EXTENDING CONTEXT WINDOW OF LARGE LANGUAGE MODELS VIA POSITION INTERPOLATION

Shouyuan Chen **Sherman Wong** **Liangjian Chen** **Yuandong Tian**
Meta Platforms Inc.
{chenshouyuan,shermanwong,clj,yuandong}@meta.com

ABSTRACT

We present Position Interpolation (PI) that extends the context window sizes of RoPE-based (Su et al., 2021) pretrained LLMs such as LLaMA (Touvron et al., 2023) models to up to 32768 with minimal fine-tuning (within 1000 steps), while demonstrating strong empirical results on various tasks that require long context, including passkey retrieval, language modeling, and long document summarization from LLaMA 7B to 65B. Meanwhile, the extended model by Position Interpolation preserve quality relatively well on tasks within its original context window. To achieve this goal, Position Interpolation linearly down-scales the input position indices to match the original context window size, rather than extrapolating 外推 beyond the trained context length which may lead to catastrophically 灾难性地 high attention scores that completely ruin the self-attention mechanism. Our theoretical study shows that the upper bound of interpolation is at least $\sim 600\times$ smaller than that of extrapolation, further demonstrating its stability. Models extended via Position Interpolation retain its original architecture and can reuse most pre-existing optimization and infrastructure.

1 INTRODUCTION

Large language models (LLMs) typically come with a pre-defined context window size. For example, inputs to LLaMA models (Touvron et al., 2023) must be fewer than 2048 tokens. This pre-set context window limit is frequently exceeded in applications such as conducting long conversations, summarizing long documents, or executing long-term planning. For these applications, LLMs with longer context windows are preferred. However, training an LLM from scratch with long context windows requires significant investments. This naturally leads to a question: Can we extend the context window of an existing pre-trained LLM?

One straightforward approach is to fine-tune an existing pre-trained Transformer with a longer context window. However, empirically, we found that models trained this way adapt to long context windows very slowly. After training for more than 10000 batches, the effective context window saw a minimal increase, moving from 2048 to 2560 (Table 4). This suggests that such method is inefficient for extending to substantially longer context windows.

While certain techniques such as ALiBi (Press et al., 2022) and LeX (Sun et al., 2022) enable length extrapolation of Transformers, i.e. train on short context windows and inference on longer ones, many existing pre-trained LLMs, including LLaMA (Touvron et al., 2023), use positional encodings that have weak extrapolation properties (e.g., RoPE (Su et al., 2021)). Therefore, the applicability of these techniques for extending the context window sizes of such LLMs remains limited.

In this work, we introduce Position Interpolation to enable context window extensions for certain existing pre-trained LLMs, including LLaMA. The key idea is, instead of extrapolation, we directly down-scale the position indices so that the maximum position index matches the previous context window limit in the pre-training stage. See Figure 1 for an illustration. In other words, to accommodate more input tokens, we interpolate the position encodings at neighboring integer positions, utilizing the fact that position encodings can be applied on non-integer positions, as opposed to extrapolating outside the trained positions, which may lead to catastrophic values. We verify our approach theoretically, by showing that the interpolated attention score has a much smaller upper

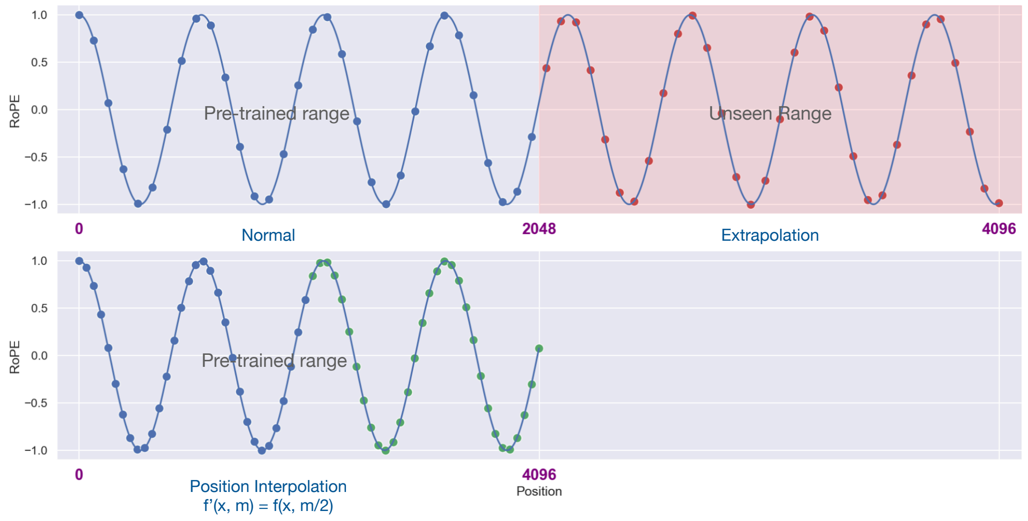

Figure 1: An illustration of our Position Interpolation method. Consider a Llama model pre-trained with a 2048 context window length. Upper left illustrates the normal usage of an LLM model: input position indices (blue dots) are within the pre-trained range. Upper right illustrates length extrapolation where models are required to operate unseen positions (red dots) up to 4096. Lower left illustrates Position Interpolation where we downscale the position indices (blue and green dots) themselves from [0, 4096] to [0, 2048] to force them to reside in the pretrained range.

bound ($\sim 600\times$ smaller in LLaMA 7B setting) than the extrapolated one, and is thus much more stable. Therefore, interpolated position encodings are easier for the model to adapt.

Empirically, we found that Position Interpolation is highly effective and efficient, requiring only a very short period of fine-tuning for the model to fully adapt to greatly extended context windows. We present experimental results for extending the context window to up to 32768 from the initial 2048 across 7B to 65B LLaMA models using Position Interpolation. Our results show that

1. Position Interpolation can easily enable very long context windows (e.g. 32768), requiring only fine-tuning for 1000 steps on the Pile (Gao et al., 2020) to achieve a good quality. The cost of fine-tuning is negligible compared to the pre-training costs. This confirms our hypothesis that it is relatively easy for the models to adapt to interpolated position encodings.
2. Position Interpolation generates strong models that can effectively make use of much extended context window. We show that models extended by Position Interpolation enjoy significant perplexity gains from greatly extended context windows for text modeling, and we show that the perplexity reduces graceful with the enlargement of context windows. 优雅地 We also applied Position Interpolation in a long text summarization task, and demonstrate competitive performances.
3. Position Interpolation preserves model quality relatively well for tasks within its original context window sizes. We present a variety of evaluation results for the extended LLaMA models on the original LLaMA benchmark. Compared with original LLaMA models, the extended LLaMA models saw a minor degradation on several standard benchmarks within a 2048 token limit.

