PoSE: Efficient Context Window Extension of LLMs via Positional Skip-wise Training

Dawei Zhu [*†‡] **Nan Yang** [‡] **Liang Wang** [‡] **Yifan Song** [†] **Wenhao Wu** [†]
Furu Wei [‡] **Sujian Li** [† ✉]
[†] School of Computer Science, Peking University
[‡] Microsoft Corporation
https://github.com/dwzhu-pku/PoSE

Abstract

In this paper, we introduce **Po**sitional **S**kip-wis**E** (PoSE) training for efficient adaptation of large language models (LLMs) to extremely long context windows. PoSE decouples train length from target context window size by simulating long inputs using a fixed context window with manipulated position indices during training. Concretely, we select several short chunks from a long input sequence, and introduce distinct skipping bias terms to modify the position indices of each chunk. These bias terms, along with the length of each chunk, are altered for each training example, allowing the model to adapt to all positions within the target context window without training on full length inputs. Experiments show that, compared with fine-tuning on the full length, PoSE greatly reduces memory and time overhead with minimal impact on performance. Leveraging this advantage, we have successfully extended the LLaMA model to 128k tokens. Furthermore, we empirically confirm that PoSE is compatible with all RoPE-based LLMs and various position interpolation strategies. Notably, by decoupling fine-tuning length from target context window, PoSE can theoretically extend the context window infinitely, constrained only by memory usage for inference. With ongoing advancements for efficient inference, we believe PoSE holds great promise for scaling the context window even further.

1 Introduction

Large Language Models (LLMs) have revolutionized language modeling and demonstrated impressive abilities to perform various tasks [2]. However, even with their remarkable capacity, these LLMs remain restricted by pre-defined context window sizes, suffering from notable performance decline when input tokens exceeds these limits. Nevertheless, numerous application scenarios demand extremely long input sequences, including long document summarization [13], in-context learning with numerous examples [18], and long document retrieval [38], etc. This naturally poses a significant challenge of **context window extension**: Extending the context window of a pre-trained LLM to accommodate longer sequences.

Naively extending pre-trained models to longer context windows by fine-tuning on longer inputs has received limited success due to the large disruption introduced by the new position indices [4]. In response to this challenge, various Position Interpolation (PI) strategies have been proposed [4, 22], demonstrating improved context window extension performance when fine-tuned on sequences of the target length. Nevertheless, even with the use of position interpolation, fine-tuning LLMs on sequences of the target length, referred to as *Full-length* fine-tuning in this paper, remains both

[*]Work done during internship at MSRA.

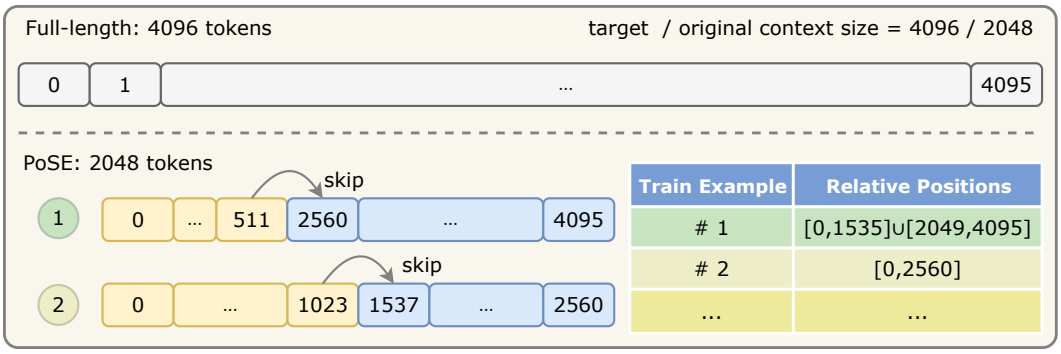


Figure 1: Position indices of Full-length fine-tuning v.s. PoSE fine-tuning for extending the context window size from 2,048 to 4,096. At each iteration, the former directly takes 4,096 tokens for fine-tuning, while PoSE manipulates the position indices of 2,048 tokens to simulate longer inputs. For example, we partition the original context window of 2,048 tokens into two chunks, and adjust the position indices of the second chunk by adding a distinct skipping bias term. These bias terms, as well as the length of each chunk, are altered for each training example, so that the model can adapt to all relative positions of the target context window through fine-tuning.

memory and time-intensive. Moreover, it becomes impractical when the target context window size is extremely large, due to the computational complexity that increases quadratically with input length.

