Long-Context Extrapolation of Language Models

Mingjie Lim
Shanghai Jiao Tong University
mjie522@sjtu.edu.cn

Peishuo Wang
Shanghai Jiao Tong University
wangps@sjtu.edu.cn

Xiaojie Cai
Shanghai Jiao Tong University
xiaojie_cai@sjtu.edu.cn

Abstract

With the outstanding performance of large language models (LLMs) across various NLP tasks, enhancing their ability to handle long texts has become a prominent research focus. As a core component of the Transformer architecture, positional encoding directly affects a model's capacity to capture sequence structure, which in turn influences its performance on long-text tasks. This project focuses on the impact of different positional encoding methods on model generalization and explores strategies to extend the context length of language models, aiming to improve performance in tasks such as long document understanding and extended dialogue. We implemented and evaluated long-context extension methods for both absolute positional encoding and Rotary Position Embedding (RoPE). For GPT-2, we adopted a position offset method. For LLaMA3.2-1B and Qwen2.5-0.5B, we implemented linear interpolation and extended the NTK method. On the Leval long-context benchmark dataset, our proposed NTK-C (NTK with frequency dampening) method significantly improved long-sequence processing capabilities, raising the accuracy on the Topic Retrieval task (with a sequence length of 12.5K) from 12.0

1 Contributions

Mingjie Lin: Coding and experiments for the SHAPE part, and partially involved in running the code for the PI section.

Peishuo Wang: Surveying existing positional encoding methods, and in charge of coding and experiments for the PI part.

Xiaojie Cai: The coding and experiments of the NTK part. Proposed improvements to the NTK method and conducted related experiments.

2 Introduction

With the continued advancement of Pretrained Language Models (PLMs), open-source large models such as GPT-2, Qwen2.5-0.5B, and LLaMA3.2-1B have demonstrated remarkable performance across a wide range of natural language processing tasks. Their success can be attributed to powerful modeling capabilities, extensive corpus training, and well-designed architectures. Among these components, the positional encoding mechanism—an essential part of the Transformer architecture—plays a critical role in capturing the relative or absolute positional relationships between tokens in a sequence.

Preprint.

However, the context length of mainstream pretrained models remains limited, making it challenging to directly process extremely long texts, such as lengthy conversations, academic papers, and legal documents. For example, GPT-2 typically supports a maximum context length of 1024 tokens. Although newer models like Qwen2.5-0.5B and LLaMA3.2-1B have extended their context lengths to several thousand tokens, their performance still degrades significantly when processing sequences that are 10 to 100 times longer (e.g., on the order of 100,000 tokens).

To address this challenge, researchers have recently proposed various positional encoding extrapolation methods, aiming to extend a model's context length without requiring retraining. These methods mainly include modifications to relative positional encoding, functional reparameterization of position embeddings, remapping mechanisms based on interpolation or mixed interpolation strategies, and several coordinate transformation-based RoPE extension techniques (e.g., RoPE extrapolation).

In this project, we focus on three representative open-source language models—GPT-2, Qwen2.5-0.5B, and LLaMA3.2-1B—to systematically analyze the impact of positional encoding on model generalization in downstream tasks. We implement several mainstream positional encoding extension methods and evaluate their performance on long-text benchmark tasks, such as synthetic long-document understanding and long dialogue modeling. Our goal is to explore the differences in performance and applicability of various methods in practical scenarios.

For GPT-2, we applied position shift augmentation and fine-tuned a Shape-GPT variant on the WikiText-2-raw-v1 dataset. We evaluated its perplexity under varying shift values to assess inductive bias under positional perturbation. In parallel, we explored context window extension techniques for models with rotary embeddings. Specifically, for LLaMA 3 1.2B and Qwen2.5 0.5B, we implemented position interpolation and extended the NTK-aware RoPE method. Both models were fine-tuned on the RedPajama dataset using a 32k token context window, and evaluated on sequences of length from 2k to 32k to analyze perplexity degradation under extended contexts.