Our results highlight the innate ability of Transformer models to "extrapolate to sequence lengths longer than the ones encountered during training" as hypothesized in the seminal work of Vaswani et al. (2017). We reaffirm this hypothesis and suggest that the previously known weakness of extrapolating to longer sequences for language modeling (Press et al., 2022) may be due to direct extrapolation of positional encodings and it can be largely mitigated by interpolating position encodings instead.

Concurrent work. Right before our release, we are informed with a concurrent blogpost (SuperHOT kaiokendev (2023)) that also interpolates positional encoding in RoPE to extend the context window from 2K to 8K. Recently, open source community picks it up in Reddit post [1] and Github Issues [2], which shows that fine-tuning with LoRA (Hu et al., 2021) also seems to work well. Our paper shows a full fine-tuning with up to 65B model work well with Position Interpolation, and we also give theoretical explanations why interpolation achieves much more stable results than extrapolation, by showing that the upper bound of interplated attention score is much lower than that of extrapolated ones.

2 METHOD

2.1 BACKGROUND: ROTARY POSITION EMBEDDING (ROPE)

Transformer models require explicit positional information to be injected, typically in the form of positional encodings, to represent the order of inputs. We consider Rotary Position Embedding (RoPE) (Su et al., 2021), which is the position encoding used in the LLaMA model (Touvron et al., 2023). Given a position index $m \in [0, c)$ and an embedding vector $\mathbf{x} := [x_0, x_1, \ldots, x_{d-1}]^\top$, where d is the dimension of the attention head, RoPE defines a vector-valued complex function $\mathbf{f}(\mathbf{x}, m)$ as follows

$$\mathbf{f}(\mathbf{x}, m) = [(x_0 + \mathrm{i}x_1)e^{\mathrm{i}m\theta_0}, (x_2 + \mathrm{i}x_3)e^{\mathrm{i}m\theta_1}, \ldots, (x_{d-2} + \mathrm{i}x_{d-1})e^{\mathrm{i}m\theta_{d/2-1}}]^\top \tag{1}$$

where $\mathrm{i} := \sqrt{-1}$ is the imaginary unit and $\theta_j = 10000^{-2j/d}$. Using RoPE, the self-attention score

$$\begin{aligned} a(m, n) &= \mathrm{Re}\langle \mathbf{f}(\mathbf{q}, m), \mathbf{f}(\mathbf{k}, n)\rangle \\ &= \mathrm{Re}\left[\sum_{j=0}^{d/2-1} (q_{2j} + \mathrm{i}q_{2j+1})(k_{2j} - \mathrm{i}k_{2j+1})e^{\mathrm{i}(m-n)\theta_j}\right] \\ &= \sum_{j=0}^{d/2-1} (q_{2j}k_{2j} + q_{2j+1}k_{2j+1})\cos((m-n)\theta_j) + (q_{2j}k_{2j+1} - q_{2j+1}k_{2j})\sin((m-n)\theta_j) \\ &=: a(m-n) \end{aligned} \tag{2}$$

is only dependent on relative position $m - n$ through trigonometric functions. Here $\mathbf{q}$ and $\mathbf{k}$ are the query and key vector for a specific attention head. At each layer, RoPE is applied on both query and key embeddings for computing attention scores.

2.2 DIRECT EXTRAPOLATION

While the attention score in RoPE only depends on the relative positions, which is what we want, its extrapolation performance is not great . In particular, when directly extending to larger context windows unseen in the training, the perplexity may shoot up to very high numbers (i.e., $\geq 10^3$), comparable to untrained models. RoPE外推性很差

Ideally, we want to see the model trained on a context window of size $L = 2048$ to still work reasonably well on longer context window, but may not have the capability to leverage information that appears beyond L. For example, to answer a question located at 3000, the model trained on maximal window size of $L = 2048$ cannot leverage evidences provided at location 0, but still can leverage the evidences provided at location 2900. In contrast, in reality we see catastrophic behaviors, i.e., question at location 3000 cannot be answered correctly, even if the evidences are located at location 2900.

What is the reason behind? How could this happen if the attention score a_{m-n} decays as the relative distance $|m - n|$ increases, according to Section 3.4.3 of (Su et al., 2021), and content from very

[1] https://www.reddit.com/r/LocalLLaMA/comments/14fgjqj/a_simple_way_to_extending_context_to_8k/

