LongRoPE: Extending LLM Context Window Beyond 2 Million Tokens

Yiran Ding [1 2] **Li Lyna Zhang** [1 †] **Chengruidong Zhang** [1] **Yuanyuan Xu** [1 3] **Ning Shang** [1] **Jiahang Xu** [1]
Fan Yang [1] **Mao Yang** [1]

Abstract

Large context window is a desirable feature in large language models (LLMs). However, due to high fine-tuning costs, scarcity of long texts, and catastrophic values introduced by new token positions, current extended context windows are limited to around 128k tokens. This paper introduces LongRoPE that, for the first time, extends the context window of pre-trained LLMs to an impressive **2048k** tokens, with up to only 1k fine-tuning steps at within 256k training lengths, while maintaining performance at the original short context window. This is achieved by three key innovations: (i) we identify and exploit two forms of non-uniformities in positional interpolation through an efficient search, providing a better initialization for fine-tuning and enabling an 8× extension in non-fine-tuning scenarios; (ii) we introduce a progressive extension strategy that first fine-tunes a 256k length LLM and then conducts a second positional interpolation on the fine-tuned extended LLM to achieve a 2048k context window; (iii) we readjust LongRoPE on 8k length to recover the short context window performance. Extensive experiments on LLaMA2 and Mistral across various tasks demonstrate the effectiveness of our method. Models extended via LongRoPE retain the original architecture with minor modifications to the positional embedding, and can reuse most pre-existing optimizations. Code is available at https://github.com/microsoft/LongRoPE

[1]Microsoft Research [2]Hangzhou Dianzi University [3]University of Science and Technology of China; Yiran Ding and Yuanyuan Xu did this work during the internship at MSRA. Correspondence to: Li Lyna Zhang <lzhani@microsoft.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Large Language Models (LLMs), despite remarkable success on various tasks (OpenAI et al., 2023; Touvron et al., 2023), often suffer from limited context window size, e.g., LLaMA2's 4096 token limit (Touvron et al., 2023). Beyond the context window, LLM's performance declines due to the additional positions that the model has not been trained on. This poses challenges in important scenarios like in-context learning with numerous examples (Huang et al., 2023) and LLM agents (Park et al., 2023; Madaan et al., 2023).

Recent works show that a pre-trained LLM context window can be extended to around 128k by fine-tuning on longer texts (Chen et al., 2023b;a; Peng et al., 2023; Zhang et al., 2024; Liu et al., 2023). There are three major obstacles to further extend the context window. First, untrained new position indices introduce many catastrophic values, leading to out-of-distribution issues and making fine-tuning difficult to converge (Chen et al., 2023a). This is particularly challenging when an extension from 4k to >1000k introduces more than 90% new positions. Second, fine-tuning usually requires texts of corresponding lengths. However, long texts in current datasets, especially those exceeding 1000k, are limited. Moreover, training on extra-long texts is computationally expensive, requiring prohibitively extensive training hours and GPU resources. Third, when extending to extremely long context windows, the attention becomes dispersed as it's spread thinly across numerous token positions, degrading performance on the original short context (Chen et al., 2023a).

One approach to mitigate the first challenge is to interpolate RoPE positional embedding (Su et al., 2021; Chen et al., 2023a), which downscales new position indices to the pre-trained range, as shown in Fig.1. Position Interpolation (PI) (Chen et al., 2023a) linearly interpolates RoPE's rotary angles by the extension ratio. NTK (LocalLLaMA, 2023b;a) advocates unequal interpolation and extrapolation across RoPE dimensions. YaRN (Peng et al., 2023) categorizes RoPE dimensions into three frequency-based groups and applies extrapolation, NTK, and linear interpolations, respectively. However, positional embedding exhibits *complex non-uniform information entropy* in the Transformer architecture. Such subtle non-uniformity is not effectively

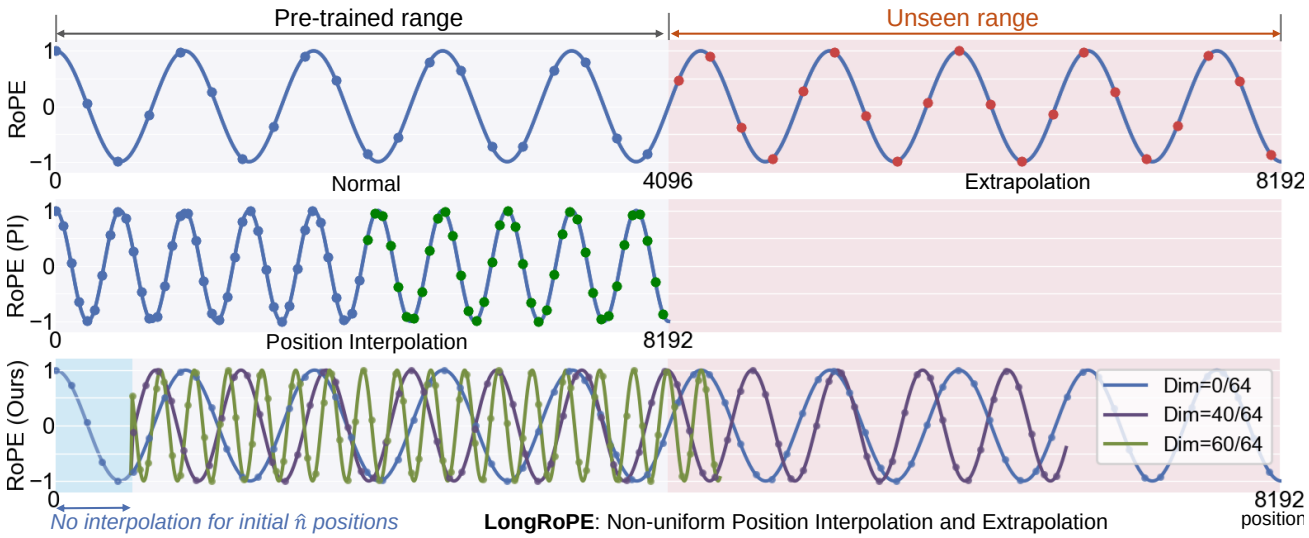

Figure 1. An illustrative example to show RoPE embedding under different interpolation methods. *Upper*: RoPE under direct extrapolation. *Middle*: Rescaled RoPE under linear positional interpolation. *Down*: LongRoPE fully exploits the identified two non-uniformities, leading to varied interpolation and extrapolation across RoPE dimensions at different token positions.

leveraged by existing approaches, leading to information loss and hence limiting the context window size.

Section 2 reveals two key findings empirically: **(1)** Effective positional interpolation should consider two forms of non-uniformities: varying RoPE dimensions and token positions. Lower RoPE dimensions and initial starting token positions benefit from less interpolation, but the optimal solutions depend on the target extended length. **(2)** By considering these non-uniformities into positional interpolation, we can effectively retain information in the original RoPE, particularly key dimensions and token positions. This minimizes the loss caused by positional interpolation, and thus provides better initialization for fine-tuning. Moreover, it allows an $8\times$ extension in non-fine-tuning scenarios.

Motivated by the findings, we introduce **LongRoPE**, an effective method that extends the LLM context window beyond 2 *million* tokens. LongRoPE is based on three key innovations. First, LongRoPE fully exploits multidimensional non-uniformities in positional interpolation. It identifies effective rescale factors for RoPE's rotation angles for each RoPE dimension, based on token positions. As the search space that identifies rescale factors expands exponentially with the target extension ratio, LongRoPE introduces an evolutionary search algorithm with two optimization techniques to boost search efficiency. Fig. 1 shows an example of the searched rescaled RoPE.