In this paper, we introduce **Po**sitional **S**kip-wis**E** (PoSE) fine-tuning to decouple the fine-tuning length from the target context window length, unleashing the possibility of efficiently extending context window to an extreme size. The key idea of PoSE is to simulate long inputs by manipulating position indices within a fixed context window. As depicted in Figure 1, we partition the original context window into several chunks, and adjust the position indices of each chunk by adding a distinct skipping bias term. These bias terms, as well as the length of each chunk, are altered for each training example, so that the model can adapt to all positions (including both absolute and relative) within the target context window through fine-tuning. Meanwhile, by maintaining continuous position indices within each chunk, PoSE bears a close resemblance to pre-training. As a result, the model's pre-trained capacity for language modeling and comprehension is retained to the greatest degree.

The advantages of our PoSE are threefold: **1) Memory and Time Efficiency:** By only requiring the original context size for fine-tuning, PoSE circumvents the quadratic increase in computational complexity with respect to target length during the fine-tuning stage, thereby significantly reducing memory and time overhead. 2) **Potential for Extremely-Long Context:** We manage to extend the context window of LLaMA [31] by up to 64 times (2k→128k, k=1,024) while preserving decent ability of language modeling and understanding. 3) **Compatible with all RoPE-based LLMs and PI strategies:** The effectiveness of PoSE has been empirically validated across several RoPE-based LLMs, including LLaMA, LLaMA2 [32], and GPT-J [33]. Additionally, PoSE has been demonstrated to be compatible with a variety of position interpolation methods, such as linear [4], Neural Tangent Kernel (NTK) [21], and YaRN [22] interpolation.

Notably, by decoupling the fine-tuning length and the target context window length, PoSE can theoretically extend context window to an infinite length. The only constraint is the memory usage during the inference phase. Hopefully, with the continuous advancements in efficient inference techniques, including Flash Attention [9, 8], xFormers [17], vLLM [16], etc, we believe PoSE can promisingly push the context window size to a even larger scale.

2 Related Work

2.1 Length Extrapolation

Length extrapolation aims to ensure that the model continues to perform well, even when the number of input tokens during inference exceeds the size of the context window on which the model is trained [23]. To this end, a series of positional embedding schemes have been proposed, including ALibi [23], xPos [30], NoPos [12], etc.

Similar to our work, Ruoss et al. [28] also attempted to simulate longer sequences during training time to mitigate out-of-distribution lengths. They proposed randomized positional encoding (RandPos), which randomly selected an ordered subset of position indices from longer sequences. Our proposed method, PoSE, diverges from their approach in several key aspects: First, RandPos is a positional embedding scheme designed for pre-training encoder-only models from scratch to enhance length generalization ability. In contrast, PoSE is a fine-tuning method that aims to efficiently extend the context window of pre-trained LLMs, the majority of which follow a decoder-only architecture. Second, in RandPos, the position indices between adjacent tokens are not continuous. However, in PoSE, the position indices within each chunk are intentionally made continuous to closely resemble the pre-training phase, therefore reducing the risk of disrupting the abilities learned during the pre-training stage.

2.2 Context Window Extension

Differing from length extrapolation, which primarily involves training a model from scratch to support lengths exceeding those it was initially trained for, context window extension focuses on extending the context window of a pre-trained LLM. Directly fine-tune an existing pre-trained Transformer with a longer context window has been shown to progress slowly [4]. To expedite and stabilize training, Chen et al. [4] first down-scaled position indices to match original context size through linear position interpolation, followed by full-length fine-tuning. Subsequently, a range of positional interpolation strategies have been introduced, including NTK [21] and YaRN [22]. While also leveraging position interpolation, our method managed to simulate long inputs by manipulating position indices, requiring only the original context size for fine-tuning.

2.3 Memory Transformers

An alternative strategy for managing exceedingly lengthy input sequences involves the implementation of memory mechanisms [7, 3, 35, 34, 20]. Generally, there are two predominant techniques for exploiting memory: recurrence-based and retrieval-based methodologies. With respect to recurrence-based approaches, the Transformer-XL [7] processes inputs within fixed-length windows, allowing each token to attend not only to tokens within the current window but also those within the preceding window. The RMT [3] incorporates dedicated memory tokens at the beginning of the input sequence for the purpose of encoding and transmitting memory. However, a significant limitation of this approach is its reliance on a compressed memory format, which may result in the loss of information, and its restricted capacity for random access.

In the retrieval-based paradigm, Wu et al. [35] encode previous sequences as (key, value) pairs. These pairs are subsequently retrieved through kNN lookup. LongMem [34] further advances this approach by incorporating a frozen Language Model as a memory encoder. This is supplemented by an adaptive residual side-network, which operates as both a memory retriever and reader. The major constraint of this approach is the lack of interaction between distinct memory chunks.