[2] https://github.com/ggerganov/llama.cpp/discussions/1965

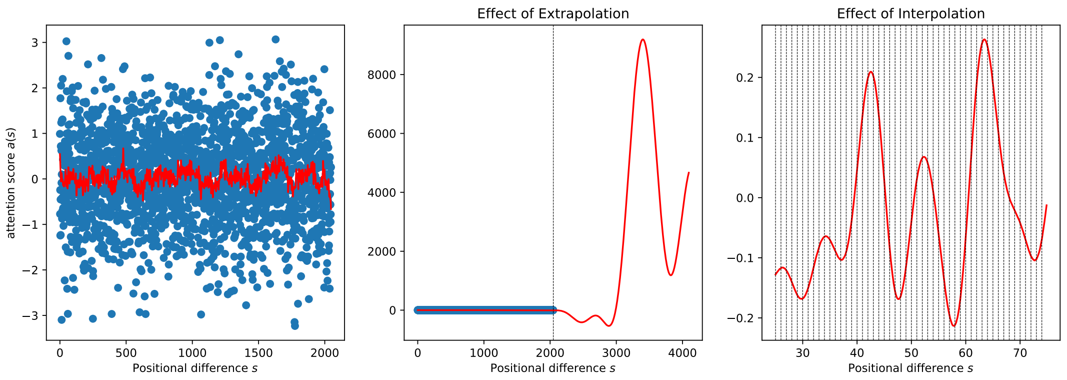

Figure 2: Extrapolation versus interpolation. **Left:** a fitted attention score function (in red) in the form of Eqn. 3 with $d = d_{\mathrm{model}}/n_{\mathrm{head}} = 4096/32 = 128$ (setting of LLaMA 7B). Dots are random input points to be fitted and red curve is the fitted score function via least square, which is approximately within $[-1, 1]$. **Middle:** While the fitted function seems to be well bounded in $[0, L]$, where $L = 2048$, out of this region it may goes beyond 8000, causing catastrophic issues in attention computation. Note that here we do not cherry pick at all: almost every learned curve from a set of randomly generated input points within $[0, L]$ has the extrapolation issue. **Right:** On the other hand, interpolation is much more stable. Curves in between vertical dotted lines (i.e., integer positional difference) are smooth and well-behaved. Please check Appendix C.1 for the source code used to generate the figure.

far distances should not matter that much? It turns out that the upper bound derived in Section 3.4.3 of (Su et al., 2021) may be too loose: while it indeed decays with respect to $|m - n|$, the bound can still be quite large (i.e., the bound can be critically depends on the magnitude of v_j) and thus vacuous. In fact, if we treat all trigonometric functions as basis functions (i.e, $\phi_j(s) := e^{\mathrm{i}s\theta_j}$), and think about Eqn. 2 as basis expansion as the following:

$$a(s) = \mathrm{Re}\left[\sum_{j=0}^{d/2-1} h_j e^{\mathrm{i}s\theta_j}\right] \tag{3}$$

where s is the positional span between a query and a key and $h_j := (q_{2j} + \mathrm{i}q_{2j+1})(k_{2j} - \mathrm{i}k_{2j+1})$ are complex coefficients depending on $\mathbf{q}$ and $\mathbf{k}$ (here the definition of h_j is exactly the same as the definition of h_j in Sec 3.4.3 in RoPE (Su et al., 2021)). Now the the issue becomes clear: as shown in Fig. 2, a_s can be small in magnitude in the range of $[0, 2048]$, but gives huge values out of the region. The underlying reason is that the trigonometric family $\{\phi_j\}$ (with sufficiently large d) is a universal approximator and can fit any arbitrary functions. Therefore, for a_s, there always exist coefficients $\{h_j\}$ (i.e. key and query) that corresponds to small function values in [0, 2048] but much larger in regions beyond.

2.3 Proposed approach: Position Interpolation (PI)

In Fig. 2, thanks to the smoothness of bases functions ϕ_j *interpolation* is much more stable and will not lead to wild values. Therefore, instead of extrapolate the attention score in Eqn. 3 to $s > L$, how about we define an attention score $\tilde{a}(s) = a(Ls/L')$ where L' is the longer context window? Formally, we replace RoPE $\mathbf{f}$ by $\mathbf{f}'$ defined as follows

$$\mathbf{f}'(\mathbf{x}, m) = \mathbf{f}\left(\mathbf{x}, \frac{mL}{L'}\right). \tag{4}$$

We call this transformation on the position encoding **Position Interpolation**. In this step, we reduce position indices from $[0, L')$ to $[0, L)$ to match the original range of indices before computing RoPE. Consequently, as inputs to RoPE, the maximum relative distance between any two tokens has been reduced from L' to L. Since we align the ranges of position indices and relative distances before and after extension, we mitigate the effect on attention score computation due to context window extensions, which can allow the model easier to adapt. To further demonstrate this is the case, in the following theorem, we show that the interpolated attention score is well-behaved:

Theorem 2.1 (Interpolation bound). *For attention score $a(s) = \mathrm{Re}\left[\sum_{j=0}^{d/2-1} h_j e^{\mathrm{i}s\theta_j}\right]$, where $\theta_j = c^{-2j/d}$, its interpolation value $a(s)$ for $s \in [s_1, s_2]$ is bounded as follows:*

$$|a(s) - a_{\mathrm{linear}}(s)| \le d \left(\max_j |h_j|\right) \frac{(s - s_1)(s_2 - s)}{8 \ln c} \tag{5}$$

where $a_{\mathrm{linear}}(s)$ is the linear interpolation of two grid point $a(s_1)$ and $a(s_2)$ that are known to behave well, enforced by LLM pre-training:

$$a_{\mathrm{linear}}(s) := (1 - \lambda(s))a(s_1) + \lambda(s)a(s_2), \qquad \lambda(s) := \frac{s - s_1}{s_2 - s_1} \tag{6}$$

Please check Appendix A for the proof. Intuitively, in LLM pre-training, we know that the attention score $a(s)$ behaves well on integer grid s_1 and s_2. Therefore, for any interpolation $s \in [s_1, s_2]$, we have $(s - s_1)(s_2 - s) \le 1/4$. Note that $c = 10000$, the bound becomes:

$$|a(s) - a_{\mathrm{linear}}(s)| \le \frac{d}{32 \ln c} \max_j |h_j| \approx \frac{d \max_j |h_j|}{294.73} \tag{7}$$

In comparison, Sec. 3.4.3 in RoPE (Su et al., 2021) yields an extrapolation bound (i.e., it works for all positional distance s):

$$|a(s)| \le \left(\max_j |h_j - h_{j+1}|\right) \sum_{k=0}^{d/2-1} |A_{k+1}(s)| \le 2 \left(\max_j |h_j|\right) \sum_{k=0}^{d/2-1} |A_{k+1}(s)|, \tag{8}$$

where $A_k(s) := \sum_{j=0}^{k-1} e^{\mathrm{i}s\theta_j}$. While there is no close form for $B(s) := \sum_{k=0}^{d/2-1} |A_{k+1}(s)|$, numerically it is at least larger than d, and for many positional difference s, $B(s)$ is much larger than d (check Appendix B for the plot). Therefore, the interpolation bound is at least $2 \cdot 294.73 \sim 600\times$ smaller than the extrapolation bound, and thus the interpolated attention score is much more stable than extrapolated one.

Notably, our method of rescaling of position indices does not introduce extra weight, or modify the model architecture in any way. This makes it attractive in practical applications, since most infrastructure and optimization for the original model can be reused after the extension.