3 Background

In the Transformer architecture, due to the absence of recurrent and convolutional structures, the model itself lacks the inherent ability to process positional information within a sequence. Therefore, designing an effective positional encoding mechanism is one of the key factors contributing to the success of Transformers, as it enables the model to understand the order of tokens in a sequence.

This chapter introduces two mainstream positional encoding methods: Absolute Positional Encoding, as used in GPT-2, and Rotary Positional Encoding (RoPE), which has been widely adopted in recent high-performance language models.

3.1 Absolute Positional Encoding of GPT-2

The Transformer [6] uses sinusoidal positional encoding. Experimental results in the original paper show that sinusoidal and learnable positional encodings yield no significant difference in performance.

GPT-2 does not directly follow the sinusoidal positional encoding used in the original Transformer. Instead, it adopts an earlier approach known as learnable absolute positional embedding, where each position is assigned an independent vector that is updated during training alongside the token embeddings. As part of the model parameters, these embeddings are learned during pretraining, which effectively resolves most positional alignment issues.

More specifically, assuming the model's maximum context length is L and the embedding dimension is d, GPT-2 initializes a positional embedding matrix $P \in \mathbb{R}^{L \times d}$. During each forward pass, the token embeddings $X \in \mathbb{R}^{L \times d}$ are added element-wise to the corresponding positional embeddings:

$$X_{input} = X + P \tag{1}$$

This approach is simple and intuitive but comes with certain limitations:

Fixed positional dependency: The model only learns positional information up to a fixed length during pretraining, making it difficult to generalize to longer contexts.

No extrapolation capability: Positions beyond the maximum training length have no corresponding embedding vectors during inference. As evident in the code, the original design offers little room for extrapolation($max_positoin_embeddings$), since the parameter max_position_embeddings fixes the dimensionality.

As a result, GPT-2 performs poorly when handling extremely long texts, which has motivated researchers to explore positional encoding methods with better extrapolation capabilities.

3.2 Rotary Positional Encoding

Rotary Positional Encoding (RoPE) was first introduced in RoFormer [7] and has since been widely adopted in high-performance language models such as LLaMA and Qwen. The core idea of RoPE is to introduce relative positional information into self-attention computation by applying position-dependent rotational transformations to the query and key vectors in multi-dimensional subspaces.

Unlike traditional absolute positional encoding, RoPE does not explicitly add position vectors to token representations. Instead, it applies an angular rotation to the query and key vectors at each position before computing attention scores.

3.3 Mathematical Definition

Let the input vector $x \in \mathbb{R}^d$, with its i-th pair of dimensions as (x_{2i}, x_{2i+1}). RoPE introduces a position-dependent 2D rotational operation on each adjacent pair of dimensions. For the i-th dimension pair, the rotation angle is defined as:

$$\theta_i(p) = \frac{p}{10000^{\frac{2i}{d}}}$$

where p denotes the position of the current token, and d is the embedding dimension.

The rotation operation can be represented as applying a 2D rotation matrix to the pair (x_{2i}, x_{2i+1}):

$$\begin{bmatrix} x'_{2i} \\ x'_{2i+1} \end{bmatrix} = \begin{bmatrix} \cos\theta_i(p) & -\sin\theta_i(p) \\ \sin\theta_i(p) & \cos\theta_i(p) \end{bmatrix} \begin{bmatrix} x_{2i} \\ x_{2i+1} \end{bmatrix}$$

that is:

$$x'_{2i} = x_{2i}\cos\theta_i(p) - x_{2i+1}\sin\theta_i(p)$$
$$x'_{2i+1} = x_{2i}\sin\theta_i(p) + x_{2i+1}\cos\theta_i(p)$$

This operation can be equivalently viewed as treating the input vector as a complex number $z_i = x_{2i} + jx_{2i+1}$, which is then rotated by multiplying with the unit-modulus complex number $e^{j\theta_i(p)}$.