More recently, Mohtashami and Jaggi [20] introduced landmark attention, which facilitates random access to any chunk of the input by introducing landmark tokens. In contrast, our method achieves full access to the entire input without requiring any modifications to the attention mechanism.

3 Methodology

3.1 Preliminaries

Rotary Position Embedding (RoPE). The use of RoPE [29] has become pervasive in contemporary LLMs, including LLaMA [31], GPT-J [33], etc. It encodes position information of tokens with a rotation matrix that naturally incorporates explicit relative position dependency. To elucidate, given a hidden vector $\boldsymbol{h} = [h_0, h_1, ..., h_{d-1}]$, where d is the hidden dimension, and a position index m,

RoPE operates as follows:

$$f(\boldsymbol{h}, m) = \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_{d-2} \\ h_{d-1} \end{pmatrix} \otimes \begin{pmatrix} \cos m\theta_0 \\ \cos m\theta_0 \\ \cos m\theta_1 \\ \cos m\theta_1 \\ \vdots \\ \cos m\theta_{d/2-1} \\ \cos m\theta_{d/2-1} \end{pmatrix} + \begin{pmatrix} -h_1 \\ h_0 \\ -h_3 \\ h_2 \\ \vdots \\ -h_{d-1} \\ h_{d-2} \end{pmatrix} \otimes \begin{pmatrix} \sin m\theta_0 \\ \sin m\theta_0 \\ \sin m\theta_1 \\ \sin m\theta_1 \\ \vdots \\ \sin m\theta_{d/2-1} \\ \sin m\theta_{d/2-1} \end{pmatrix} \tag{1}$$

where $\theta_j = 10000^{-2j/d}, j \in \{0, 1, ..., d/2-1\}$. Unlike previous absolute position encodings that are directly applied to the input vector $\boldsymbol{x}$, RoPE is employed on the query and key vectors at each layer. Considering a query vector $\boldsymbol{q}$ at position m and a key vector $\boldsymbol{k}$ at position n, the attention score $a(\boldsymbol{q}, \boldsymbol{k})$ is defined as follows:

$$\begin{aligned} a(\boldsymbol{q}, \boldsymbol{k}) &= < f(\boldsymbol{q}, m), f(\boldsymbol{k}, n) > \\ &= \sum_{j=0}^{d/2-1} \left[(q_{2j}k_{2j} + q_{2j+1}k_{2j+1}) \cos{(m-n)\theta_j} + (q_{2j}k_{2j+1} - q_{2j+1}k_{2j}) \sin{(m-n)\theta_j} \right] \\ &:= g(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{\theta}, m-n) \end{aligned} \tag{2}$$

Hence, RoPE encodes position information in a relative manner, as the attention score depends on the relative distances between positions rather than their absolute position values.

Problem Formulation. Given a Large Language Model pre-trained with a context window size of L_c, our objective is to extend this context size to a target length L_t, so that the model maintains good performance when processing input sequences containing a maximum of L_t tokens.

Position Interpolation (PI). In contrast to directly extending the position indices to $L_t - 1$ when dealing with an input text $\boldsymbol{x} = \{x_0, x_1, ..., x_{L_t}\}$, position interpolation down-scales the position indices to align with the original context window size L_c. This approach effectively mitigates the risk of encountering extreme values and has been empirically demonstrated to enhance stability during fine-tuning. Various interpolation strategies have been proposed, with $\alpha = L_t/L_c$ denoting the scaling factor:

- *Linear Interpolation.* As described by Chen et al. [4] and kaiokendev [14], linear interpolation involves a proportional down-scaling of the position index m to m/α. Consequently, the attention score between a query $\boldsymbol{q}$ at position m and a key $\boldsymbol{k}$ at position n becomes $g(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{\theta}, (m-n)/\alpha)$, as defined in Equation 2. Theoretical analysis has substantiated that the interpolated attention score exhibits significantly greater stability compared to the extrapolated counterpart.
- *Neural Tangent Kernel (NTK) Interpolation.* In contrast to linear interpolation, NTK Interpolation alters the base of RoPE, effectively modifying the rotational "speed" of each dimension of RoPE [21]. Specifically, the original $\theta_j = 10000^{-2j/d}, j \in \{0, 1, ..., d/2-1\}$ in RoPE is transformed into $\theta'_j = (10000\lambda)^{-2j/d}$, where $\lambda = \alpha^{d/d-2}$. It is noteworthy that the value of λ is chosen to ensure that $m\theta'_{d/2-1} = (m/\alpha)\theta_{d/2-1}$.
- *YaRN Interpolation.* Different from Linear and NTK interpolation that treat each dimension of RoPE equally, YaRN [22] employs a ramp function to combine Linear and NTK interpolation at varying proportions across different dimensions. Simultaneously, it introduces a temperature factor to mitigate distribution shift of attention matrix caused by long inputs.