Fine-tuning. We can further fine-tune the interpolated model using the next token prediction task with interpolated position encodings on the extended context window size using a pre-training corpus such as the Pile (Gao et al., 2020). In the next section, we show that our fine-tuning process only needs tens to hundreds thousands of examples. We also find that the result of the fine-tuning is not sensitive to the choice of examples. The reason may be that the model is only adapting to the new context window during the fine-tuning phase, starting from a good initialization, as opposed to acquiring new knowledge.

Other ways to reduce interpolation/extrapolation bound. From the expression of the interpolation (Eqn. 5) and extrapolation bound (Eqn. 8), a common term is $\max_j |h_j|$, which is the maximal magnitude of query/key products. If we enforce a regularization on $|h_j|$ during LLM training, it is possible that the catastrophic extrapolation error can be mitigated or even resolved. In fact, if we apply ridge regression with proper regularization to fit a curve in Fig. 2, the magnitude of extrapolated $a(s)$ when $s > L$ can be comparable to that within $[0, L]$. To our knowledge, we are not aware of existing LLM pre-training techniques that leverage this regularization and will leave it for future work.

3 Experiments

We show Position Interpolation can effectively extend context window up to 32 times of the original size, and such extension can be done with only several hundreds of training steps. We show the resulting models are strong LLMs with fully effective long context windows. We demonstrate its performance in a number of tasks including language modeling, passkey retrieval, and long document summarization. We also present benchmark results of the extended models on the original LLaMA evaluation benchmarks.

3.1 Setup

Model Variants. We extended the pre-trained 7B, 13B, 33B and 65B LLaMA models (Touvron et al., 2023) to various context window of sizes up to 32768, using either direct fine-tuning or Position Interpoloation method. Except for rescaling the position indices for models extended with Position Interpolation, we did not modify LLaMA model architectures (Touvron et al., 2023) in any ways.

Training Procedure. We fine-tune all model variants using the next token prediction objective. We use AdamW (Loshchilov & Hutter, 2019) with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. We use a linear learning rate warmup of 20 steps starting from 10% of the maximum learning rate. For 7B and 13B models, we set the learning rate to 2×10^{-5} and for 33B and 65B models we set the learning rate to 10^{-5}. We set the weight decay to zero. For extending 7B, 13B and 33B models to the 8192 context window size, we use 32 A100 GPUs and 64 global batch size. For all other cases we use 128 A100 GPUs and 128 global batch size. We note that the main need of using more GPUs is memory limitation during fine-tuning, and it is possible to use fewer GPUs in certain cases. We train all models using PyTorch (Paszke et al., 2019) with Fully Sharded Data Parallel (Zhao et al., 2023) and Flash Attention (Dao et al., 2022).

If not specified otherwise, for the Position Interpolation method, we fine-tune the models for 1000 steps. For the direct fine-tuning method, we use 10000 steps. We primarily fine-tune using the Pile training dataset (Gao et al., 2020). In Section 3.4 we also compared fine-tuning performance on the RedPajama dataset (Computer, 2023).

3.2 Long Sequence Language Modeling

We evaluate the long sequence language modeling performance of our extended models and baselines on two datasets: book corpus (PG-19) (Rae et al., 2020) and cleaned Arxiv Math proof-pile dataset (Azerbayev et al., 2022).

We use the test splits of PG19 (Rae et al., 2020) and proof-pile (Azerbayev et al., 2022). For PG19, we use the whole test split consisting of 100 documents. For the proof-pile dataset, we use a random subsample of 128 documents with at least 32768 SentencePiece (Kudo & Richardson, 2018) tokens and truncate to the first 32768 tokens for each test document. We evaluate perplexity at various context window size by using a sliding window approach following Press et al. (2022) with stride $S = 256$.

In Table 1 and Table 2, we report the perplexity results for our models and baselines on the datasets. From the results, we found that models extended with our method enjoy a significantly improved perplexity from longer context window sizes. By increasing the context window size from 2048 to 16384, we observed -0.28 and -0.5 reductions of perplexity for extending LLaMA 7B models on both datasets, -0.27 and -0.48 reductions for extending LLaMA 13B models, and -0.14 and -0.42 reductions for extending LLaMA 33B models. For LLaMA 65B models, we observed -0.12 and -0.3 reductions of perplexity by extending to the 8192 context window size.

In general, we observed a consistent trend of our models achieving better perplexity with longer context windows. This indicates our models can effectively make use of the longer context windows to better predict next tokens in language modeling tasks. Moreover, we found this trend extends to 32768 window size without diminishing on the PG19 dataset for LLaMA 7B and 13B models. This indicates that our method may enable extension to even longer context windows.

In contrast, we observed that models extended via the direct fine-tuning method has shown regression (up to +0.48) or minor improvement (up to -0.12) on the perplexity at longer context windows. This indicates that models extended this way have limited capability of making use of context windows longer than their pre-trained settings.

We saw a minor degradation of the perplexity on the original context window of 2048 for our extended models in some cases. For example, on the Proof-pile dataset, we saw a degradation ranging from 0.01 to 0.05 across all models with extended with Position Interpolation. A small degradation of performance within original evaluation context window is expected since Position Interpolation forces position encodings in original context window to reside in a much narrower region, which may negatively affect the language model's performance. We present more benchmark results on the original context window size in Section 3.4.

In Table 3 we report the relationship between perplexity and the number of fine-tuning steps for LLaMA 7B model extending to 8192 and 16384 context window sizes using Position Interpolation evaluated on the PG19 dataset. We can see without fine-tuning (at step 0) the model can exhibit certain language modeling capability, as indicated by < 20 perplexity for extending to 8192 context window (in contrast, the direct extrapolation method leads to $> 10^3$ perplexity). With fine-tuning, we observed that the perplexity improves quickly. At 200 steps the models surpassed the original model's perplexity on 2048 context window size, indicating the models gaining ability of effectively using sequences longer than the pre-training settings for language modeling. At 1000 steps, we can see the models have improved steadily and achieve a significantly better perplexity.