Then, LongRoPE leverages an efficient, progressive extension strategy to achieve a 2048k context window without the need of direct fine-tuning on texts with extremely long lengths, which are scarce and hardly available. The strategy begins by searching for a 256k length on the pre-trained LLM and fine-tuning it under this length. Then, as our non-uniform positional interpolation allows for an $8\times$ extension in non-fine-tuning settings, we conduct a second search for new RoPE rescale factors on the fine-tuned extended LLM. This ultimately achieves the 2048k context window for LLaMA2 and Mistral (Jiang et al., 2023).

Finally, to mitigate performance degradation on the original (shorter) context window, LongRoPE continues to adjust the RoPE rescale factors on the extended LLM. Similar to scaling up from 256k to 2048k, we scale down to 4k and 8k context windows on the 256k fine-tuned LLM using our search algorithm to encourage less positional interpolation. During inference, if the sequence length is less than 8k, we update RoPE with the searched rescale factors.

Extensive experiments across different LLMs and various long-context tasks demonstrate the effectiveness of our method. We show that LongRoPE is highly effective in maintaining low perplexity from 4k to 2048k evaluation length, achieving over 90% passkey retrieval accuracy, and delivering comparable accuracy on standard benchmarks designed within the 4096 context window. LongRoPE can be applied to any LLMs based on RoPE embedding.

2. Non-uniformity in Positional Interpolation

2.1. Preliminary

Transformer models require explicit positional information, often in the form of position embedding, to represent the order of input tokens. Our work focuses on the RoPE (Su et al., 2021) position embedding, which is widely used in recent LLMs. For a token at position index n, its corresponding RoPE encoding can be simplified as follows:

$$[cos(n\theta_0), sin(n\theta_0), cos(n\theta_1), \cdots, cos(n\theta_{d/2-1}), sin(n\theta_{d/2-1})] \quad (1)$$

where d is the embedding dimension, $n\theta_i$ is the rotary angle of token at position n, $\theta_i = \theta^{-2i/d}$ represents the rotation frequencies. In RoPE, the default base value of θ is 10000.

Context window extension ratio s **and positional interpolation**. We define s as the ratio of extended context length L' to the original length L: $s = \frac{L'}{L}$.

To extend the context window from L to L', current positional interpolation methods suggest downscaling rotation frequencies θ_i based on extension ratio s. Let $\beta = \theta^{2/d}$, and λ denote the actual rescale factor related to s, we unify these positional interpolation methods as follows:

$$\left[cos\left(\frac{n}{\lambda(\beta)^0}\right), sin\left(\frac{n}{\lambda(\beta)^0}\right), cos\left(\frac{n}{\lambda(\beta)^1}\right), \cdots, sin\left(\frac{n}{\lambda(\beta)^{d/2-1}}\right)\right] \tag{2}$$

Linear positional interpolation (PI). PI (Chen et al., 2023a) suggests linear interpolation of position indices within the pre-trained length limit. For a target extension ratio s, the rotation angles of all positions are linearly reduced by $\lambda = s$ across all RoPE dimensions. However, this makes the position information very "crowded", hindering the model's ability distinguish closely positioned tokens. Therefore, PI tends to underperform at high extension ratios.

NTK-based interpolation and extrapolation. LocalLLaMA (2023b;a) look at RoPE from an information encoding perspective and apply the Neural Tangent Kernel (NTK) theory (Jacot et al., 2018; Tancik et al., 2020). To mitigate the crowded-positions issue in PI, they suggest to distribute interpolation pressure across RoPE dimensions. It scales lower (high frequency) dimensions less and higher (low frequency) dimensions more, resulting in both positional interpolation and extrapolation, where $\lambda = s^i$. The improved dynamic NTK (LocalLLaMA, 2023a) adjusts the extension ratio at each position based on the current sequence length. Unlike PI, which necessitates fine-tuning, NTK-aware methods can extend context windows in non-fine-tuning scenarios, but usually with a maximum extension ratio of 4×.

YaRN (Peng et al., 2023) introduces a significant improvement to positional interpolation performance. It divides RoPE dimensions into three frequency-based groups, each with a different interpolation strategy. High frequency dimensions undergo extrapolation (λ=1), while low frequency dimensions use linear interpolation (PI). The RoPE dimensions that fall in-between employs the NTK. The key of YaRN lies in its grouping of RoPE dimensions, which currently depends on human-led empirical experiments. This may result in sub-optimal performance for new LLMs.

2.2. Study on Non-uniform Positional Interpolation

Inspired by NTK and YaRN, we notice their gains from non-linearity, specifically in considering different frequencies across RoPE dimensions for specialized interpolation and extrapolation. However, the non-linearities currently implemented in NTK and YaRN are heavily dependent on rules designed by humans. This naturally raises two questions: (1) Is the current positional interpolation optimal? (2) Are there unexplored non-linearities?

Table 1. Perplexity of LLaMA2-7B extended via different methods. By a simple search for the rescale factors of each RoPE dimension, we can greatly reduce the perplexity.

(LLaMA2-7B) Extension method	Context Window Size			
	PG19 (5 samples)		Proof-pile (10 samples)	
	8192	16384	8192	16384
PI	10.65	20.49	3.65	4.93
Dy-NTK	10.21	23.29	3.50	3.87
YaRN	32.64	87.89	3.49	3.25
Search for RoPE Dim-wise λ	**9.37**	**11.34**	**3.45**	**3.13**

Table 2. Perplexity of LLaMA2-7B extended on PG19 test set. When retaining the first $\hat{n}$ tokens without positional interpolation, the performance of both PI and Dynamic-NTK are improved.

(LLaMA2-7B) Extension method	L'	No interpolation for first $\hat{n}$ tokens								
		0	2	4	8	16	32	64	128	256
PI	8k	8.12	8.12	8.13	8.11	8.11	8.06	7.99	**7.97**	8.12
	16k	14.15	14.11	14.07	13.94	13.89	**13.85**	13.97	14.33	23.96
Dy-NTK	8k	7.65	7.65	7.66	7.66	7.66	7.64	7.61	**7.60**	7.75
	16k	14.69	14.68	14.67	14.57	**14.37**	14.44	14.69	16.44	43.87

To answer these questions, we use evolution search (see Sec. 3) to discover better non-uniform positional interpolations for LLaMA2-7B. The search is guided by perplexity, using 5 random samples from PG19 (Rae et al., 2019) validation set. Through our empirical analysis, we reveal the following key findings.

Finding 1: *RoPE dimensions exhibit substantial non-uniformities, which are not effectively handled by current positional interpolation methods.*

We search the optimal λ for each RoPE dimension in Eq. 2. Table 1 compares the perplexity of LLaMA2-7B under different methods on PG19 and Proof-pile (Azerbayev et al., 2022) test sets, without fine-tuning. Our searched solution shows significant improvements, suggesting that current linear (PI) and non-uniform (Dynamic-NTK and YaRN) interpolations are sub-optimal. Notably, YaRN underperforms than PI and NTK on PG19, as it doesn't reach the target context window length for non-fine-tuned LLM. For example, YaRN's perplexity spikes after 7k in an 8k context size.

Through our search, the rescaled factors λ in Eq. 2 become non-uniform, differing from the fixed scale s in PI, NTK's formula calculation, and YaRN's group-wise calculation. These non-uniform factors significantly improve LLaMA2's language modeling performance (i.e., perplexity) for 8k and 16k context windows without fine-tuning. This is because the resulting positional embedding effectively preserves the original RoPE, especially key dimensions, thus reducing LLM's difficulty in distinguishing close token positions.

Finding 2: *RoPE for the initial tokens in the input sequence should be extrapolated with less interpolation.*

Table 3. Proof-pile perplexity of the extended LLaMA2-7B with a 64k context window in non-fine-tuned and fine-tuned settings.

