# YaRN: Efficient Context Window Extension of Large Language Models

**Bowen Peng**[1*] **Jeffrey Quesnelle**[1†] **Honglu Fan**[23] **Enrico Shippole**[‡]

[1]Nous Research [2]EleutherAI [3]University of Geneva

## Abstract

Rotary Position Embeddings (RoPE) have been shown to effectively encode positional information in transformer-based language models. However, these models fail to generalize past the sequence length they were trained on. We present YaRN (Yet another RoPE extensioN method), a compute-efficient method to extend the context window of such models, requiring 10x less tokens and 2.5x less training steps than previous methods. Using YaRN, we show that LLaMA models can effectively utilize and extrapolate to context lengths much longer than their original pre-training would allow, while also surpassing previous the state-of-the-art at context window extension. In addition, we demonstrate that YaRN exhibits the capability to extrapolate beyond the limited context of a fine-tuning dataset. We publish the checkpoints of Llama 2 7B/13B fine-tuned using YaRN with 64k and 128k context windows at https://github.com/jquesnelle/yarn.

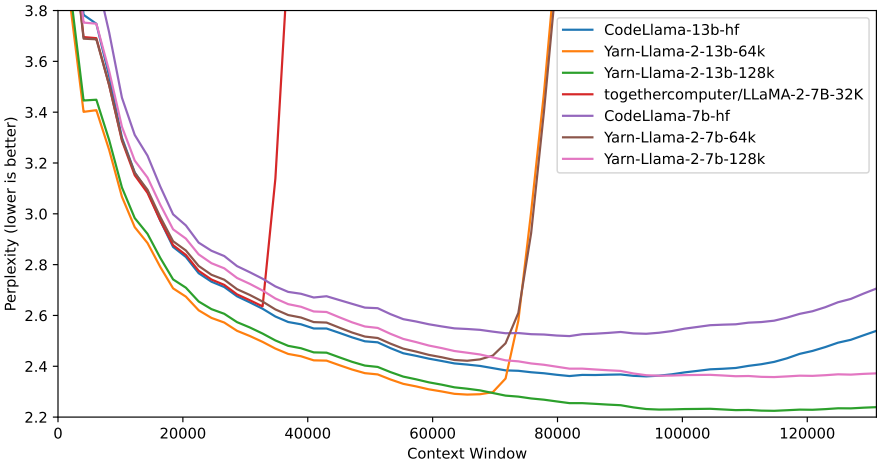


Figure 1: Sliding window perplexity ($S = 256$) of ten 128k Proof-pile documents truncated to evaluation context window size

[*]Reddit: `/u/bloc97` GitHub: `bloc97`

[†]Reddit: `/u/emozilla` X: `@theemozilla` GitHub: `jquesnelle`

[‡]X: `@EnricoShippole` GitHub: `conceptofmind`

Preprint. Under review.

# 1 Introduction

Transformer-based Large Language Models [34] (LLMs) have demonstrated strong abilities to perform *in-context learning* (ICL) and have become the near-ubiquitous choice for many natural language processing (NLP) tasks. The self-attention mechanism of transformers makes training highly parallelizable, allowing a long sequence to be processed in a distributed manner. The length of the sequences that an LLM is trained on is referred to as its *context window*.

The context window of a Transformer directly determines the amount of space where the examples can be provided, thus limiting its ICL ability. However, if the context window of a model is limited, there is less space to provide the model with robust examples from which ICL can be performed. Moreover, other tasks such as summarization would be severely hampered when the context window of a model is particularly short.

Following from the very nature of language itself, the position of tokens is crucial for effective modeling, whereas self-attention does not directly encode the positional information due to its parallel nature. In the architecture of Transformers, positional encoding is introduced to remedy this problem.

The original Transformer architecture used an absolute sinusoidal position encoding, which was later improved to a learnable absolute position encoding [12]. Since then, relative positional encoding schemes [26] have further increased the performance of Transformers. Currently, the most popular relative positional encodings are *T5 Relative Bias* [24], *RoPE* [28], *XPos* [29], and *ALiBi* [22].

One reoccurring limitation with positional encodings is the inability to generalize past the context window seen during training. While some methods such as ALiBi are able to do limited generalization, none are able to generalize to sequences significantly longer than their pre-trained length [18].

Some works have been done to overcome such limitation. [7] and concurrently [17] proposed to extend the context length by slightly modifying RoPE via Position Interpolation (PI) and fine-tuning on a small amount of data. As an alternative, we proposed the "NTK-aware" interpolation by taking the loss of high frequency into account in [4]. Since then, we have proposed two improvements of the "NTK-aware" interpolation, with different emphasis:

- the "Dynamic NTK" interpolation method for pre-trained models without fine-tuning.
- the "NTK-by-part" interpolation method which performs the best when fine-tuned on a small amount of longer-context data.