Extended to higher dimensions, the formula for RoPE is:

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^a_{\Theta,m}\boldsymbol{W}_{\{q,k\}}\boldsymbol{x}_m$$

in which

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

RoPE can capture relative positional information. In the two-dimensional case:

$$R_t = \begin{pmatrix} \cos\alpha_t & -\sin\alpha_t \\ \sin\alpha_t & \cos\alpha_t \end{pmatrix}$$

Since RoPE only operates on Q and K, and in the transformer Q and K are multiplied as QK^T, it can be derived that the product of the rotation matrices at two positions encodes positional information that depends solely on the relative positions.

$$R_m^\top R_n = \begin{pmatrix} \cos(\alpha_n - \alpha_m) & -\sin(\alpha_n - \alpha_m) \\ \sin(\alpha_n - \alpha_m) & \cos(\alpha_n - \alpha_m) \end{pmatrix} = R_{n-m}$$

3.4 The role of RoPE in self-attention

Let the query matrix be Q and the key matrix be K. After applying RoPE, the resulting vectors are denoted as $\widetilde{Q}$ and $\widetilde{K}$, and the attention scores are computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{\widetilde{Q} \cdot \widetilde{K}^T}{\sqrt{d}}\right) V$$

Because RoPE's rotational transformation introduces angular differences between positions, the dot product of the rotated vectors effectively captures the **relative positional differences**. Therefore, the self-attention mechanism can implicitly utilize relative positional information, improving generalization ability and exhibiting more stable performance when handling contexts longer than those seen during training. Currently, RoPE is widely used in mainstream large language models such as LLaMA and Qwen.

3.5 Advantages and Challenges of RoPE

RoPE, as a positional encoding method, has the following advantages:

- **Strong relative position modeling**: Since the rotation introduces positional differences rather than fixed position vectors, the model can learn the relative positional relationships between tokens.
- **Extrapolation capability**: The angle $\theta_i(p)$ is defined as a function, which theoretically allows extension to contexts much longer than the training length.
- **Parameter efficiency**: RoPE is essentially a functional transformation and does not require additional learnable parameters.

However, RoPE also faces certain challenges when dealing with ultra-long sequences (e.g., on the order of 100,000 tokens):

- **Angle divergence issue**: When the position p is very large, $\theta_i(p)$ can become excessively large, causing rapid oscillations in the rotation angle that affect stability.
- **Low-frequency dimension failure**: Certain dimensions have low frequencies and are insensitive to large positions, which may lead to imbalanced information representation.

Therefore, subsequent research has proposed various improvements—such as Linear RoPE, NTK RoPE, YaRN, etc.—to enhance RoPE's long-text processing capability based on its original principles.

4 Extrapolation of Absolute Positional Encoding

4.1 Shape

The design of positional encoding in Transformer models is crucial for capturing sequence order. Classical learnable Absolute Position Embeddings (APE) assign a unique vector to each position but lack shift invariance, making it difficult for the model to generalize to unseen position distributions during training. In contrast, Relative Position Embeddings (RPE) model the relative distances between tokens within the attention mechanism, naturally providing shift invariance and demonstrating superior performance in many sequence-to-sequence tasks. However, RPE usually requires modifications to the attention structure and increases computational complexity.

To combine the simplicity of APE with the generalization ability of RPE, we introduced the SHAPE (Shifted Absolute Position Embedding) mechanism in the GPT-2 model. During training, a random offset $k \sim \mathcal{U}[0, K)$ is added to each position embedding to break the model's dependence on fixed positions and encourage learning of relative positional information. Let the original position index be i; in the SHAPE paper [5], the sinusoidal position encoding is replaced by:

$$PE(i+k, m) = \begin{cases} \sin\left(\frac{i+k}{10000^{\frac{2m}{D}}}\right), & \text{if } m \text{ is even} \\ \cos\left(\frac{i+k}{10000^{\frac{2m}{D}}}\right), & \text{if } m \text{ is odd} \end{cases}$$

Where D denotes the embedding dimension, and m denotes the dimension index. We only apply the position index perturbation during the training phase, while keeping $k = 0$ during inference to ensure inference stability and positional consistency.