Method	non-fine-tuned	fine-tuned
PI	72.54	2.44
YaRN	4.15	2.42
Search (Dim-wise λ and $\hat{n}$)	**3.22**	**2.36**

For the initial sequence of tokens, represented by $\hat{n}$, we hypothesize that their RoPE should do less interpolation. This is because they receive large attention scores, making them crucial to attention layers, as observed in Streaming LLM (Xiao et al., 2023) and LM-Infinite (Han et al., 2023). To verify this, we extend the context window to 8k and 16k using PI and NTK, keeping the first $\hat{n}$ (0,2, ..., 256) tokens without interpolation. When $\hat{n}$=0, it reverts to the original PI and NTK. Table 2 highlights two key observations: **(1)** retaining the starting tokens without position interpolation indeed improves the performance. **(2)** The optimal number of starting tokens, $\hat{n}$, depends on the target extension length.

Finding 3: *Non-uniform positional interpolation effectively extends LLM context window in both fine-tuning and non-fine-tuning settings.*

While we've shown that our searched non-uniform position interpolation significantly improves the extension performance at 8k and 16k without fine-tuning, longer extensions require fine-tuning. As such, we fine-tune LLaMA2-7B with our searched RoPE for a 64k context window size (see Appendix for settings). As Table 3 shows, our method significantly outperforms PI and YaRN, both before and after fine-tuning LLaMA2-7B. This is due to our effective use of non-uniform positional interpolation, minimizing information loss and providing a better initialization for fine-tuning.

Summary. Our study uncovers two non-uniformities: varying RoPE dimensions and token positions. Utilizing these non-uniformities effectively in positional interpolation greatly improves LLM context extension performance.

3. LongRoPE

Motivated by the findings, we present LongRoPE, which first introduces an efficient search algorithm to fully exploit the two non-uniformities, and then uses it to extend LLM context window beyond 2 million tokens.

3.1. Problem Formulation

The two non-uniformities can lead to a vast solution space and introduce complexities in optimization. To address it, we frame the multidimensional non-uniform position interpolation optimization problem as a search problem.

For an LLM targeting a context window size of L' and lengthy input documents $\mathbf{X}$, where each $\mathbf{x} \in \mathbf{X}$ surpasses L' in token length, we denote the original rotary angle of the i^{th} dimension in RoPE embedding at token position n as $\frac{n}{\beta^i}$.

Table 4. Search space for RoPE rescale factors. Tuples of three values represent the lowest value, highest, and step size.

Non-uniformity	Notation	Search Space
RoPE dimension	λ_i	(1.0, extension ratio s×1.25, 0.01)
Starting tokens	$\hat{n}$	{0, 1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 64, 128, 256}

The optimization problem is then formulated as follows:

$$\underset{\mathbf{x}\in\mathbf{X};\,|\mathbf{x}|\geq L'}{\arg\min}\ \mathcal{L}\left(\text{LLM}(\text{RoPE},\mathbf{X})\right), \text{where}$$

$$\underset{\substack{i=0,\cdots,\frac{d}{2}-1;\\ n\in[0,|\mathbf{x}|);}}{\text{RoPE}(n)} = \left[\cdot\cdot, cos\left(\mathbb{I}(\hat{\lambda}_i,\hat{n})\times\frac{n}{\beta^i}\right), sin\left(\mathbb{I}(\hat{\lambda}_i,\hat{n})\times\frac{n}{\beta^i}\right), \cdot\cdot\right]$$

$$\text{where}\ \ \mathbb{I}(\hat{\lambda}_i,\hat{n}) = \begin{cases} 1 & n<\hat{n} \\ \frac{1}{\hat{\lambda}_i} & n\geq\hat{n} \end{cases} \tag{3}$$

where we introduce a set of rescale factors, $\mathbb{I}(\hat{\lambda}_i,\hat{n})$, to cover the two forms of non-uniformities. $\hat{\lambda}_i$ and $\hat{n}$ denote the non-uniformity of RoPE dimensions and token positions, respectively. Specifically, we use $\mathbb{I}(\hat{\lambda}_i,\hat{n})$ to rescale the rotation angle for the i^{th} RoPE dimension, where $\hat{\lambda}_i$ is the rescale factor and $\hat{n}$ is token position threshold. For initial $\hat{n}$-1 token positions, the rescale factor $\hat{\lambda}_i$ will not take effect, and the original RoPE rotary angle $\frac{n}{\beta^i}$ is used. For tokens at positions $n \geq \hat{n}$, the rescale factor is applied.

Given a target context window size of L', our objective is to find the optimal rescale factors ($\mathbb{I}(\hat{\lambda}_0,\hat{n})$, $\mathbb{I}(\hat{\lambda}_1,\hat{n})$,...$\mathbb{I}(\hat{\lambda}_i,\hat{n})$...) from the 1^{st} to d^{th} RoPE dimension. As a result, the target LLM, with the rescaled RoPE, can achieve a minimum next token prediction loss, $\mathcal{L}$ (i.e., the perplexity), for input samples $\mathbf{X}$ with a token length of L'.

3.2. Searching the Non-uniform Position Interpolation

To solve the problem in Eq. 3, we now introduce our simple yet highly effective method, which searches for the optimal RoPE rescale factors to fully exploit the multidimensional non-uniformities in position embedding.

Search space. We design a large search space to include the two non-uniformies. Table 4 illustrates the search space. Specifically, we allow the search of a specialized rescale factor for each dimension in RoPE embedding. To simply search space design, we search λ_i and $\hat{n}$ instead of searching for $\mathbb{I}(\hat{\lambda}_i,\hat{n})$, where $\hat{\lambda}_i = 1/\lambda_i$. As shown in Table 4, λ_i is allowed to search from a minimum value of 1.0 (i.e., direct extrapolation) to a maximum value of $s \times 1.25$ (i.e., larger interpolation than PI) with a step size of 0.01, where s is the target context window extension ratio.

$\hat{n}$ controls the number of initial token positions that are retained without position interpolation (i.e., use the original RoPE embedding). Empirically, we allow $\hat{n}$ to search from $\{0, 1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 64, 128, 256\}$. When $\hat{n} = 0$, all token positions use the searched rescale factors.

Algorithm 1 The search algorithm for effective non-uniform positional interpolation

Input: target LLM, input samples $\mathbf{X}$, population size P, mutation size N_1, crossover size N_2, max iterations $\mathcal{T}$, mutate probability p

```
Top-k=φ;
P_0=Initialize_population_with_optimization (P, X, p);
for i=1 to T do
    Compute_perplexity (LLM, P_{i-1}, X);
    Top-k = Update_Topk (Top-k, P_{i-1});
    P_mutation=Mutation_with_mono_constraint (Top-k, N_1, p);
    P_crossover=Crossover_with_mono_constraint (Top-k, N_2);
    P_i = P_mutation ∪ P_crossover ∪ Top-k;
end for
Return the individual with lowest perplexity in Top-k;
```

Evolution-based search. Our search space in Table 4 spans numerous positional interpolation solutions, posing a significant challenge for efficient exploration. For example, a $s = 4\times$ extension leads to $400^{128/2} \times 14 = 4 \times 10^{167}$ choices. With larger extension ratio, the search space expands exponentially. To address this, we use evolution search (Guo et al., 2020) and introduce two optimization techniques to greatly boost search efficiency. Algorithm 1 illustrates the overall search procedure.

Optimized initial population generation. Instead of initializing a population of P rescale factors randomly, we add the three RoPE rescale factors corresponding to PI, NTK, and YaRN as individuals into the initial population. For the remaining P-3 individuals, we randomly mutate the three rescale factors with a probability of p.

Monotonically non-decreasing constraint. After generating the initial population, we compute LLM perplexity for each individual. Specifically, we apply the corresponding RoPE rescale factors to the target LLM and compute the perplexity of input $\mathbf{X}$. The top-k individuals become parents for evolution. However, the vast search space can cause naive mutation and crossover to explore poor solutions, leading to unnecessary perplexity computations. This is particularly inefficient when L' is large, given the time-consuming inference of each perplexity calculation.