3.2 Proposed Approach: Positional Skip-wise Training (PoSE)

Although position interpolation effectively addresses out-of-distribution position indices, extending to an extreme length by fine-tuning on context window of this size remains impractical, owing to the quadratic growth in computational complexity of attention as sequence length increases. Instead, we explore to train within the original context window L_c and achieve context window extension via manipulating position indices to simulate longer inputs.

There are two designing desiderata for this endeavor: First, to avoid out-of-distribution positions during inference, the relative distance of manipulated position indices should comprehensively cover the range of $\{0, 1, \ldots, L_t - 1\}$. Second, fine-tuning with the manipulated position indices should not harm the original abilities of LLMs, so the structure of manipulated position indices should closely adhere to the original structure to the greatest extent possible.

Initially, we divide the original context window L_c into N chunks $c_0, c_1, \ldots, c_{N-1}$, each with lengths $l_0, l_1, \ldots, l_{N-1}$, where $\sum_{i=0}^{N-1} l_i = L_c$. We introduce the starting index st_i for each chunk c_i, which facilitates the formulation of its position indices as follows:

$$\text{Pos}(c_i) = \{st_i, st_i + 1, \ldots, st_i + l_i - 1\}, \quad st_i = \sum_{j=0}^{i-1} l_j \tag{3}$$

Subsequently, we employ the discrete uniform distribution $\mathcal{U}(S)$ to sample a *skipping bias* term $u_i \sim \mathcal{U}(\{u_{i-1}, \ldots, L_t - \sum_{j=i}^{N} l_j\})$ for each chunk c_i. This bias term is applied to the corresponding chunk to transform the original position indices into:

$$\text{PoSE}(c_i) = \{u_i + st_i, u_i + st_i + 1, \ldots, u_i + st_i + l_i - 1\} \tag{4}$$

Note that the constraint of $u_i > u_{i-1}$ is applied to prevent position index overlaps between chunks.

Intuitively, the introduction of skipping bias terms exposes model to a more diverse range of relative positions. It is essential to highlight that, at each iteration, we dynamically adjust both the length and skipping bias term for each chunk to achieve comprehensive coverage of the target context window. Moreover, the continuity of position indices within each chunk closely resembles the structure employed during pre-training. Consequently, fine-tuning the model on these new position indices for language modeling does not compromise its original capabilities.

Concerning the text contained within each chunk, a similar procedure is followed to select continuous spans of tokens from the input text $\boldsymbol{x} = \{x_0, x_1, ...\}$. To elaborate, we begin by sampling a bias term $v_i \sim \mathcal{U}(\{v_{i-1}, \ldots, L_t - \sum_{j=i}^{N} l_j\})$ followed by assigning the content of chunk c_i as below:

$$c_i = \boldsymbol{x}[v_i + st_i : v_i + st_i + l_i] \tag{5}$$

It is noteworthy that we have conducted experiments under various settings, including scenarios where $v_i = 0$, which results in genuinely continuous content for the chunks, or $v_i = u_i$, aligning the manipulated position indices with actual positions in the original text. However, we observe that these variations have relatively little impact on the outcomes of fine-tuning.

After position indices for each chunk is settled, we perform position interpolation for stabilized fine-tuning of language modeling. For simplicity, we set the initial bias terms u_0 and v_0 to 0. Number of chunks explored in this paper is $N = 2$.

4 Experiments

In this section, we conduct experiments to verify the effectiveness of PoSE for context window extension. Our method demonstrates impressive results on context lengths of both 16k and 32k for language modeling as well as passkey retrieval. Other advantages of PoSE are elaborated upon in Section 5.

4.1 Setups

Training Procedure. For each setting in the main experiments, we train LLaMA-7B with the next token prediction objective. This training process comprises 1,000 steps, employing a global batch size of 64 on 8 V100 GPUs using Deepspeed ZeRO stage 3 [27]. The fine-tuning dataset is sourced from The Pile [11], with a minimum length requirement of 2,048 tokens. Our default choice for interpolation strategies is linear interpolation. During the inference phase, we employ Flash Attention V2 [8] to enable the model to process extremely long documents, accommodating up to 128k tokens (k=1,024) on a single A100 GPU.

Table 1: Perplexity of models trained with different methods on GovReport and Proof-pile. Our PoSE, with a fixed training window size of 2k, effectively extended to a target context size of 16k / 32k for inference while receiving only minimal performance degradation compared to Full-length.