Algorithm 1 Forward Pass with SHAPE Position Embedding

function FORWARD(input_ids, position_ids=None, attention_mask, ...)
  if training **and** pe_noise_size > 0 **then**
    if position_ids is None **then**
      $position\_ids \leftarrow$ `torch.arange`$(0,$ `input_ids.size`$(1))$
      $position\_ids \leftarrow$ `position_ids.unsqueeze`(0)`.expand_as`$(input\_ids)$
    end if
    $shift_k \leftarrow$ `torch.randint`$(0, pe\_noise\_size)$
    $position\_ids \leftarrow position\_ids + shift_k$
    return parent_forward(input_ids, position_ids, attention_mask, ...)
  else
    return parent_forward(input_ids, position_ids, attention_mask, ...)
  end if
end function

Algorithm 1 demonstrates our practical implementation of position encoding in the GPT2 forward function. By simply adding a random offset `shift_k` to the `position_ids`, the SHAPE strategy is injected. This implementation requires no modifications to the Transformer architecture and does not increase inference computational cost.

We fine-tuned the original GPT2 and SHAPE-GPT2 on the WikiText-2-raw-v1 dataset. The training parameters were set as follows: learning rate of 5×10^{-5}, 3 training epochs, batch size of 1, optimizer AdamW, and random offset upper bound $K = 500$. During training, we used the standard language modeling task setup and evaluated performance with perplexity (PPL).

To verify that SHAPE encoding improves the model's shift robustness, we artificially applied various degrees of shift perturbations on the test set and recorded the PPL for both models. As shown in Figure 1, across all shift ranges, SHAPE-GPT2 consistently achieves significantly lower PPL than the original GPT2, especially maintaining low perplexity at offset $k = 500$.

4.2 conclusion

In summary, our method demonstrates a minimalistic and plug-and-play position encoding improvement strategy, SHAPE, which requires only minor code changes to endow the model with stronger generalization and positional robustness, especially showing practical potential in long-context inference and transfer tasks.

5 Extrapolation of RoPE

By examining the RoPE formula, we find that its key components include the rotation angle θ and the base (default value 10000). Most mainstream extrapolation methods modify these two key parameters.