To address this, we impose a non-decreasing monotonicity constraint on the sampled RoPE rescaled factors: $\lambda_i \leq \lambda_{i+1}$. Only RoPE that satisfies this constraint is applied to LLM for perplexity evaluation, significantly reducing the search costs. Specifically, we require that λ_i increases monotonically with the RoPE dimension (i.e., i=0,...,63). This dimensional monotonicity is based on the NTK theory (Jacot et al., 2018; Tancik et al., 2020; LocalLLaMA, 2023b), suggesting that lower dimensions with higher frequency requires less interpolation (i.e., a smaller λ_i), and higher dimensions with lower frequency can do more interpolation (i.e., a larger λ_i).

8× extension without fine-tuning. Our evolutionary search effectively identifies non-uniform RoPE rescale factors, preserving key dimensions and positions to minimize interpolation-induced information loss. As depicted in Fig.2, our method is able to extend LLaMA2's context window from 4k to 32k without fine-tuning. In contrast, existing methods such as PI, and non-uniform NTK and YaRN cause perplexity to spike after 2× extension.

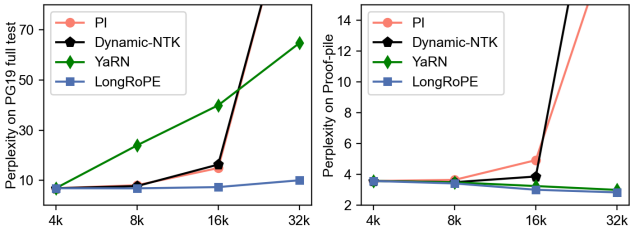

Figure 2. LLaMA2-7B perplexity on PG19 and Proof-Pile after extension using different methods, measured without fine-tuning. By fully exploiting the non-uniformities, LongRoPE achieves an **8× extension without fine-tuning**.

3.3. Extending LLM Context Window to 2048K

Progressive extension to 2048k. We now introduce our method to extend the context window of pre-trained LLMs from the traditional 4k to over 2048k. As demonstrated, our non-uniform positional interpolation can achieve 8× extension without fine-tuning. For larger extensions (i.e., 512×) is required, fine-tuning is necessary. One method is to search for RoPE rescaled factors under the target 2048k size and then fine-tune. However, this faces challenges due to the prohibitively expensive training resources. Moreover, public datasets contain a very limited number of long text documents (e.g., those exceeding 1000k tokens).

Fortunately, LongRoPE is effective for both the original and fine-tuned extended LLM. Therefore, we introduce an efficient, progressive method that achieves the target 2048k with just 1k fine-tuning steps at within 256k training length.

♢ *Extending pre-trained LLM to 256k with LongRoPE search*. Taking LLaMA2 as an example, we conduct search for target context window size of 128k and 256k. The extension ratio at this stage is 32× and 64×, respectively.

♢ *Fine-tuning to 256k*. Then, we fine-tune the pre-trained LLM to achieve the context window size of 256k. Specifically, we first fine-tune LLaMA2 for 400 steps using the RoPE rescaled factors for 128k. Then, we replace the RoPE rescaled factors to 256k on the finished checkpoint and conduct an additional 600 steps of fine-tuning. This method proves more efficient than directly fine-tuning to 256k.

♢ *Extending fine-tuned extended LLM to 2048k with LongRoPE search*. Finally, we perform a secondary LongRoPE search on the fine-tuned 256k-length LLM. This ultimately results in an extremely large context window size without further fine-tuning. Due to resource constraints, we extend the 256k context window to 2048k, achieving a total extension ratio of 512×.

Shorter context window recovery. After extending to an extremely long 2048k context window, we notice a performance drop within the original context window. This is a known issue of positional interpolation (Chen et al., 2023a), as it forces position embedding in higher dimensions within the original context window to reside in a much narrower region, negatively affecting the language model's performance. With a 512× extension ratio, positions within the original 4k context window become particularly crowded.

To mitigate this, we perform an extra evolution search on the extended LLM to adjust RoPE rescale factors for short context lengths (e.g., 4k and 8k). We reduce the maximum allowed searched λ due to less positional interpolation required for shorter lengths. During inference, the LLM dynamically adjusts the corresponding RoPE rescale factors.

4. Experiments

4.1. Setup

Evaluation Tasks and models. We apply LongRoPE on LLaMA2-7B and Mistral-7B, and evaluate the performance on three aspects: (1) perplexity of extended-context LLMs on long documents; (2) Passkey retrieval task that measures a model's ability to retrieve a simple passkey from a sea of irrelevant text; and (3) Standard LLM benchmarks within a short 4096 context window size.

Fine-tuning. For LLaMA2, we use a learning rate of 2e-5 with linear decay and a global batch size of 32. We fine-tune for 400 steps on Redpajama (Computer, 2023) dataset, chunked into 128k segments bookended with the BOS and EOS tokens. Then, based on the finished checkpoint, we train an additional 600 steps to achieve 256k context window. The 128k context size is trained on 8 A100 GPUs with the distributed training system (Lin et al., 2023), while the 256k requires 16 A100 GPUs. In the case of Mistral, a constant learning rate of 1e-6 and a global batch size of 64 are used. For both 128k and 256k models, we follow the setting in YaRN (Peng et al., 2023), with 400 steps on the Together Computer's Long-Data Collections (mis, 2024) using 16k sequence length. We use 4 A100 GPUs for training. We reuse the data precision settings from the original Huggingface model checkpoints. Specifically, we use FP16 for LLaMA2 and BF16 for Mistral.

Search. For target window size within 256k, we use: P=64, N_1=N_2=16, p=0.3, $\mathcal{T}$=40, and select top-32 for mutation/crossover in each iteration. Perplexity is calculated using 5 random PG19 validation set samples, with a minimum length requirement of the target context length. For windows over 512k, we halve the population, mutation, and crossover sizes. Perplexity is measured on 3 random samples from Pile-Books3 (Gao et al., 2020) validation set.

Baselines. To reach 2048k, we fine-tuned models with 128k and 256k context windows. This yields LongRoPE-2048k (ft=128k) and LongRoPE-2048k (ft=256k) for LLaMA2 and Mistral, respectively. We compare the four models with state-of-the-art context window extension baselines, specifically open-sourced LLMs fine-tuned after positional interpolation using PI, NTK and YaRN. This includes Together-32k (Together, 2023), Code LLaMA (Rozière et al., 2023), LongLoRA-full-FT-100k (Chen et al., 2023b), YaRN-LLaMA and YaRN-Mistral (Peng et al., 2023).

4.2. Main Results

Long sequence language modeling within 256k. We begin by comparing with state-of-the-art extended LLMs within a 256k evaluation length. We use two datasets to demonstrate our generalizability: Proof-pile (Rae et al., 2019) and PG19 (Gao et al., 2020) test splits. We evaluate perplexity at various context lengths using sliding window of 256. For PG19, we use the whole test split of 100 documents. For Proof-pile, we follow YaRN (Peng et al., 2023) to randomly select 10 samples, each with at least 128k lengths.

Table 5 and Table 7 compare the perplexity of LLaMA2 and Mistral extended via different interpolation methods on Proof-pile and PG19, respectively. We highlight two key observations: **(1)** our extended models show an overall decreasing perplexity trend from 4k to 256k evaluation lengths, proving their abilities to leverage longer context. **(2)** Even with a context window 16× longer, a condition typically challenging for maintaining performance at shorter lengths, our LongRoPE-2048k models significantly outperform state-of-the-art baselines within 256k context length.