Method	Context size	GovReport					Proof-pile				
	Train / Target	2k	4k	8k	16k	32k	2k	4k	8k	16k	32k
None	- / -	4.74	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$	2.83	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$
Full-length	16k / 16k	4.87	4.70	4.61	4.59	-	2.93	2.71	2.58	2.53	-
RandPos	2k / 16k	11.63	11.17	11.54	15.16	-	7.26	6.83	6.76	7.73	-
	2k / 32k	93.43	95.85	91.79	93.22	97.57	60.74	63.54	60.56	63.15	66.47
PoSE (Ours)	2k / 16k	4.84	4.68	4.60	4.79	-	2.95	2.74	2.61	2.65	-
	2k / 32k	4.91	4.76	4.68	4.64	4.74	3.01	2.78	2.66	2.60	2.62

Evaluation Tasks and Datasets. We examine the ability of long text modeling on two tasks: language modeling and passkey retrieval. The language modeling task is a fundamental task that reflects the overall capability of a model in handling long text. Passkey retrieval, on the other hand, can effectively measure the maximum distance that a token can attend to during the inference stage. For language modeling, we use two datasets, GovReport [13] and Proof-pile [37]. For passkey retrieval, we follow Mohtashami and Jaggi [20] to construct synthetic prompts for evaluation.

Baseline Methods. We compare our PoSE training method against following baselines:

- *Full-length* fine-tuning takes input tokens of target length for fine-tuning. For this method, computation complexity scales quadratically with target context window size.
- *RandPos* [28] is initially designed to train an encoder-only model from scratch for length extrapolation. However, since it shares similar idea of simulating longer sequences via changing position indices, we include it for a comprehensive comparison. Given the original context length L_c and the target context length L_t, it uniquely samples L_c positions from the set $\{0, ..., L_t - 1\}$, arranges them in ascending order, and employs them as new position indices for training.

4.2 Language Modeling

First, we investigate the impacts of different fine-tuning methods on long sequence language modeling using the GovReport and Proof-pile datasets. GovReport is a summarization dataset comprising 19,402 reports published by the Congress and the U.S. Government, with an average document length of 7,866 tokens. We randomly select 50 reports containing more than 32,768 tokens for evaluation. Similarly, Proof-pile is a mathematical dataset of 13 gigabytes containing numerous lengthy mathematical documents. In line with the approach taken for GovReport, we choose 50 samples from Proof-pile that contain more than 32,768 tokens for evaluation.

Table 1 presents the results of scaling to 16k and 32k using Full-length training, RandPos, and PoSE. For each scaled model, as well as the non-fine-tuned LLaMA model (None), we report perplexity scores at various evaluation context window sizes, ranging from 2k to 32k, employing the sliding window approach proposed by Press et al. [23]. For evaluation efficiency, we set the stride of the sliding window to 1,024.

First, we observe an overall decreasing trend of perplexity for both models scaled to 16k and 32k via PoSE as evaluation context window size increases, proving their abilities to leverage longer context. Second, with significantly shorter context length during fine-tuning, our PoSE achieves comparable results with Full-length, consolidating its effectiveness. Third, our method achieves much stronger results than RandPos. We suppose it is because our manipulated position indices closely resembles that of pre-training, hereby preserving the pre-trained language modeling ability to the greatest extent.

We also notice that all the scaling methods suffers certain performance degradation as the supported context length increases. We perceive this as a trade-off between the quantity of tokens the model can process and the level of granularity in the attention the model can pay to each individual token.

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.

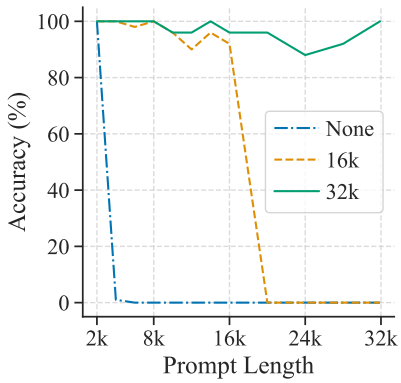
The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. (repeat x times)

The pass key is 81501. Remember it. 81501 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

(a)



(b)

Figure 2: (a) Prompt template used for passkey retrieval; (b) retrieval accuracy for the non-fine-tuned LLaMA model (*None*), and the PoSE-extended counterparts for 16k and 32k context window size. Both PoSE-extended models maintain a high retrieval accuracy ($\geq 90\%$) within their respective target context window.

4.3 Passkey Retrieval for Effective Context Window

To effectively measure the maximum distance that a token can attend to during the inference stage, we adopt the passkey retrieval test proposed by Mohtashami and Jaggi [20]. In this test, models are tasked with recovering a random passkey hidden within a lengthy document. Prompt template used for this task is presented in Figure 2a.