	Model		Evaluation Context Window Size				
Size	Context Window	Method	2048	4096	8192	16384	32768
7B	2048	None	7.20	$> 10^3$	$> 10^3$	$> 10^3$	$> 10^3$
7B	8192	FT	7.21	7.34	7.69	-	-
7B	8192	PI	7.13	6.96	6.95	-	-
7B	16384	PI	7.11	6.93	6.82	6.83	-
7B	32768	PI	7.23	7.04	6.91	6.80	6.77
13B	2048	None	6.59	-	-	-	-
13B	8192	FT	6.56	6.57	6.69	-	-
13B	8192	PI	6.55	6.42	6.42	-	-
13B	16384	PI	6.56	6.42	6.31	6.32	-
13B	32768	PI	6.54	6.40	6.28	6.18	6.09
33B	2048	None	5.82	-	-	-	-
33B	8192	FT	5.88	5.99	6.21	-	-
33B	8192	PI	5.82	5.69	5.71	-	-
33B	16384	PI	5.87	5.74	5.67	5.68	-
65B	2048	None	5.49	-	-	-	-
65B	8192	PI	5.42	5.32	5.37	-	-

Table 1: Evaluation perplexity on PG19 dataset (Rae et al., 2020). FT: Direct Fine-tuning. PI: Position Interpolation. Model fine-tuned with PI shows progressively lower perplexity with longer context window, showing that PI can leverage long context well, while the perplexity of FT increases over longer window. Note that overall the perplexity is higher compared to Table 2 since PG19 has very different writing styles.

3.3 Measuring Effective Context Window Size through Passkey Retrieval

We study the effective context window size, i.e. the maximum distance of a token can *effectively* attend to during inference, of our models after extension. To measure this, we follow a synthetic evaluation task of passkey retrieval proposed by Mohtashami & Jaggi (2023). In this task, the models are asked to recover a random passkey hidden in a long document. See Figure 3 for the format of the document.

Given a language model, we estimate the upper and lower bounds of effective context windows as follows. Suppose the random passkey is k tokens away from the end of the input. When a model persistently fails to retrieve the correct passkey value across several independent attempts, it suggests that the effective context window size of the model is less than k. Conversely, if a model consistently succeeds in retrieving the correct passkey value, we deduce that the effective context window size of the model is at least k.

We evaluate the 7B and 33B LLaMA model variants that are extended via Position Interpolation or direct fine-tuning. For each model, we use 32 different k uniformly spaced in the targeted context window L' and run the above tests for 10 times for each k, where each time a random passkey of 5 random digits is used. In Table 4, we report $k_{\max}$ as a function of the number of fine-tuning steps,

	Model		Evaluation Context Window Size				
Size	Context Window	Method	2048	4096	8192	16384	32768
7B	2048	None	2.77	-	-	-	-
7B	8192	FT	2.85	2.74	2.73	-	-
7B	8192	PI	2.79	2.57	2.39	-	-
7B	16384	PI	2.79	2.57	2.37	2.25	-
7B	32768	PI	2.82	2.59	2.39	2.24	2.48
13B	2048	None	2.66	-	-	-	-
13B	8192	FT	2.71	2.56	2.50	-	-
13B	8192	PI	2.67	2.47	2.30	-	-
13B	16384	PI	2.68	2.47	2.29	2.18	-
13B	32768	PI	2.68	2.46	2.28	2.15	2.35
33B	2048	None	2.49	-	-	-	-
33B	8192	FT	2.56	2.48	2.47	-	-
33B	8192	PI	2.50	2.32	2.18	-	-
33B	16384	PI	2.53	2.34	2.18	2.07	-
65B	2048	None	2.42	-	-	-	-
65B	8192	PI	2.43	2.26	2.12	-	-

Table 2: Evaluation perplexity on Arxiv Math Proof-pile dataset (Azerbayev et al., 2022). FT: Direct Fine-tuning. PI: Position Interpolation.

	Model	Number of fine-tuning steps					
Size	Context Window	0	200	400	600	800	1000
7B	8192	16.10	7.12	7.10	7.02	6.99	6.95
7B	16384	112.13	7.05	6.93	6.88	6.84	6.83

Table 3: Evaluation perplexity on PG19 dataset (Rae et al., 2020) with respect to the number of fine-tuning steps using Position Interpolation.

where $k_{\max}$ is defined as the maximum k such that, for all $k' \leq k$, the model has a success rate of at least 20% on k'.

We can see that models extended via Position Interpolation all successfully attain their desired extension objectives in terms of effective context window sizes, indicating by the effective context window size reaching maximum $k_{\max} = L'$, after merely fine-tuning for 200 steps, consistently across both 7B and 33B model sizes and up to 32768 context windows. In contrast, LLaMA models that are extended via direct fine-tuning only saw a minimal increase of the effective context window size $k_{\max}$ from 2048 to 2560, even after fine-tuning for more than 10000 steps, with no clear indication of an acceleration in the increase of window size.

	Model		Fine-tuning steps					
Size	Context Window	Method	200	400	600	800	1000	10000
7B	8192	FT	1792	2048	2048	2048	2304	2560
33B	8192	FT	1792	2048	1792	2048	2304	-
7B	8192	PI	8192	8192	8192	8192	8192	-
7B	16384	PI	16384	16384	16384	16384	16384	-
7B	32768	PI	32768	32768	18432	32768	32768	-
33B	8192	PI	8192	8192	8192	8192	8192	-
33B	16384	PI	16384	16384	16384	16384	16384	-

Table 4: Effective context window sizes after fine-tuning. FT: Direct fine-tuning. PI: Position Interpolation.

```
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 12345.  Remember it.  12345 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
```

Figure 3: Prompt format for passkey retrieval. We use the exact same prompt as proposed by Mohtashami & Jaggi (2023). Here the passkey 12345 is replaced with a random 5-digit numbers during test.

3.4 Benchmarks on Original Context Window Size

We evaluate the models extended by Position Interpolation on several standard benchmark tasks within the original context window size of 2048. The evaluation results are listed in Table 5. From the results, we saw that models extended to 8192 produce comparable results on the original benchmark which is designed for a much smaller context window, with a degradation of up to 2% on the benchmark tasks, for both 7B and 33B model sizes. Models extended to longer context windows regressed more on the benchmarks, but still in reasonable ranges for most tasks. We also note that the choice of fine-tuning datasets does not seem to lead significant difference in the benchmark performances, which may be due to the limited number of fine-tuning steps used in our method. The regression on benchmark tasks is consistent with our observation on perplexity regression in Section 3.2.