We have been excited to see that the "NTK-aware" interpolation and the "Dynamic NTK" interpolation have already seen their presence in the open-source models such as Code Llama [25] (using "NTK-aware" interpolation) and Qwen 7B [1] (using "Dynamic NTK").

In this paper, in addition to making a complete account of the previous works on the "NTK-aware", the "Dynamic NTK" and the "NTK-by-part" interpolations, we present YaRN (Yet another RoPE extensioN method), a method to efficiently extend the context window of models trained with Rotary Position Embeddings (RoPE), which include the LLaMA [32], GPT-NeoX [3], and PaLM [8] families of models. YaRN reaches state-of-the-art context window extension after fine-tuning on a less than $\sim$0.1%-sized representative sample of the original model's pre-training data.


## 1.1 Acknowledgements

The authors would like to thank Stability AI, Carper AI, and Eleuther AI for their generous support of significant computing resources that enabled the training of these models and the completion of this research. We would also like to thank Jonathan Tow and Dakota Mahan directly for their help in advising on the use of the Stability AI compute cluster. Additionally, we would like to thank a16z, and PygmalionAI, for providing resources to run evaluations and experiments on the models.


# 2 Background and Related Work

## 2.1 Rotary Position Embeddings

The basis of our work is the Rotary Position Embedding (RoPE) introduced in [28]. Given a sequence of input tokens $w_1, w_2, \cdots, w_L$, denote their embedding vectors by $\mathbf{x}_1, \cdots, \mathbf{x}_L \in \mathbb{R}^{|D|}$ where $|D|$ is

the dimension of the hidden states. Following the notation of [28], the attention layer first converts the embeddings plus position indices into the query and the key vectors:

$$\mathbf{q}_m = f_q(\mathbf{x}_m, m) \in \mathbb{R}^{|L|},\ \mathbf{k}_n = f_k(\mathbf{x}_n, n) \in \mathbb{R}^{|L|}, \tag{1}$$

where $|L|$ is the hidden dimension per head. Later, the attention score will be calculated as

$$\text{softmax}(\frac{\mathbf{q}_m^T \mathbf{k}_n}{\sqrt{|D|}}),$$

where $\mathbf{q}_m, \mathbf{k}_n$ are considered as column vectors and we are simply calculating the Euclidean inner product in the numerator. In RoPE, we first assume that $|D|, |L|$ are even and identify the embedding space and the hidden states as complex vector spaces:

$$\mathbb{R}^{|D|} \cong \mathbb{C}^{|D|/2},\ \mathbb{R}^{|L|} \cong \mathbb{C}^{|L|/2}$$

where the inner product $\mathbf{q}^T\mathbf{k}$ becomes the real part of the standard Hermitian inner product $\text{Re}(\mathbf{q}^*\mathbf{k})$. More specifically, the isomorphisms interleaves the real part and the complex part

$$\big((\mathbf{x}_m)_1, \cdots, (\mathbf{x}_m)_d\big) \mapsto \big((\mathbf{x}_m)_1 + i(\mathbf{x}_m)_2, \cdots, ((\mathbf{x}_m)_{|D|-1} + i(\mathbf{x}_m)_{|D|})\big), \tag{2}$$

$$\big((\mathbf{q}_m)_1, \cdots, (\mathbf{q}_m)_l\big) \mapsto \big((\mathbf{q}_m)_1 + i(\mathbf{q}_m)_2, \cdots, ((\mathbf{q}_m)_{|L|-1} + i(\mathbf{q}_m)_{|L|})\big). \tag{3}$$

To convert embeddings $\mathbf{x}_m, \mathbf{x}_n$ into query and key vectors, we are first given $\mathbb{R}$-linear operators

$$W_q, W_k : \mathbb{R}^{|D|} \cong \mathbb{C}^{|D|/2} \to \mathbb{R}^{|L|} \cong \mathbb{C}^{|L|/2}.$$

In complex coordinates, the functions $f_q, f_k$ are given by

$$f_q(\mathbf{x}_m, m) = W_q\mathbf{x}_m e^{im\theta},\ f_k(\mathbf{x}_n, n) = W_k\mathbf{x}_n e^{in\theta}, \tag{4}$$

where $\theta_d = b^{-2d/|D|}$ and $b = 10000$. The main point of doing so is that the dot product between the query vector and the key vector only depends on the relative distance $m - n$ as follows

$$\begin{aligned}
&\langle f_q(\mathbf{x}_m, m), f_k(\mathbf{x}_n, n)\rangle_{\mathbb{R}} && (5)\\
=&\text{Re}(\langle f_q(\mathbf{x}_m, m), f_k(\mathbf{x}_n, n)\rangle_{\mathbb{C}}) && (6)\\
=&\text{Re}(\mathbf{x}_m^* W_q^* W_k \mathbf{x}_n e^{i\theta(m-n)}) && (7)\\
=&g(\mathbf{x}_m, \mathbf{x}_n, m-n). && (8)
\end{aligned}$$