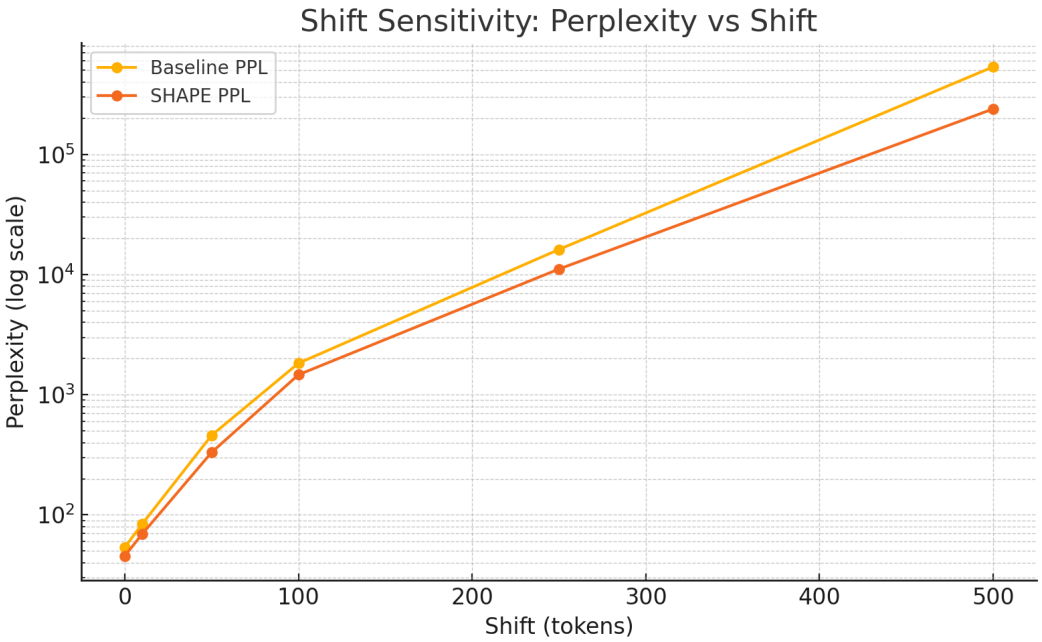

Figure 1: Comparison of PPL on the WikiText-2 test set for GPT2 and SHAPE-GPT2 under different shift conditions. SHAPE demonstrates stronger robustness to positional perturbations.

5.1 Linear Interpolation

Since the RoPE rotation angle mainly depends on the position id and the base value, we consider improvements from these two perspectives. Starting from the position id, interpolation methods are quite common, such as PI [2].

Linear interpolation is very straightforward. Suppose f' is the modified positional encoding function, f is the original positional encoding function, m is the position index, L is the input length, and L' is the length we want the model to extrapolate to. The linear interpolation formula is as follows:

$$\mathbf{f}'(\mathbf{x}, m) = \mathbf{f}\left(\mathbf{x}, \frac{mL}{L'}\right)$$

Assume the training dataset length is L: [1,2,3,4,5,6,...,L]; during inference, the length is [1,2,3,4,...,L,L+1,...,4L]. Since the position L+1 has never been seen during training, it leads to an out-of-distribution (OOD) scenario and poor performance. To address this, we can scale down all positions by a factor of 4, compressing them back into the trained range: [0.25,0.5,0.75,1,...,L/4,(L+1)/4,...,L]. The advantage of linear interpolation is that it avoids the issue of positions exceeding the training range, but its drawback is that it prevents the neural network from distinguishing the order and position of tokens that are very close, thus disturbing the model's local resolution. Generally, some fine-tuning is required for it to be effective.

We implemented linear interpolation (PI) on Qwen and LLaMA and trained on subsets of the RedPajama dataset. We filtered samples by different length segments and selected multiple gradients for training and testing.

In the original paper, the authors only performed minimal fine-tuning. In our experiment, we set steps to 1000/500. Based on the LLaMA 3.2 1B model, we fine-tuned on the RedPajama text corpus with the maximum context length set to 8192. We evaluated RoPE-PI with different scaling factors (scaling = 1.0, 0.85, 0.7) under multiple context lengths (2048, 4096, 8192, 16384, 32768) using perplexity as the metric. Lower perplexity indicates more accurate next-token prediction, thus intuitively reflecting differences in RoPE-PI's extrapolation capability.

From the results, as context length increases from 2k to 3k, the model's perplexity continuously decreases, indicating that longer history effectively improves prediction quality. Meanwhile, different scaling factors show varying effects across different lengths: scaling = 1.0 performs best in short to medium lengths (2048–8192); scaling = 0.85 is most stable at medium to long lengths (8192–16384), often slightly better than the original setting; scaling = 0.7, while performing worst at short context (2048), sees the fastest perplexity decline as context lengthens, even matching or slightly surpassing others at 32768.