3.5 Long Document Summarization

In this task, we evaluate our models' performance on the long document summarization task. In particular, we consider the GovReport (Huang et al., 2021) dataset, which contains 17457 documents for training and 972 documents for evaluation. Each document comes with a human generated summary. We truncate all input documents to their first 15000 tokens.

We fine-tune the LLaMA models extended with Position Interpolation with a context window of 16384. Note the rescaling of position indices are still required during this fine-tuning step. We first

Model Size	Context Window	Fine-tune on	BoolQ	PIQA	Race-M	Race-H	WinoGrande
7B	2048	None	76.1	78.9	55.7	42.2	69.6
7B	8192	Pile	73.2	78.2	53.8	41.7	69.0
7B	16384	Pile	69.8	77.6	53.3	40.9	67.8
7B	32768	Pile	64.7	77.2	50.1	39.6	66.9
7B	8192	RedPajama	75.5	77.4	54.5	41.5	68.1
33B	2048	None	81.6	80.2	61.1	45.9	76.2
33B	8192	Pile	80.2	80.7	60.2	45.7	75.9

Table 5: Zero-shot performance on a subset of LLaMA Benchmarks. Models extended by Position Interpolation comparable performance as the original models, except for BoolQ dataset that may require models to pay close attention to word ordering in a short reference paragraph.

Model		Evaluation Score		
Model	Context Window	ROUGE-1	ROUGE-2	ROUGE-L
CoLT5 Base (Ainslie et al., 2023)	16K	58.7	29.6	31.4
CoLT5 XL (Ainslie et al., 2023)	16K	61.3	32.2	33.8
LLaMA-7B Extended	16K	60.0	28.0	29.5

Table 6: ROUGE Score on GovReport Dataset.

format the raw document using the prompt template in Figure 4, and then concatenate the prompt with the ground-truth summary (truncate to 1000 tokens) associated with each document. We fine-tune the model using the next token prediction task with the above setup for 10 epochs. The losses from the input prompt proportion of training examples are excluded during our fine-tuning.

We use a generation temperature of 0.5 and top$_p = 0.95$ as our inference parameter to generate a summarization of each document in the test set. The final output is truncated at 1000 tokens. We used the ROUGE-1/ROUGE-2/ROUGE-L scores (Lin, 2004) as the evaluation metrics to evaluate the models' outputs vs the ground-truth summaries.

In Table 6 we report our evaluation results. We have also included results from two baselines in existing SCROLLS Leaderboard (Shaham et al., 2022; Ainslie et al., 2023). In general, we have obtained competitive R1 score among other models with minimal tuning of hyper-parameters. This result suggests our models with 16384 context window can effectively handle the long document summarization task.

```
Read the following article and then summarize it.
# ....  Document goes here
Now summarize the above article.
Summary:
```

Figure 4: Input format for long doc summarization.

4 Related Work

Retrieval-augmented LLM. One line of work extends LLMs by augmenting it with retrieval modules which fetch related documents and include the retrieval results into the input context of an LLM (Karpukhin et al., 2020; Guu et al., 2020; Izacard et al., 2022; Jiang et al., 2022; Khattab et al., 2021; Santhanam et al., 2022). Our work is complementary to these works as our extended context window allows more documents being included in the input. In addition, with an unmodified attention mechanism and model architecture, our method may be more versatile as it can natively handle tasks beyond retrieval oriented ones, such as long document summarization, few-shots learning, etc.

Recurrent Transformers and Memory Transformers. Several works add memory capabilities to Transformers through recurrence, which increase the models' capability of handling very long sequences (Bulatov et al., 2022; Wu et al., 2020; Dai et al., 2019; Wu et al., 2022; Martins et al., 2021; Mu et al., 2023). One limitation of these works is that they only allow attending to a lossy compressed version of past inputs. Mu et al. (2023) suggested that this may prevent models from remembering specific details in the past inputs. In contrast, our work allows attending to all previous tokens, preserving all details without compression, albeit with higher inference costs. Mohtashami & Jaggi (2023) proposed landmark attention which allows full random access to any chunk of the input through introducing landmark tokens. Our work allows full access of the entire input through unmodified attention, which may be useful for tasks such as summarization.

Approximated Multi-head Attention. There is a large body of research that focuses on decreasing the memory and computational complexity of the multi-head attention (MHA) mechanism through approximation or sparsification (Child et al., 2019; Zaheer et al., 2020; Beltagy et al., 2020; Wang et al., 2020; Choromanski et al., 2021; Kitaev et al., 2020; Ren et al., 2021). Although not the focus of this work, as these methods are not used in LLaMA (Touvron et al., 2023), we note that our method is compatible with most of them since our changes are restricted to position encodings, and not attention mechanisms.

Length Extrapolation. A recent line of research aims to train Transformers models on short sequences and inference on longer (Press et al., 2022; Sun et al., 2022; Haviv et al., 2022). However, these methods have not been applied in some of the largest language models such as LLaMA (Touvron et al., 2023), or OPT (Zhang et al., 2022). This has prevented them from enabling length extrapolation of many pre-existing pre-trained language models. Our work focuses on extending existing LLMs, which can save substantial pre-training costs. In addition, our method preserves the quality of the original models, even for small context window tasks, since it does not deviate far from existing definitions of position encoding or attention mechanisms.

Interpolation. The most related technique to ours is proposed by Dosovitskiy et al. (2021) in their work on Vision Transformers, where the authors proposed to linearly interpolate learnt position embeddings to support higher resolution, which translates to an increased number of input embeddings, in the fine-tuning stage. The interpolated position embedding weights are used as initialization in the fine-tuning process for the newly added positions. Our work differs from their work in several ways (1) Instead of interpolating position embeddings, our method interpolates position indices, which is more suitable for RoPE like position encodings and may require less training since no trainable parameters are added. (2) We report successful results of extending the context window to 32 times while Dosovitskiy et al. (2021) explored up to 4 times. Our results extend theirs in exploring the upper limit of context window extension via interpolation. (3) We evaluated and confirmed the effectiveness of Position Interpolation for extending context windows for language models.