Long sequence language modeling beyond 2000k. To evaluate the effectiveness on extremely long documents, we use the Books3 (Gao et al., 2020) dataset. For evaluation efficiency, we randomly select 20 books, each exceeding 2048k in length, and use a sliding window of 256k.

As shown in Table 6, LongRoPE successfully extends LLaMA2-7B and Mistral-7B's context window to 2048k, while also achieving perplexity comparable or superior to baselines within shorter lengths of 8k-128k. We also observe notable performance differences between the 2048k LLaMA2 and Mistral. Mistral outperforms baselines at shorter lengths, but perplexity exceeds 7 beyond 1024k. LLaMA2 performance aligns with expectations: the perplexity decreases gratefully with longer contexts, with marginal increases at 1024k and 2048k. Moreover, on LLaMA2, LongRoPE-2048k performs better at a fine-tuning length of 256k over 128k, due to the smaller secondary extension ratio (i.e., 8× vs. 16×). In contrast, Mistral performs better at fine-tuning window size of 128k. The main reason is that for Mistral's 128k and 256k fine-tuning, we follow YaRN's setting to use a 16k training length, which affects Mistral's

Table 5. Proof-pile perplexity of models with various positional interpolation methods. ft: the context window size used in fine-tuning. Even with a context window 16× longer than current long-context models, our models also outperform them within 256k context length.

Base LLM	Model Name	Context Window	Extension Method	Evaluation Context Length 4096	8192	32768	65536	98304	131072	262144
LLaMA2-7B	LLaMA2-7B	4k	-	**3.58**	$>10^4$	$>10^4$	$>10^4$	$>10^4$	$>10^4$	$>10^4$
	Together	32k	PI	3.69	3.50	2.64	$>10^2$	$>10^3$	$>10^4$	$>10^4$
	LongLoRA	100k	PI	3.83	3.62	2.68	2.44	2.33	9.89	$>10^3$
	Code LLaMA	100k	NTK	3.95	3.71	2.74	2.55	2.54	2.71	49.33
	YaRN (s=16)	64k	YaRN	3.69	3.51	2.65	2.42	$>10^1$	$>10^1$	$>10^4$
	YaRN (s=32)	128k	YaRN	3.75	3.56	2.70	2.45	2.36	2.37	99.64
	LongRoPE-2048k (ft=128k)	**2048k**	LongRoPE	3.67	**3.49**	**2.60**	**2.36**	**2.27**	**2.26**	**1.88**
	LongRoPE-2048k (ft=256k)	**2048k**	LongRoPE	3.69	3.64	**2.63**	**2.38**	**2.28**	**2.26**	**1.87**
Mistral-7B	Mistral v0.1	8k	-	**3.09**	2.96	$>10^2$	$>10^3$	$>10^3$	$>10^3$	$>10^4$
	YaRN (s=8)	64k	YaRN	3.18	3.04	2.37	2.20	10.39	57.4	$>10^4$
	YaRN (s=16)	128k	YaRN	3.21	3.08	2.41	2.24	2.18	2.19	4.91
	LongRoPE-2048k (ft=128k)	**2048k**	LongRoPE	**3.09**	**2.95**	**2.31**	**2.12**	**2.06**	**2.06**	**1.77**
	LongRoPE-2048k (ft=256k)	**2048k**	LongRoPE	3.10	**2.96**	**2.30**	**2.12**	**2.06**	**2.06**	**1.77**

Table 6. Perplexity evaluation on Books3 dataset. Without additional fine-tuning, our LongRoPE-2048k models, with a training context window size of 128k and 256k, effectively scale to an extremely long context size of 2048k. 1k=1024 tokens.

Base LLM	Model Name	Context Window	Extension Method	Evaluation Context Length 8k	16k	32k	64k	128k	256k	512k	1024k	2048k
LLaMA2-7B	LongLoRA	100k	PI	6.99	6.80	6.66	6.59	20.57	246.45	$>10^3$	$>10^4$	$>10^4$
	Code LLaMA	100k	NTK	7.68	7.49	7.38	7.88	9.80	98.30	$>10^3$	$>10^4$	$>10^4$
	YaRN (s=16)	64k	YaRN	**6.33**	**6.20**	**6.11**	**6.06**	$>10^4$	$>10^4$	$>10^4$	$>10^4$	$>10^4$
	YaRN (s=32)	128k	YaRN	6.38	6.25	6.16	6.11	**6.12**	$>10^4$	$>10^4$	$>10^4$	$>10^4$
	LongRoPE-2048k (ft=128k)	**2048k**	LongRoPE	6.53	6.35	6.24	6.18	6.17	**6.17**	**6.36**	**6.83**	**7.80**
	LongRoPE-2048k (ft=256k)	**2048k**	LongRoPE	6.79	6.66	6.31	6.27	6.21	**6.17**	**6.17**	**6.35**	**7.08**
Mistral-7B	Mistral v0.1	8k	-	**6.32**	66.61	$>10^2$	$>10^3$	$>10^3$	$>10^3$	-	-	-
	YaRN (s=8)	64k	YaRN	6.59	6.48	6.42	6.45	104.15	727.20	$>10^3$	$>10^4$	$>10^4$
	YaRN (s=16)	128k	YaRN	6.70	6.63	6.65	6.72	6.85	99.90	$>10^3$	$>10^4$	$>10^4$
	LongRoPE-2048k (ft=128k)	**2048k**	LongRoPE	6.42	**6.25**	**6.14**	**6.18**	**6.31**	**6.51**	**6.93**	**7.51**	**9.48**
	LongRoPE-2048k (ft=256k)	**2048k**	LongRoPE	6.44	**6.28**	**6.19**	**6.19**	**6.35**	**6.61**	**7.40**	**7.75**	**11.25**

Table 7. Perplexity evaluation within 256k context length on PG19.

Base LLM	Model Name	Context Window	Extension Method	Evaluation Context Length 8k	64k	128k
LLaMA2-7B	LongLoRA	100k	PI	7.16	6.81	$>10^3$
	Code LLaMA	100k	NTK	7.58	8.92	16.80
	LongRoPE-2048k (ft=128k)	2048k	LongRoPE	**6.98**	**6.59**	**6.35**
	LongRoPE-2048k (ft=256k)	2048k	LongRoPE	7.37	**6.64**	**6.31**
Mistral-7B	YaRN (s=8)	64k	YaRN	7.12	7.17	$>10^3$
	YaRN (s=16)	128k	YaRN	7.30	7.53	7.32
	LongRoPE-2048k (ft=128k)	2048k	LongRoPE	7.13	**7.01**	**7.02**
	LongRoPE-2048k (ft=256k)	2048k	LongRoPE	**7.10**	**6.98**	**7.13**

ability to further extend context window after fine-tuning.

Passkey retrieval. We now study the effective context window size in generation tasks. We follow a synthetic evaluation task of passkey retrieval proposed by (Mohtashami & Jaggi, 2023). In this task, the model is asked to retrieve a random passkey (i.e., a five-digit number) hidden in long document. The prompt template is detailed in appendix. We perform 10 iterations of the passkey retrieval task with the passkey placed at a random location uniformly distributed across the evaluation context length.

Fig. 3 shows the retrieval accuracy comparison with baselines. Existing models' accuracy rapidly drops to 0 beyond 128k. In contrast, despite the very challenging task of retrieving a passkey from million-level tokens, our LongRoPE-LLaMA2-2048k (ft=256k) manage to maintain a high retrieval accuracy (≥90%) from 4k to 2048k. LongRoPE-Mistral-2048k (ft=128k) keeps 100% accuracy up to 1800k, dropping to 60% at 2048k, aligning with expectations from Table 6, where the perplexity slightly increases at 2048k.