5.1.1 Linear RoPE *Interpolation* in LLaMA

We implemented a linear scaling inference strategy in the LLaMA3.2-1B [3] model. This method compresses the position indices during inference to within one quarter of the training range. Its core idea can be expressed as:

$$\mathbf{f}'(\mathbf{x}, m) = \mathbf{f}\left(\mathbf{x}, \frac{mL}{L'}\right) \tag{2}$$

Here, m is the original position index, L is the maximum length during training, and L is the length during inference. Through this formula, we map position indices that exceed the training range back into the learnable interval. The advantage of this method is that it avoids the RoPE saturation problem caused by out-of-bound positions; however, it also has certain drawbacks. For example, the scaled continuous integer indices become floating-point numbers, which coarsens the model's perception granularity of local positions.

We conducted experiments on the RedPajama [8] text dataset. During preprocessing, we truncated the original corpus, retaining only samples with context lengths between 2048 and 32768 tokens, in order to control memory usage and cover typical long-text ranges.

Subsequently, we fine-tuned the RoPE interpolation model under a scaling factor of 0.25. The training configuration included: a maximum of 500 steps, learning rate (not specified in your text), batch size set to 1, and gradient accumulation over 16 steps to save memory. During training, we preserved the original text fields for re-tokenization, and launched training via the Huggingface Trainer API.

After training completion, we evaluated the model's language modeling capability on the validation set under different context length and scaling factor combinations. We tested context lengths of 2048, 4096, 8192, 16384, 32768 and scaling factors of 1.0, 0.85, 0.7, producing the heatmap shown in Figure 2.

From the results, we can draw the following conclusions:

- The model's perplexity continuously decreases as the context length increases from 2k to 32k, indicating that long-range historical information indeed helps improve language modeling performance.
- It performs best in the short to medium range (2048–8192).
- It remains stable in the medium to long range (8192–16384), often slightly better than the original setting.
- Although its performance is suboptimal for short contexts, the perplexity decreases the fastest as the context lengthens, achieving the global best (PPL = 8.48) at 32768 tokens.

This experiment demonstrates the practicality of the linear interpolation mechanism in addressing the RoPE extrapolation problem and also verifies that the scaling strategy can play an important role in long-context inference.

5.2 advanced NTK

5.2.1 background

Rotary Position Embedding (RoPE), with its excellent positional representation capability and relative position awareness mechanism, has become a standard component in large language models. However, as models are applied to increasingly longer sequences, the geometric progression frequency