We believe our results, in conjunction with (Dosovitskiy et al., 2021), provide empirical evidence on Transformer's remarkable ability of handling significantly longer sequences beyond training. Further, we conjecture that a method similar to theirs is directly applicable in LLMs with learnable position embeddings such as OPT (Zhang et al., 2022) and we plan to investigate this in the future.

5 Conclusions

Position Interpolation can effectively extend LLaMA models' context window to be significantly larger, using minimal fine-tuning. The extended models are fully capable to perform a variety of tasks on the extended context windows, and preserve its original ability relatively well for tasks within the original extended models, making them good choices of generic language models for both long and short input prompts. Further, models extended by Position Interpolation can reuse most pre-existing infrastructure and optimization, making this method attractive in many practical applications. We believe that Position Interpolation is a general method that could be apply to other types of position encodings, which can allow extension for more types of LLMs, and we plan to investigate in such directions in the near future.

Acknowledgements

We thank Mike Lewis for his input on evaluation.

References

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. Colt5: Faster long-range transformers with conditional computation, 2023.

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

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. 2020.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. 2022.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. 2019.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, May 2021.

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

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformerxl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285.

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.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

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.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. 2020.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. Transformer language models without positional encodings still learn positional information. 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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*, pp. 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`.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. 2022.

Zhengbao Jiang, Luyu Gao, Jun Araki, Haibo Ding, Zhiruo Wang, Jamie Callan, and Graham Neubig. Retrieval as attention: End-to-end learning of retrieval and reading within a single transformer. 2022.

kaiokendev. Things iḿ learning while training superhot. `https://kaiokendev.github.io/til#extending-context-to-8k`, 2023.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550.

Omar Khattab, Christopher Potts, and Matei Zaharia. Relevance-guided supervision for openqa with colbert. *Transactions of the Association for Computational Linguistics*, 9:929–944, 2021. doi: 10.1162/tacl_a_00405.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net, April 2020.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL `https://aclanthology.org/D18-2012`.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-1013`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. ∞-former: Infinite memory transformer. 2021.

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

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=R8sQPpGCv0`.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SylKikSYDH`.

Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pp. 22470–22482. Curran Associates, Inc., 2021.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3715–3734, Seattle, United States, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.272.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 12007–12021, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL `https://aclanthology.org/2022.emnlp-main.823`.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021.

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer, 2022.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. 2020.

Qingyang Wu, Zhenzhong Lan, Kun Qian, Jing Gu, Alborz Geramifard, and Zhou Yu. Memformer: A memory-augmented transformer for sequence modeling. 2020.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *The Tenth International Conference on Learning Representations, ICLR 2022*. OpenReview.net, April 2022.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*. Curran Associates, Inc., 2020.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023.

Appendix

A Proof

Theorem 2.1 (Interpolation bound)**.** *For attention score $a(s) = \mathrm{Re}\left[\sum_{j=0}^{d/2-1} h_j e^{\mathrm{i}s\theta_j}\right]$, where $\theta_j = c^{-2j/d}$, its interpolation value $a(s)$ for $s \in [s_1, s_2]$ is bounded as follows:*

$$|a(s) - a_{\mathrm{linear}}(s)| \le d\left(\max_j |h_j|\right)\frac{(s-s_1)(s_2-s)}{8\ln c} \tag{5}$$

where $a_{\mathrm{linear}}(s)$ is the linear interpolation of two grid point $a(s_1)$ and $a(s_2)$ that are known to behave well, enforced by LLM pre-training:

$$a_{\mathrm{linear}}(s) := (1-\lambda(s))a(s_1) + \lambda(s)a(s_2), \qquad \lambda(s) := \frac{s-s_1}{s_2-s_1} \tag{6}$$

Proof. Using Taylor expansion, we have:

$$a(s_1) \quad = \quad a(s) + a'(s)(s-s_1) + \frac{1}{2}a''(\xi_1)(s-s_1)^2 \tag{9}$$

$$a(s_2) \quad = \quad a(s) + a'(s)(s-s_2) + \frac{1}{2}a''(\xi_2)(s-s_2)^2 \tag{10}$$

where $\xi_1 \in [s_1, s]$ and $\xi_2 \in [s, s_2]$. Multiplying Eqn. 9 with $s - s_2$ and Eqn. 10 with $s - s_1$ and subtract, we get:

$$a(s) - a_{\mathrm{linear}}(s) = R(s) := -\frac{(s-s_1)(s-s_2)}{2(s_1-s_2)}\left[a''(\xi_1)(s-s_1) - a''(\xi_2)(s-s_2)\right] \tag{11}$$

Now we bound the second order derivative $a''(s)$. Note that for any complex number x, $|\mathrm{Re}(x)| \le |x|$ so we have:

$$|a''(s)| \quad \le \quad \sum_{j=0}^{d/2-1} |h_j||\phi_j''(s)| \le \sum_{j=0}^{d/2-1} |h_j|\theta_j^2 \tag{12}$$

$$\le \quad \left(\max_j |h_j|\right)\sum_{j=0}^{d/2-1} c^{-4j/d} = \left(\max_j |h_j|\right)\frac{1}{1-c^{-4/d}} \tag{13}$$

Note that when $x < 0$ and $c > 1$, $c^x \le 1 + x\ln c$, therefore $c^{-4/d} \le 1 - 4/d\ln c$ and we have:

$$\frac{1}{1-c^{-4/d}} \le \frac{1}{4/d\ln c} = \frac{d}{4\ln c} \tag{14}$$

So

$$|a''(s)| \le \left(\max_j |h_j|\right)\frac{d}{4\ln c} =: M \tag{15}$$

Let the above bound to be M, we have:

$$|R(s)| \le \frac{(s-s_1)(s_2-s)}{2(s_2-s_1)}\left[M(s-s_1) + M(s_2-s)\right] = \frac{M}{2}(s-s_1)(s_2-s) \tag{16}$$

As a result:

$$|a(s) - a_{\mathrm{linear}}(s)| = |R(s)| \le d\left(\max_j |h_j|\right)\frac{(s-s_1)(s_2-s)}{8\ln c} \tag{17}$$

□

B Visualization of quantities in extrapolation bound

As shown in Eqn. 8, the extrapolation bound contains the term $B(s) := \sum_{k=0}^{d/2-1} |A_{k+1}(s)|$ where $A_k(s) := \sum_{j=0}^{k-1} e^{\mathrm{i}s\theta_j}$. Here we check how large the bound is. We use $\theta_j = c^{-2j/d}$ with $c = 10000$ and $d = 4096/32 = 128$ (LLaMA-7B setting), and Fig. 5 shows that $B(s)/d$ almost always larger than 1 and in many places it is much larger than 1.

