES consistently improves average performance over both the baseline and MoICE across diverse tasks.

---

### Algorithm 1 Scaling factor search algorithm

---

**Input:** an LLM  $\mathcal{M}$ , a dataset  $\mathcal{D}$ , population size  $N_{ps}$ , the number of offspring generated by crossover  $N_{cr}$ , the number of mutated individuals  $N_{mu}$ , and maximum number of generations  $T$ .

- 1:  $\mathcal{S}_0 = \text{Initial-Population-Generation}(\mathcal{D}, N_{ps});$  // Randomly generate the initial population.
  - 2: **for**  $i = 1$  to  $T$  **do**
  - 3:   Evaluate-Fitness( $\mathcal{S}_{i-1}, \mathcal{M}, \mathcal{D}$ ); // Evaluate the fitness of all individuals in the population.
  - 4:    $\mathcal{S}_{pa} = \text{Select-Parents}(\mathcal{S}_{i-1});$  // Select the parent pool according to fitness values.
  - 5:    $\mathcal{S}_{cr} = \text{Crossover-Operator}(\mathcal{S}_{pa}, N_{cr});$  // Produce offspring using the crossover operator.
  - 6:    $\mathcal{S}_{mu} = \text{Mutation-Operator}(\mathcal{S}_{pa}, N_{mu});$  // Generate offspring using the mutation operator.
  - 7:    $\mathcal{S}_i = \mathcal{S}_{pa} \cup \mathcal{S}_{cr} \cup \mathcal{S}_{mu};$  // Merge the individuals to form the next generation's population.
  - 8: **end for**
  - 9: Return the individual with the highest fitness in  $\mathcal{S}_T$ .
-