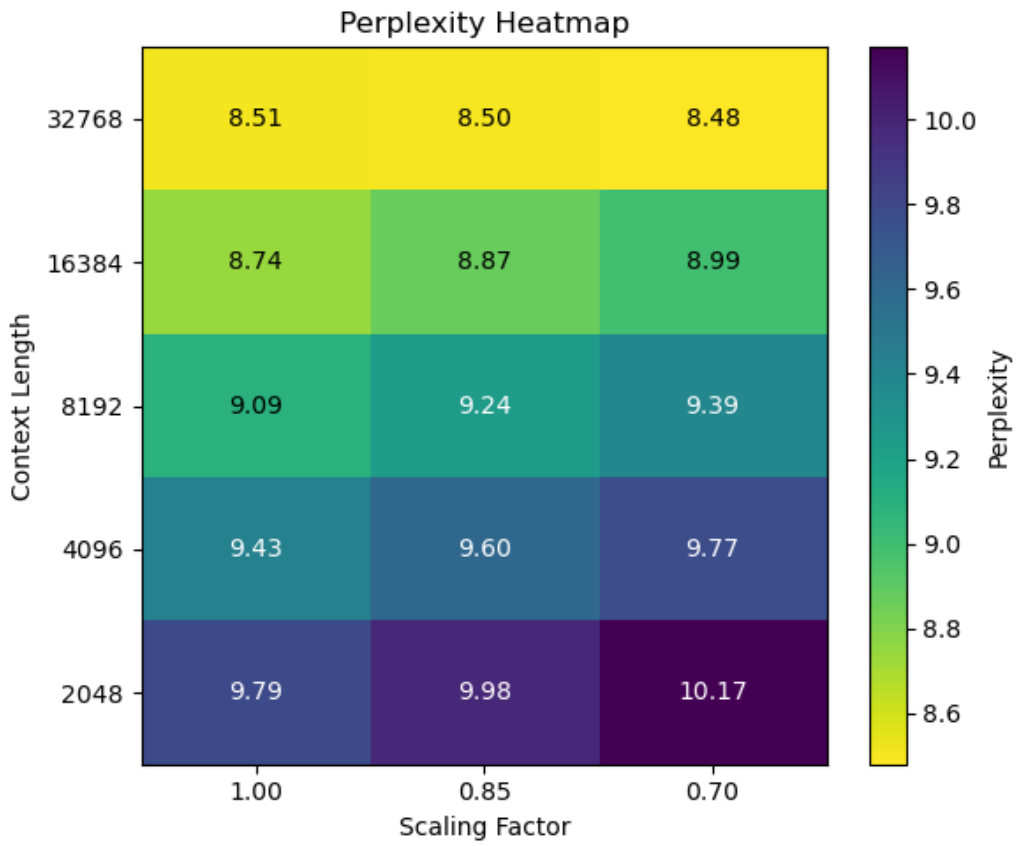

Figure 2: Heatmap of perplexity for the RoPE interpolation model under different context lengths and scaling factors. Lighter colors indicate lower PPL, which means the model's predictions are more accurate.

distribution in RoPE's original design limits its long-sequence modeling ability. Although methods like Position Interpolation (PI) provide RoPE with preliminary length extrapolation capability by simply "stretching" the position indices, such adjustments lack a solid theoretical foundation and exhibit clear performance ceilings in practice.

This motivates researchers to seek more fundamental improvements to RoPE from deeper mathematical theories. In particular, Neural Tangent Kernel (NTK) theory [4] offers a new perspective on position representation, providing theoretical support for addressing long-sequence modeling challenges.

Mathematically, the core of the NTK method lies in modifying RoPE's frequency distribution function by using an NTK-scaled version:

$$\theta_i = 10000^{-\frac{2i}{d} \cdot \alpha} \tag{3}$$

The frequency calculation formula that replaces the original RoPE is:

$$\theta_i = 10000^{-\frac{2i}{d}} \tag{4}$$

Here, α is a scaling factor, typically less than 1. This modification is based on the training dynamics theory of deep neural networks, particularly the frequency-dependent signal propagation characteristics within the network.

5.2.2 Attempts and Findings on Improving NTK

Whether scaling the position indices or modifying the base, all tokens become closer to each other in the representation space, which impairs the LLM's ability to distinguish the order of closely

positioned tokens. Considering that the model has already seen the full range of high-frequency components during training, we hypothesize that interpolating only the low-frequency components may be more effective. The NTK-by-parts method proposed by YaRN [6] suggests dynamically adjusting the interpolation strategy based on the context length L.

We implemented a minimal piecewise method that sets a cut-off threshold, applying the original positional encoding in the frequency domain above this threshold, while applying the NTK method below it. Let r be the cut-off ratio (default 0.9), and define $i_{\text{cut}} = \left\lfloor r\frac{d}{2} \right\rfloor$, then:

$$\omega_i^{\text{NTK-C}} = \begin{cases} \left(\theta K^{\frac{d}{d-2}}\right)^{-\frac{2i}{d}}, & i < i_{\text{cut}} \\ \theta^{-\frac{2i}{d}}, & i \geq i_{\text{cut}} \end{cases}$$

Additionally, introducing a temperature t on the compatibility scores before the Softmax can continuously reduce perplexity; this is called attention scaling. This method is orthogonal to NTK-by-parts, and after applying attention scaling, further performance improvements can still be observed on long-sequence samples.