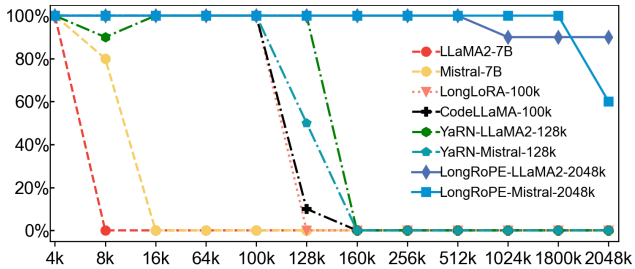

Figure 3. Passkey retrieval accuracy of long-context LLMs. It showcases the remarkable ability of our models to accurately retrieve a passkey from a vast pool of million-level tokens.

Standard benchmarks within original context window. We evaluate LongRoPE-2048k models on the original context window using Hugging Face Open LLM Leaderboard (Face, 2024) in zero-shot and few-shot settings. We use 25-shot ARC-Challenge (Clark et al., 2018). 10-shot HellaSwag (Zellers et al., 2019), 5-shot MMLU (Hendrycks et al., 2020), and 0-shot TruthfulQA (Lin et al., 2021).

As Table 8 shows, our models achieve comparable results on the original benchmark designed for a smaller context window, and even outperform the original model. For exam-

Table 8. Comparison of long-context LLMs with original LLaMA2 and Mistral on the Hugging Face Open LLM benchmark.

Model	Context Window	ARC-c	HellaSwag	MMLU	TruthfulQA
(a) LLaMA2-7B with extended context window					
Original LLaMA2-7B	4k	53.1	78.6	46.6	39.0
Together	32k	47.6	76.1	43.3	39.2
Code LLaMA	100k	42.4	64.8	40.1	37.1
YaRN (s=16)	64k	52.4	**78.7**	42.4	38.2
YaRN (s=32)	128k	52.2	78.5	41.8	37.4
LongRoPE-2048k (ft=128k)	2048k	**53.3**	77.6	**45.2**	**39.6**
LongRoPE-2048k (ft=256k)	2048k	**54.1**	77.8	**44.4**	38.9
(b) Mistral-7B with extended context window					
Original Mistral-7B	8k	60.6	83.2	63.6	42.6
MistralLite (Amazon, 2023)	16k	59.2	81.6	50.4	38.3
YaRN (s=8)	64k	59.3	81.3	**61.3**	42.5
YaRN (s=16)	128k	59.0	80.5	60.5	42.5
LongRoPE-2048k (ft=128k)	2048k	59.0	**81.7**	60.9	**43.9**
LongRoPE-2048k (ft=256k)	2048k	**59.8**	81.4	60.9	**44.1**

Table 9. Books3 perplexity comparison of extending LLaMA2-256k via different secondary positional interpolation methods.

Model Name	Extension Method	Context Window Size 512k	1024k	2048k
LLaMA2-7B (ft=256k)	PI	6.60	8.73	20.17
	YaRN	6.39	6.79	8.27
	LongRoPE	**6.17**	**6.35**	**7.08**

ple, our long context LLMs outperform LLaMA2-7B on the ARC-c and exceed Mistral on TruthfulQA.

4.3. Ablation Results

Effectiveness of the second positional interpolation. In our progressive extension strategy, we use our search algorithm to conduct a second non-uniform positional interpolation on the fine-tuned extended LLMs. We validate its effectiveness by running experiments on our fine-tuned LLaMA2-256k model. We extend it to 512k, 1024k, and 2048k using PI and YaRN. As Table 9 shows, our non-uniform positional interpolation sustains a consistent level of perplexity. In contrast, the perplexity under PI and YaRN quickly increases with the extension ratio.

Effectiveness of recovery at shorter context lengths. To mitigate performance loss at shorter context lengths, we readjust the RoPE factors for LongRoPE-2048k via our search algorithm. Specifically, we decrease the maximum allowable scale factors for the search to encourage less interpolation at short 4k and 8k lengths. Table 10 shows the perplexity comparison of LongRoPE-LLaMA2-2048k on Proof-pile at 4k and 8k lengths, along with the average LLM benchmark accuracy. The results clearly demonstrate a significant performance improvement at short context lengths.

Analysis on the two forms of non-uniformities. Finally, we ablate on the two non-uniformities to see how each part contributes to the performance. We setup two experiments: (i) extending LLaMA2-7B to short 16k and 32k using different methods—PI, searching for RoPE dimension only, and searching for both non-uniformities; (ii) extending our fine-tuned 256k-length LLaMA2 to 2048k following the same procedure. The perplexity is evaluated without fine-tuning. As Table 11 shows, non-uniformity in RoPE dimension significantly reduces perplexity compared to PI's linear interpolation. Non-uniformity in token position clearly improves performance at 16k and 32k lengths but does not show the same impact at 2048k, possibly due to the extremely long length. Preserving only the initial tokens without interpolation becomes non-useful, and we leave this as future work.

Table 10. Ablation study on LongRoPE readjustment for performance recovery at shorter context lengths.

FT Model	With Recovery	Proof-Pile Perplexity 4k	8k	LLM Benchmark Avg. Accuracy
LLaMA2-7B-2048k (ft=128k)	×	4.16	3.72	49.3
	✓	**3.67**	**3.49**	**53.9**
LLaMA2-7B-2048k (ft=256k)	×	4.51	3.82	47.9
	✓	**3.69**	**3.64**	**53.8**

Table 11. Ablation study on the two forms of non-uniformities.

Methods	LLaMA2-7B PG19 Perplexity 16k	32k	LLaMA2-7B (ft=256k) Books3 Perplexity 2048k
Linear interpolation (PI)	14.88	136.30	20.17
RoPE dim (Ours)	7.28	13.00	7.08
RoPE dim+Start tokens (Ours)	**7.22**	**11.51**	**7.08**

5. Related Works

In addition to methods based on position interpolation, this section discusses related works of other approaches.

Retrieval-based approaches use an external memory module to memorize long past context and retrieval modules for related documents fetching at inference (Tworkowski et al., 2023; Wang et al., 2023; Borgeaud et al., 2022). These designs typically need explicit modifications on the LLM architectures. Our work, in contrast, is more lightweight, with minor positional embedding modifications. We can also handle more long context tasks beyond retrieval, such as long document summarization and few-shot learning.

Attention-based context window extensions. Beyond positional embedding interpolation, some research achieves input context extension using the original LLM context window length by manipulating attention mechanisms (Han et al., 2023; Xiao et al., 2023; Ratner et al., 2022). The key idea is to mitigate the attention explosion issue caused by new positions using novel attention masks. These efforts and positional interpolation methods are complementary.

Fine-tuning based approaches focus on how to effectively fine-tune pre-trained LLMs with modified position embeddings for longer context. Works like Code LLaMA (Rozière et al., 2023), LLaMA2 Long (Xiong et al., 2023) and ScaledRoPE (Liu et al., 2023) choose a very large base value for RoPE and fine-tune on the target length. Our method

offers flexibility for various target lengths and can achieve beyond 2M length. More recently, as fine-tuning for long context lengths (i.e., over 128k) demands substantial GPU resources, LongLoRA (Chen et al., 2023b) and PoSE (Zhu et al., 2023) are proposed to mitigate this overhead. Our method is orthogonal to these efficient fine-tuning works.

6. Conclusion

In this work, we present LongRoPE, a method that remarkably extends the context length of LLMs to an unprecedented 2048k, while maintaining their capabilities within original shorter context window. We exploit two forms of non-uniformities in RoPE positional embedding using an efficient evolutionary search. This offers twofold benefits: it provides good initialization for fine-tuning and enables an $8\times$ context window extension without fine-tuning. Building on this, we propose a progressive extension strategy using 256k-length fine-tuned LLMs to reach a 2048k context window size without extra fine-tuning. Extensive experiments validate the effectiveness of LongRoPE. We envision that our LongRoPE-2048k models will enable many new long context applications and inspire further research.