In real coordinates, the RoPE can be written as the following,

$$f_q(\mathbf{x}_m, m, \theta_d) = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_l & -\sin m\theta_l \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_l & \cos m\theta_l \end{pmatrix} W_q\mathbf{x}_m.$$

### 2.2 Positional Interpolation

As language models are usually pre-trained with a fixed context length, it is natural to ask how to extend the context length by fine-tuning on relatively less amount of data. For language models using RoPE as the positional embedding, [7], and concurrently `/u/kaiokendev` [17], proposed the Position Interpolation (PI) to extend the context length beyond the pre-trained limit. While a direct extrapolation does not perform well on sequences $w_1, \cdots, w_L$ with $L$ larger than the pre-trained limit, they discovered that interpolating the position indicies within the pre-trained limit works well with the help of a small amount of fine-tuning. Specifically, given a pre-trained language model with RoPE, they modify the RoPE by

$$f'_q(\mathbf{x}_m, m, \theta_d) = f_q\left(\mathbf{x}_m, \frac{mL'}{L}, \theta_d\right),\ f'_k(\mathbf{x}_m, m, \theta_d) = f_k\left(\mathbf{x}_m, \frac{mL'}{L}, \theta_d\right) \tag{9}$$

where $L' > L$ is a new context window beyond the pre-trained limit. With the original pre-trained model plus the modified RoPE formula, they fine-tuned the language model further on several orders of magnitude fewer tokens (a few billion in [7]) and successfully acheived context window extension.

### 2.3 Additional Notation

Since we assume that the interpolation of the RoPE embeddings is symmetric in both $(q, k)$ and $(\sin, \cos)$ domains, we rewrite and simplify Eq. 9 into the following general form:

$$f'_{q,k}(\mathbf{x}_m, m, \theta_d) = f_{q,k}(\mathbf{x}_m, g(m), h(\theta_d)) \tag{10}$$

where for PI,

$$s = \frac{L'}{L} \tag{11}$$

$$g(m) = s \cdot m \tag{12}$$

$$h(\theta_d) = \theta_d \tag{13}$$

The value $s$ is also commonly referred to as the *scale factor* of the context length extension.

In the subsequent sections, whenever we introduce a new interpolation method, we only need to specify the functions $g(m)$ and $h(\theta_d)$.

Additionally, we define $\lambda_d$ as the *wavelength* of the RoPE embedding at the dimension $d$

$$\lambda_d = 2\pi b^{\frac{2d}{|D|}} \tag{14}$$

for base $b$, dimension $d$ and total number of dimensions $|D|$. The wavelength describes the length of tokens needed in order for the RoPE embedding at dimension $d$ to perform a full rotation ($2\pi$). It is the highest at the lowest dimension and the lowest at the highest dimension.

Given that some interpolation methods (eg. Positional Interpolation) do not care about the wavelength of the dimensions, we will refer to those methods as "blind" interpolation methods, while others do (eg. YaRN), which we will classify as "targeted" interpolation methods.

### 2.4 Related work

ReRoPE [27] also aims to extend the context size of existing models pre-trained with RoPE, and claims "infinite" context length without needing any fine-tuning. This claim is backed by a monotonically decreasing loss with increasing context length up to 16k on the Llama 2 13B model. It achieves context extension by modifying the attention mechanism and thus is not purely an embedding interpolation method. Since it is currently not compatible with Flash Attention 2 [10] and requires two attention passes during inference, we do not consider it for comparison.

Concurrently with our work, LM-Infinite [13] proposes similar ideas to YaRN, but focuses on "on-the-fly" length generalization for non-fine-tuned models. Since they also modify the attention mechanism of the models, it is not an embedding interpolation method and is not immediately compatible with Flash Attention 2.

## 3 Methodology

Whereas PI stretches all RoPE dimensions equally, we find that the theoretical interpolation bound described by PI [7] is insufficient at predicting the complex dynamics between RoPE and the LLM's internal embeddings. In the following subsections, we describe the main issues with PI we have individually identified and solved, so as to give the readers the context, origin and justifications of each method which we use in concert to obtain the full YaRN method.

### 3.1 Loss of High Frequency information - "NTK-aware" Interpolation [4]

If we look at RoPE only from an information encoding perspective, it was shown in [30], using Neural Tangent Kernel (NTK) theory, that deep neural networks have trouble learning high frequency

information if the input dimension is low and the corresponding embeddings lack high frequency components. Here we can see the similarities: a token's positional information is one-dimensional, and RoPE expands it to an n-dimensional complex vector embedding.