In NTK-C, when the sequence length $L > 8192$, we introduce a length-dependent temperature t for scaling:

$$t = \sqrt{\max\left(1, \frac{L}{8192}\right)}$$

And scale Q before computation (equivalent to dividing everything by t):

$$\text{Attn}_{\text{CLS}}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d} \cdot t}\right) V$$

5.2.3 Experiments and Analysis

We conducted tests on the Qwen2-0.5B-Instruct [9] model, using the Qwen implementation based on the Transformers library, applying both NTK and NTK-C methods, and evaluated on the Leval benchmark [1]. To ensure reliability and comparability, we focused solely on close-ended tasks.

The experimental results are shown in Table 1. It can be observed that different positional encoding methods perform variably across sequences of different lengths. For shorter sequences (e.g., ToEFL, approximately 4K tokens), the NTK with $k = 2$ method achieved the best score of 52.41%. For medium-length sequences (e.g., QuALITY, about 7K tokens), the NTK-C method demonstrated advantages, with both cut-off settings reaching 33.66

Notably, on the long-sequence task Topic Retrieval (around 12.5K tokens), the NTK-C method significantly improved over the original RoPE implementation, increasing from 12% to 25.33%, confirming the effectiveness of segmented frequency adjustment in long-sequence scenarios. For the ultra-long sequence task CodeU (around 31.5K tokens), all methods performed relatively weakly, but NTK with $k = 2$ still achieved the best relative result of 4.4

Interestingly, for the Coursera task (about 9K tokens), the original RoPE showed the best performance (34.73%), possibly related to the specific knowledge retrieval nature of this task. On the GSM dataset, the various methods performed similarly, except that NTK-C with $k = 2$ and cut=0.8 was slightly weaker.

Overall, the cut-off value of 0.8 demonstrated a good performance balance across multiple tasks, validating the effectiveness of our proposed segmented frequency adjustment strategy.

We further conducted ablation experiments on the attention scaling technique, with results shown in Table 2. Under the same NTK-C configuration of k=2 cut=0.9, removing attention scaling led to varying degrees of performance changes across multiple tasks. For shorter sequence tasks (ToEFL, GSM, QuALITY), the impact of attention scaling was negligible. However, for longer sequence tasks such as Coursera and Topic Retrieval, removing attention scaling caused performance drops of 0.73 and 0.67 percentage points respectively, demonstrating the positive effect of attention scaling

Table 1: Results on Long-Context Understanding Benchmark

dataset	Domain	len	Position Encoding Optimization Methods			
			RoPE	**NTK k=2**	**NTK-C k=2, cut=0.9**	**NTK-C k=2, cut=0.8**
ToEFL	English Test	3907	49.81	**52.41**	52.04	51.67
GSM	Math	5557	**35**	**35**	**35**	30
QuALITY	MCQ	7169	32.17	33.16	**33.66**	**33.66**
Coursera	Advanced Course	9075	**34.73**	28.34	28.92	29.79
Topic Retrieval	Retrieving	12506	12.0	23.33	**25.33**	**25.33**
CodeU	Predict Output	31575	2.2	**4.4**	1.1	1.1

NTK-C: NTK positional encoding with frequency cutoff;k: scaling factor;cut: frequency cutoff threshold.

on long-sequence modeling. This also confirms that the cut-off strategy contributes benefits beyond those of attention scaling alone.

Table 2: Ablation Study on Attention Scaling (NTK-C k=2 cut=0.9)

dataset	With Attention Scaling	Without Attention Scaling	difference
ToEFL	52.04	52.04	0
GSM	30	30	0
QuALITY	33.66	33.66	0
Coursera	29.79	29.06	**-0.73**
Topic Retrieval	25.33	24.66	**-0.67**
CodeU	1.11	3.33	**+2.22**

In summary, our experiments demonstrate the effectiveness of the NTK-C method in long-sequence modeling, particularly excelling in the medium to long sequence range. When combined with attention scaling, it further enhances performance across most scenarios. These findings provide a viable optimization approach to improve the long-context handling capabilities of large language models.