Impact Statement

The contribution of LongRoPE is significant. First, LongRoPE stands out as the first publicly available method to break through the previous 128k context window size, to an impressive 2 million tokens. Moreover, LongRoPE has a wide applicability. It is applicable to various LLMs based on RoPE, which will promote the development of long-context LLMs and downstream long context applications. Second, LLMs with a 2048k context window will enable many new long-context applications and inspire further research. For instance, how to effectively utilize such an extremely long context window is a topic worth exploring. Third, built on open-source pre-trained LLMs, LongRoPE minimizes training resources and fine-tuning resources compared to directly fine-tuning at 2 million text lengths, thus reducing the carbon footprint. Despite these advantages, the inference cost of the LongRoPE-2048k models is high when inputting with 2 million tokens, as the model inferences with full attention over 2 million tokens. We look forward to exploring ways to reduce the inference cost.

References

Long-data collections, 2024. URL https://huggingface.co/datasets/togethercomputer/RedPajama-Data-1T.

Amazon. Mistrallite, 2023. URL https://huggingface.co/amazon/MistralLite.

Azerbayev, Z., Ayers, E., and Piotrowski, B. Proof-pile, 2022. URL https://github.com/zhangir-azerbayev/ProofNet.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G. B., Lespiau, J.-B., Damoc, B., Clark, A., et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pp. 2206–2240. PMLR, 2022.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023a.

Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. Longlora: Efficient fine-tuning of long-context large language models. *arXiv:2309.12307*, 2023b.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Computer, T. Redpajama: An open source recipe to reproduce llama training dataset, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning. 2023.

Face, H. Open llm leaderboard, 2024. URL https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J. Single path one-shot neural architecture search with uniform sampling. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pp. 544–560. Springer, 2020.

Han, C., Wang, Q., Xiong, W., Chen, Y., Ji, H., and Wang, S. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Huang, X., Zhang, L. L., Cheng, K.-T., and Yang, M. Boosting llm reasoning: Push the limits of few-shot learning with reinforced in-context pruning. *arXiv preprint arXiv:2312.08901*, 2023.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023.

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Lin, Z., Miao, Y., Liu, G., Shi, X., Zhang, Q., Yang, F., Maleki, S., Zhu, Y., Cao, X., Li, C., et al. Superscaler: Supporting flexible dnn parallelization via a unified abstraction. *arXiv preprint arXiv:2301.08984*, 2023.

Liu, X., Yan, H., Zhang, S., An, C., Qiu, X., and Lin, D. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*, 2023.

LocalLLaMA. Dynamically scaled rope further increases performance of long context llama with zero fine-tuning, 2023a. URL https://www.reddit.com/r/LocalLLaMA/comments/14mrgpr/dynamically_scaled_rope_further_increases/.

LocalLLaMA. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degration, 2023b. URL https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback, 2023.

Mohtashami, A. and Jaggi, M. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

OpenAI, :, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng,

T., Zhuang, J., Zhuk, W., and Zoph, B. Gpt-4 technical report, 2023.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023.

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

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.05507.

Ratner, N., Levine, Y., Belinkov, Y., Ram, O., Abend, O., Karpas, E., Shashua, A., Leyton-Brown, K., and Shoham, Y. Parallel context windows improve in-context learning of large language models. *arXiv preprint arXiv:2212.10947*, 2022.

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C. C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., and Synnaeve, G. Code llama: Open foundation models for code, 2023.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33: 7537–7547, 2020.

Together, 2023. URL https://huggingface.co/togethercomputer/LLaMA-2-7B-32K.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023.

Tworkowski, S., Staniszewski, K., Pacek, M., Wu, Y., Michalewski, H., and Miłoś, P. Focused transformer: Contrastive training for context scaling. 2023.

Wang, W., Dong, L., Cheng, H., Liu, X., Yan, X., Gao, J., and Wei, F. Augmenting language models with long-term memory. *arXiv preprint arXiv:2306.07174*, 2023.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv*, 2023.

Xiong, W., Liu, J., Molybog, I., Zhang, H., Bhargava, P., Hou, R., Martin, L., Rungta, R., Sankararaman, K. A., Oguz, B., Khabsa, M., Fang, H., Mehdad, Y., Narang, S., Malik, K., Fan, A., Bhosale, S., Edunov, S., Lewis, M., Wang, S., and Ma, H. Effective long-context scaling of foundation models, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Zhang, P., Liu, Z., Xiao, S., Shao, N., Ye, Q., and Dou, Z. Soaring from 4k to 400k: Extending llm's context with activation beacon. *arXiv preprint arXiv:2401.03462*, 2024.

Zhu, D., Yang, N., Wang, L., Song, Y., Wu, W., Wei, F., and Li, S. Pose: Efficient context window extension of llms via positional skip-wise training, 2023.

A. Appendix

A.1. Settings

Environments. All our experiments are conduct on 16 A100 GPUs. We employ Flash Attention-2 (Dao, 2023) to accelerate both training and inference. As the GPU memory and computation time increase exponentially with the sequence length, it's challenging to serve the fine-tuning and inference with context length beyond 512k. As a result, we utilize CUBE - an advanced version of (Lin et al., 2023), to reduce both the training and inference costs.

Passkey prompt. We follow existing literature (Mohtashami & Jaggi, 2023; Chen et al., 2023a; Peng et al., 2023; Chen et al., 2023b; Zhu et al., 2023) for the document format of passkey retrieval. We show the prompt template as follows:

```
There is an important info hidden inside a lot of irrelevant text. Find it and
memorize them. I will quiz you about the important information there.

The grass is green. The sky is blue. The sun is yellow. Here we go. There and
back again. (repeat x times)

The pass key is 17865. Remember it. 17865 is the pass key.

The grass is green. The sky is blue. The sun is yellow. Here we go. There and
back again. (repeat y times)

What is the pass key? The pass key is
```

The document length varies with the value of `x` and `y`. 17865 is the passkey number to retrieve. It is randomly sampled and varies at each testing time.

A.2. Additional details on fine-tuning

As introduced in Section 4.2, we fine-tune two context window lengths, namely 128k and 256k, for both LLaMA2 and Mistral. Specifically, the model with a 256k context window begins its fine-tuning from the 128k-length checkpoint. Fig. 4(ab) illustrates the training loss for LLaMA2 and Mistral during this fine-tuning process. We highlight three key observations: **(1)** The model with a 128k context window experiences a large initial loss due to a 32× extension. However, the loss rapidly decreases after a few steps. **(2)** LLaMA2 and Mistral employ different fine-tuning settings. Mistral achieves the desired long context window by fine-tuning on 16k-length data, while LLaMA2 necessitates text lengths that match the context window size. Furthermore, we adopt YaRN's strategy of using a constant learning rate. As a result, it can be observed that Mistral's loss begins to fluctuate after dropping to around 2.2. **(3)** For both Mistral and LLaMA2, the model with a 256k context window, which starts fine-tuning from the 128k checkpoint, exhibits a low initial training loss. This suggests that fine-tuning from 128k-length checkpoints is effective and significantly facilitates convergence.