Specifically, we compare the non-fine-tuned LLaMA model (denoted as *None*) with the PoSE-extended version for 16k and 32k context window size. For each model, we vary the prompt length from 2k to 32k. In each case, we conduct the passkey retrieval test for 50 times, with a random passkey of 5 digits generated and placed at a random position inside the prompt for each trial. Figure 2b illustrates the results. For the none-fine-tuned LLaMA model (*None*), the retrieval accuracy rapidly drops to 0 when the prompt length exceeds 2k. In contrast, both PoSE-extended models managed to maintain a high retrieval accuracy ($\geq 90\%$) within their respective target context window. This indicates that models trained via PoSE genuinely possess the capability to attend to all tokens within the extended context windows.

5 Analysis

In this section, we analyze the advantages of PoSE, including 1) memory and time efficiency; 2) potential for extremely-long context; 3) compatibility with all RoPE-based LLMs and diverse interpolation strategies. In Section 5.4, We also verify that model performance within the original context window only receives minimal degradation.

5.1 Memory and Time Efficiency

We study the memory and time efficiency of PoSE compared with Full-length fine-tuning. For each method, we scale LLaMA-7B to 16k through 1,000 training steps with a global batch size of 64 on 8 V100 GPUs. Especially, for Full-length fine-tuning, we adjust training batch size to 2 and gradient accumulation steps to 4 on each device to achieve an equivalent configuration. Experiment results are demonstrated in Figure 3. Figure 3(a) illustrates time consumption for 1,000 steps of full training versus PoSE. While full training requires 56.37 hours to complete, our PoSE method only takes 6.82 hours, effectively achieving an eight-fold reduction in total training time. Figure 3(b) compares model perplexity of the two training methods at different steps. Notably, both models achieve relatively low perplexity levels within the initial 100 training steps. Moreover, at each step, our proposed PoSE, while keeping a context size of 2k tokens during training, exhibits very close language modeling ability to Full-length fine-tuning, which requires an extended training context of 16k. Consequently, we can confidently assert that our proposed approach is both memory and time-efficient.

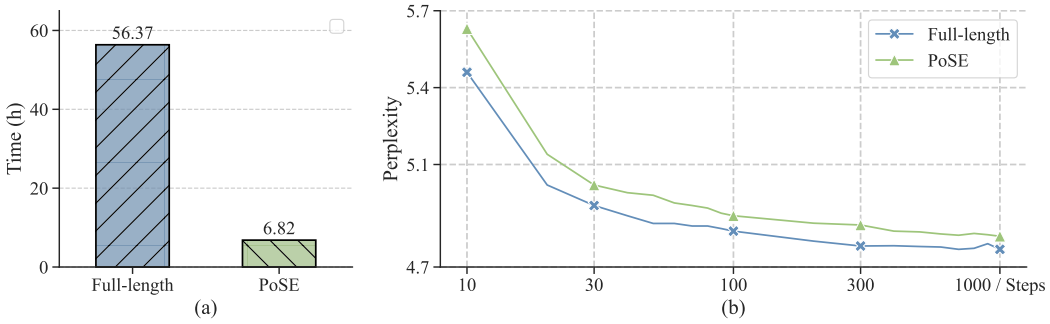


Figure 3: (a) Time consumption of Full-length fine-tuning v.s. PoSE for finishing 1,000 steps; (b) perplexity at different training steps of both models. We show that PoSE attains a comparable level of PPL performance to Full-length fine-tuning at each step, while concurrently achieving an eight-fold reduction in total training time.

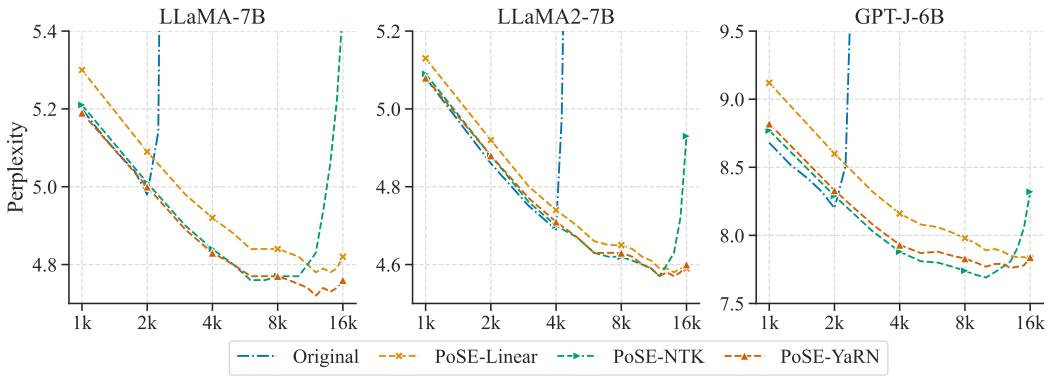


Figure 4: Perplexity of LLaMA-7B, LLaMA2-7B, GPT-J-6B extended to 16k via PoSE with Linear / NTK / YaRN interpolation, along with the non-fine-tuned *Original* model. The consistently low perplexity observed across all nine combinations serves as an indication of the effectiveness of our method across RoPE-based LLMs and diverse interpolation strategies.