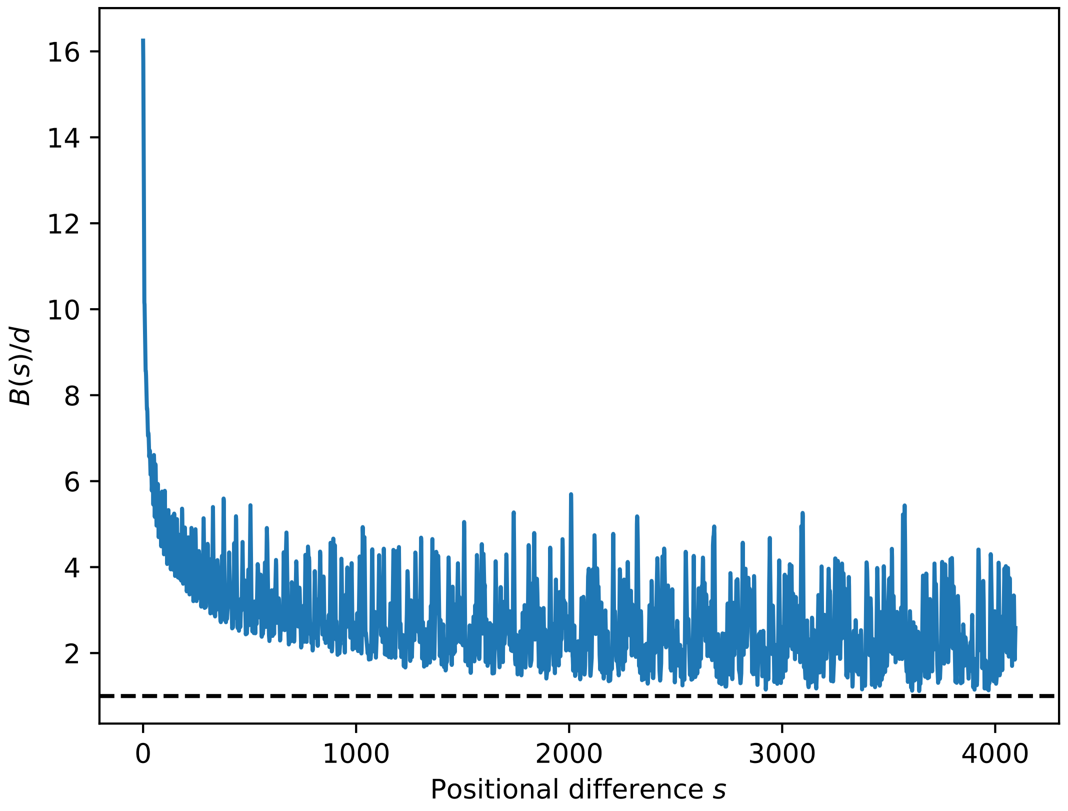

Figure 5: The bound $B(s)/d$ decays with s. While the bounds goes down with large positional difference s, numerically $B(s)/d \geq 1$ and at many s much larger than 1 (the dotted horizontal line). Please check Appendix C.2 for the source code used to draw the figure.

C CODE

C.1 CODE FOR FIG. 2

```
# build basis function
d = 4096 // 32
theta = 10000
# Frequency computation,
freqs = 1.0 / (theta ** (torch.arange(0, d, 2)[: (d // 2)].float() / d))

# construct basis function
L = 2048

x = torch.zeros(L)
x[:L] = torch.arange(0, L)

# basis functions
xfreq = torch.outer(x, freqs)

y = torch.randn(x.shape[0])

# do linear regression
X = torch.cat([xfreq.sin(), xfreq.cos()], dim=1)

eps = 0.000
coeffs = torch.linalg.solve(X.t() @ X + torch.eye(X.shape[1]) * eps, X.t() @ y)

x2 = torch.arange(0, 2*L)
xfreq2 = torch.outer(x2, freqs)
X2 = torch.cat([xfreq2.sin(), xfreq2.cos()], dim=1)

y2 = X2 @ coeffs

x3 = torch.arange(25, 75, 0.125)
xfreq3 = torch.outer(x3, freqs)
X3 = torch.cat([xfreq3.sin(), xfreq3.cos()], dim=1)

y3 = X3 @ coeffs

plt.figure(figsize=(16,5))

plt.subplot(1, 3, 1)
plt.plot(x2[:L], y2[:L], "r")
plt.scatter(x, y)
plt.ylabel("attention score $a(s)$")
plt.xlabel("Positional difference $s$")

plt.subplot(1, 3, 2)

plt.plot(x2, y2, "r")
plt.scatter(x, y)
plt.axvline(L, color="k", linestyle="--", linewidth=0.5)

plt.title("Effect of Extrapolation")
plt.xlabel("Positional difference $s$")


plt.subplot(1, 3, 3)
plt.plot(x3, y3, "r")
for i in range(25,75):
    plt.axvline(i, color="k", linestyle="--", linewidth=0.5)
plt.title("Effect of Interpolation")
plt.xlabel("Positional difference $s$")
plt.show()
```

C.2 Code for Fig. 5

```
L = 2048
x = torch.arange(0, 2*L)
d = 4096 // 32
theta = 10000
freqs = 1.0 / (theta ** (torch.arange(0, d, 2)[: (d // 2)].float() / d))

xfreq = torch.outer(x, freqs)

mags = (xfreq.sin().cumsum(dim=1).pow(2) + xfreq.cos().cumsum(dim=1).pow(2)).sqrt()

plt.plot(mags.sum(dim=1)/d)
plt.axhline(1.0, color='k', linestyle="--")
plt.xlabel("Positional difference $s$")
plt.ylabel("$B(s)/d$")
plt.show()
```