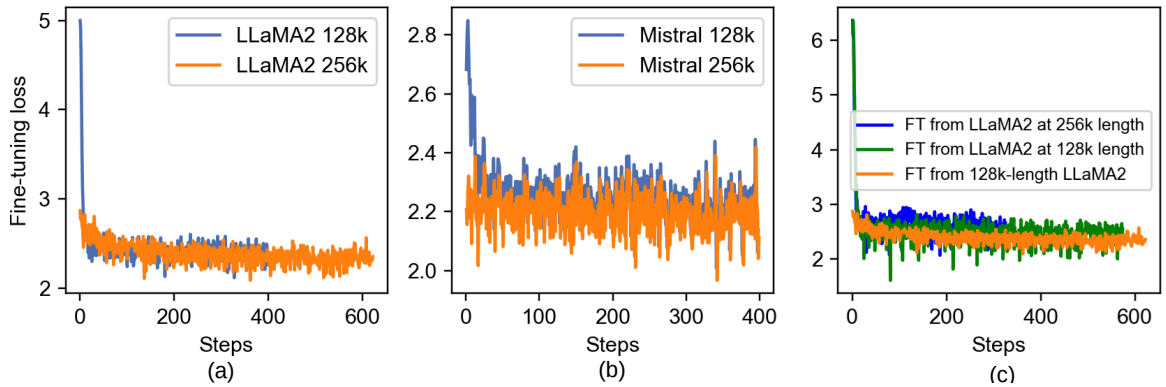

Figure 4. (ab): Loss curve in fine-tuning LLaMA2-7B and Mistral-7B with extended context window size. (c) The training loss of fine-tuning LLaMA2-7B with a 256k context window under different fine-tuning settings.

We also explore different settings to fine-tune LLaMA2 with 256k context window. As shown in Fig. 4(c), we experiment with two additional settings: (i) using the RoPE rescale factors corresponding to 256k, we directly fine-tune on LLaMA2-7B, and (ii) using RoPE rescale factors for 256k, we fine-tune on LLaMA2-7B, but truncate the text lengths to 128k. The loss

curves are displayed in Fig. 4(c). We observe that using 128k text lengths to fine-tune a model with a 256k context window results in a sharp increase in the initial loss. Directly fine-tuning from LLaMA2-7B to achieve 256k results in a relatively slow decrease in loss. Table 12 shows the test perplexity on Proof-Pile for checkpoints from three different settings. This indicates that our current approach of fine-tuning from a 128k-checkpoint is the most effective.

Table 12. Proof-pile perplexity of extended LLaMA2-7B via different fine-tuning settings. Tuples of three values represent the fine-tuning text length, context window size and initial checkpoint.

Method (fine-tune L', L', base LLM)	Evaluation Context Length 32768	65536	98304	131072	262144
(128k, 256k, LLaMA2-7B)	9.75	6.56	5.15	5.19	2.21
(256k, 256k, LLaMA2-7B)	4.51	2.87	2.53	2.39	1.95
(128k, 256k, LLaMA2-7B (ft=128k)	**2.66**	**2.38**	**2.28**	**2.26**	**1.87**

Fine-tuning cost. LLaMA2-128k uses 8 A100 GPUs for a week to fine-tune 400 steps. LLaMA2-256k doubles the resources to 16 A100 GPUs for two weeks to fine-tune 600 steps. For Mistral-128k and 256k, with a training length of 16k, we employ 4 A100 GPUs for a 2-day fine-tuning period.

Attention scaling. In addition to non-uniformly rescaled RoPE factors, we increase the attention entropy to further improve long context performance. When dealing with an extremely long context, the attention softmax logits become thinly dispersed across a large number of token positions. To mitigate this, we introduce a temperature, denoted as t, to alter the attention mechanism with a larger entropy. Let L represent the original context window size and s be the extension ratio, we modify the attention computation as follows:

$$softmax\left(\frac{\mathbf{q}_m^T \mathbf{k}_n}{t\sqrt{|D|}}\right); \sqrt{\frac{1}{t}} = 1 + \frac{\ln s}{\ln L} \tag{4}$$

where $\mathbf{q}_m$ and $\mathbf{k}_n$ are the query and key vectors for a specific attention head, which have been integrated with our rescaled RoPE embeddings.

A.3. Additional details on the search

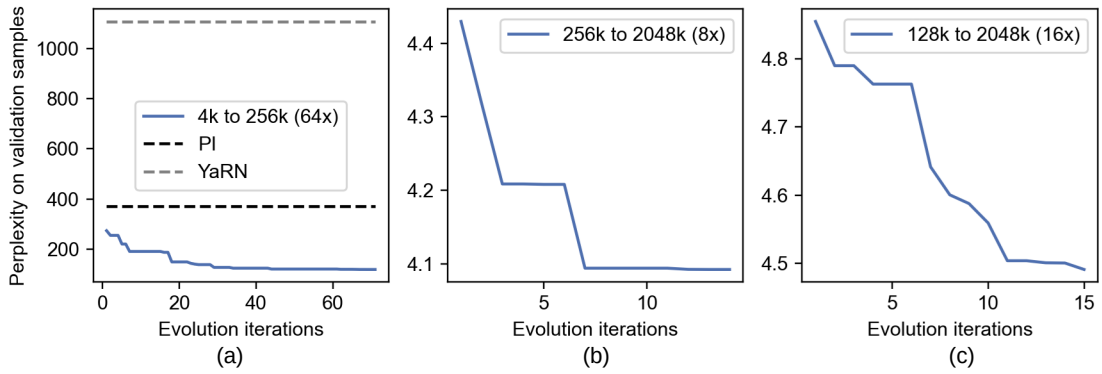

Figure 5. Perplexity on the validation samples at each evolution search iteration. (a) The 64× extension for LLaMA2-7B to reach 256k context window size. (b) The 8× extension for LLaMA2-7B-256k to reach 2048k context window size. (c) The 16× extension for LLaMA2-7B-128k to reach 2048k context window size.

Search efficiency. Fig. 5 illustrates the perplexity on the validation samples at each evolution search iteration. We can see that our search algorithm can efficiently find high-quality non-uniform RoPE rescale factors. Specifically, on the 256k context window search (Fig. 5(a)), after the first iteration, we can find solutions significantly better than PI and YaRN. As searching more iterations, we can significantly reduce the validation perplexity from 273.27 from 118.47. Furthermore, we can observe that YaRN, as the previous state-of-the-art non-uniform interpolation method, performs even worse than PI (linear interpolation) at the 64× extension. This also indicates that human-heuristic-based non-uniform interpolation is challenging to perform well in all scenarios.

For the extremely long context window at 2048k, we use the fine-tuned 128k and 256k context window's LLaMA2-7B for 16× and 8× extension, respectively. As shown in Fig. 5(bc), as expected, the perplexity of the 16× extension is larger than that of the 8× extension. Additionally, due to the time required for a single perplexity evaluation at 2048k is about 50 minutes, the search iterations are constrained. If more search time is allowed, it's highly possible to search better results.

Search cost. The search cost is primarily depending on the time required to evaluate the perplexity of input context at a given context window size. For context window lengths up to 256k, the total search time is relatively quick, achievable within 3 days using a single A100 GPU. For a 512k context window, we employ 2 A100 GPUs. For larger context windows of 1024k and 2048k, we utilize 4 and 8 A100 GPUs respectively, managing to keep the total search time within a 5-day limit.

A.4. Inference cost

Long context LLM can significantly increase the inference cost. Currently, after applying the memory optimization techniques such as flash attention, tensor parallelism with sequence partitioning, and kernel fusion, we can efficiently serve the inference of 2 million tokens on 8x80GB A100. The following are the detailed inference cost numbers. As shown in Table 13, with enabling KV cache during inference, the generation speed is rapid after the initial prefilling stage. Specifically, each token requires only 80ms-120ms for generation.

Table 13. Inference cost of LongRoPE-2048k models on 8×80GB A100.

Model	Seq len	KV Cache	Reserved (GB)	Prefill Peak (GB)	Prefill Time (s)	Generation Time (s/token)
LongRoPE-LLaMA2-7B-2048k	1M	Yes	58.62	77.08	214.31	0.080
LongRoPE-LLaMA2-7B-2048k	2M	No	2.37	47.39	808.76	808.76
LongRoPE-Mistral-2048k	2M	Yes	33.63	74.65	1020.35	0.121