5.2 Compatibility with RoPE-Based LLMs and Diverse Interpolation Strategies

We also delve into the effectiveness of PoSE when applied to different RoPE-based LLMs, as well as various interpolation strategies. Specifically, we employ PoSE on three distinct models: LLaMA-7B, LLaMA2-7B, and GPT-J-6B, all of which encompasses RoPE in their architectures. For each model, we examine the integration with Linear, NTK, and YaRN interpolation, as well as the non-fine-tuned original version for comparative purposes. The same GovReport dataset as described in Section 4.2 is utilized. The test set is truncated to the first 1k 16k tokens for plotting the perplexity curve, as depicted in Figure 4. First, it is evident that PoSE is effective across all three models and three interpolation strategies, as evidenced by the low perplexities achieved by all nine combinations in comparison to the non-fine-tuned original model. Second, we observe that NTK and YaRN interpolation generally yields superior results compared to Linear interpolation. However, it is noteworthy that NTK exhibits a significant increase in perplexity after a certain turning point, which occurs prior to reaching the target context length. This behavior is consistent with previous findings, indicating that for a given scaling factor α, NTK is unable to genuinely expand the context window by α times [21, 25, 22].

5.3 Potential for Extremely-Long Sequences

Because PoSE only takes a fixed context window at training stage to extend to target context window size, in theory we can extend LLMs to support infinite input lengths via PoSE. In this section, we extend context window size to 96k and 128k to explore PoSE's potential for context window extension. Given the need to assess documents of extreme length, we have opted to employ two

Table 2: Perplexity of models extended to extreme context size via PoSE on PG-19 and Books3. We show that our training method can effectively extend context window size to 128k when combined with YaRN interpolation.

Model	Gutenberg (PG-19)				Books3			
	32k	**64k**	**96k**	**128k**	**32k**	**64k**	**96k**	**128k**
PoSE-Linear-96k	10.18	11.11	13.57	-	9.98	10.90	13.42	-
PoSE-NTK-96k	7.98	20.39	38.73	-	8.29	20.82	40.39	-
PoSE-YaRN-96k	8.31	8.65	9.36	-	8.90	9.40	10.38	-
PoSE-Linear-128k	16.90	22.47	26.77	31.18	26.20	43.62	57.08	70.87
PoSE-NTK-128k	8.04	14.84	29.48	34.80	8.34	16.04	31.42	37.00
PoSE-YaRN-128k	9.32	10.36	10.77	11.33	10.56	12.30	13.07	13.81

Table 3: Performance of PoSE-extended LLaMA model on standard benchmarks in comparison with Full-length fine-tuning and the original LLaMA. We show that PoSE-extended models exhibit only marginal performance degradation compared with Full-length fine-tuning and the original version.

Model	Zero-Shot				Few-Shot	
	BoolQ	PIQA	WinoGrande	TruthfulQA	ARC-C	HellaSwag
LLaMA	75.11	78.67	69.85	34.08	51.19	77.75
Full-Linear-16k	70.95	77.64	69.06	31.89	48.55	74.19
Full-NTK-16k	75.80	78.08	68.98	33.83	48.81	76.57
Full-YaRN-16k	73.88	77.64	68.15	34.12	50.60	77.18
PoSE-Linear-16k	74.50	78.13	68.59	32.05	48.29	75.56
PoSE-Linear-128k	67.71	76.22	67.56	36.16	39.93	66.04
PoSE-NTK-16k	74.28	78.24	68.90	33.89	49.83	76.82
PoSE-NTK-128k	75.35	78.18	68.98	32.71	49.66	76.19
PoSE-YaRN-16k	74.28	78.02	69.06	34.00	49.23	77.04
PoSE-YaRN-128k	73.61	77.80	70.01	34.47	48.46	75.54

book datasets, namely Books3 [24] and Gutenberg (PG-19) [26]. Both of these datasets consist of extensive collections of literary works, rendering them well-suited subjects for the investigation of long-range modeling. For our evaluation, we randomly selected 20 books from each dataset, each containing more than 128k tokens.

Fine-tuning LLaMA models using PoSE, we experimented with Linear / NTK / YaRN interpolation for context extension of 96k and 128k, respectively. For each fine-tuned model, we adhere to the sliding window strategy adopted in Section 4.2 for perplexity calculation, with the only modification being an increase in the sliding window steps to 16k to enhance evaluation efficiency. The outcomes of these experiments are detailed in Table 2. Notably, we observe that, when coupled with linear interpolation, PoSE successfully extends the model's context window to 96k, and when paired with YaRN, it can further extends the context window to 128k.

5.4 Standard Benchmarks on Original Context Window Size

In this section, we examine the capabilities of the model within the original context window size using standard benchmarks. We combine the Hugging Face Open LLM Leaderboard [10] with a subset of LLaMA benchmarks to assess zero-shot and few-shot performance. For zero-shot evaluation, we employ BoolQ [5], PIQA [1], WinoGrande [15], and TruthfulQA [19]. For few-shot evaluation, we utilize 25-shot ARC-Challenge [6] and 10-shot HellaSwag [36]. Our evaluation metrics are benchmark-specific: for BoolQ, PIQA, and WinoGrande, we report accuracy; for TruthfulQA, we report mc2; and for ARC-C and HellaSwag, we report normalized accuracy.

Table 3 summarizes the results. It can be discerned that, PoSE-extended models exhibit only marginal performance degradation compared with Full-length fine-tuning and the original version, with the only

exception of the 128k model employing linear interpolation. For both NTK and YaRN, the extension from a 2k context window to a 16k and even a 128k context window resulted in minimal reductions in model performance, indicating their ability to effectively preserve the model's capabilities while extending the window size.

6 Conclusion

Context window extension aims at extending the context window of a pre-trained LLM to accommodate longer sequences. In this paper, we introduce **Po**sitional **S**kip-wis**E** (PoSE) training to efficiently extend the context window of Large Language Models. PoSE simulates long inputs by manipulating position indices, thereby requiring only the original context window for fine-tuning, successfully decoupling train length and test length. Experiments have shown that, compared with fine-tuning on the full length, PoSE greatly reduces memory and time overhead. Taking advantage of this, we have managed to extend LLaMA model to 128k with very limited training budget, observing only minimal performance degradation on standard benchmarks. We have also empirically verified that PoSE is compatible with all RoPE-based LLMs and position interpolation strategies.

References

[1] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

[4] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *ArXiv*, abs/2306.15595, 2023. URL https://api.semanticscholar.org/CorpusID:259262376.

[5] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

[6] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[7] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019.

[8] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. 2023.

[9] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

[10] Hugging Face. Open llm leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

[11] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[12] Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1382–1390, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.99. URL https://aclanthology.org/2022.findings-emnlp.99.

[13] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.112. URL https://aclanthology.org/2021.naacl-main.112.

[14] kaiokendev. Things i'm learning while training superhot. https://kaiokendev.github.io/til#extending-context-to-8k, 2023.

[15] Sakaguchi Keisuke, Le Bras Ronan, Bhagavatula Chandra, and Choi Yejin. Winogrande: An adversarial winograd schema challenge at scale. 2019.

[16] Woosuk Kwon*, Zhuohan Li*, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Yu, Joey Gonzalez, Hao Zhang, , and Ion Stoica. vllm: Easy, fast, and cheap llm serving with pagedattention. https://vllm.ai/, 2023.

[17] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, and Daniel Haziza. xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers, 2022.

[18] Mukai Li, Shansan Gong, Jiangtao Feng, Yiheng Xu, Jun Zhang, Zhiyong Wu, and Lingpeng Kong. In-context learning with many demonstration examples. *arXiv preprint arXiv:2302.04931*, 2023.

[19] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, 2022.

[20] Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers, 2023.

[21] Bowen Peng and Jeffrey Quesnelle. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation. https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have, 2023.

[22] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models, 2023.

[23] Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

[24] Shawn Presser. https://twitter.com/theshawwn/status/1320282149329784833, 2020.

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

[26] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.05507.

[27] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[28] Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. Randomized positional encodings boost length generalization of transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1889–1903, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.161. URL https://aclanthology.org/2023.acl-short.161.

[29] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[30] Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14590–14604, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.816. URL https://aclanthology.org/2023.acl-long.816.

[31] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[32] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[33] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[34] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *arXiv preprint arXiv:2306.07174*, 2023.

[35] Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

[36] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

[37] Azerbayev Zhangir, Ayers Edward, and Bartosz Piotrowski. Proof-pile. https://github.com/zhangir-azerbayev/proof-pile, 2022.

[38] Yucheng Zhou, Tao Shen, Xiubo Geng, Chongyang Tao, Guodong Long, Can Xu, and Daxin Jiang. Fine-grained distillation for long document retrieval. *arXiv preprint arXiv:2212.10423*, 2022.