RoPE closely resembles Fourier Features [30] in many aspects, as it is possible to define RoPE as a special 1D case of a Fourier Feature. Stretching the RoPE embeddings indiscriminately results in the loss of important high frequency details which the network needs in order to resolve tokens that are both very similar and very close together (the rotation describing the smallest distance needs to not be too small for the network to be able to detect it).

We hypothesise that the slight increase of perplexity for short context sizes after fine-tuning on larger context sizes seen in PI [7] might be related to this problem. Under ideal circumstances, there is no reason that fine-tuning on larger context sizes should degrade the performance of smaller context sizes.

In order to resolve the problem of losing high frequency information when interpolating the RoPE embeddings, the "NTK-aware" interpolation was developed in [4]. Instead of scaling every dimension of RoPE equally by a factor $s$, we spread out the interpolation pressure across multiple dimensions by scaling high frequencies less and low frequencies more. One can obtain such a transformation in many ways, but the simplest would be to perform a base change on the value of $\theta$.

As we want the lowest frequency to be scaled as much as linear positional scaling and the highest frequency to stay constant, we need to find a new base $b'$ such that the last dimension matches the wavelength of linear interpolation with a scale factor $s$. Since the original RoPE method skips odd dimensions in order to concatenate both $\cos(\frac{2\pi x}{\lambda})$ and $\sin(\frac{2\pi x}{\lambda})$ components into a single embedding, the last dimension $d \in D$ is $|D| - 2$.

Solving for $b'$ yields:

$$b'^{\frac{|D|-2}{|D|}} = s \cdot b^{\frac{|D|-2}{|D|}} \tag{15}$$

$$b' = b \cdot s^{\frac{|D|}{|D|-2}} \tag{16}$$

Following the notation of Section 2.3, the "NTK-aware" interpolation scheme is simply applying the base change formula as such:

$$g(m) = m \tag{17}$$

$$h(\theta_d) = b'^{-2d/|D|} \tag{18}$$

In our tests [4], this method performs much better at extending the context size of non-fine-tuned models compared to PI [7]. However, one major disadvantage of this method is that given it is not just an interpolation scheme, some dimensions are slightly extrapolated to "out-of-bound" values, thus fine-tuning with "NTK-aware" interpolation [4] yields inferior results to PI [7]. Furthermore, due to the "out-of-bound" values, the theoretical scale factor $s$ does not accurately describe the true context extension scale. In practice, the scale value $s$ has to be set higher than the expected scale for a given context length extension.

Shortly before the release of this paper, Code Llama [25] was released which uses "NTK-aware" scaling by manually setting the base $b$ to 1M.

### 3.2 Loss of Relative Local Distances - "NTK-by-parts" Interpolation [5]

One interesting observation of RoPE embeddings is that given a context size $L$, there are some dimensions $d$ where the wavelength is longer than the maximum context length seen during pretraining ($\lambda > L$), this suggests that some dimensions' embeddings might not be distributed evenly in the rotational domain.

In the case of PI and "NTK-aware" interpolation, we treat all the RoPE hidden dimensions equally (as in they have the same effect on the network). However, we have found experimentally that the network treats some dimensions differently from others. As stated before, given a context length $L$,

the wavelengths of some dimensions $\lambda$ is larger than or equal to $L$. Given that when the wavelength of a hidden dimension is larger than or equal to $L$, all pairs of positions encode an unique distance, we hypothesize that the absolute positional information is retained, whereas when the wavelength is short, only the relative positional information is available to the network.

When we stretch all the RoPE dimensions either by a scale $s$ or using a base change $b'$, all tokens become closer to each other, as the dot product of two vectors rotated by a lesser amount is bigger. This scaling severely impairs a LLM's ability to understand small and local relationships between its internal embeddings. We hypothesize that such compression leads to the model being confused on the positional order of close-by tokens, and consequently harming the model's abilities.

In order to remedy this issue, given the observation that we have found here, we choose not to interpolate the higher frequency dimensions at all. In particular,

- if the wavelength $\lambda$ is much smaller than the context size $L$, we do not interpolate;
- if the wavelength $\lambda$ is equal or bigger than the context size $L$, we want to only interpolate and avoid any extrapolation (unlike the previous "NTK-aware" method);
- dimensions in-between can have a bit of both, similar to the "NTK-aware" interpolation.

To find the dimension $d$ we want given the number of full rotations $r$ at a certain context length $L$, we can expand Eq. 14 as such:

$$\lambda_d = 2\pi b^{\frac{2d}{|D|}} \tag{19}$$

$$r = \frac{L}{\lambda_d} \tag{20}$$

Solving for $d$ yields:

$$d = \frac{|D|}{2\ln(b)} \ln\left(\frac{L}{2\pi r}\right) \tag{21}$$

We also propose that all dimensions $d$ where $r < \alpha$ to be dimensions where we linearly interpolate by a scale $s$ (exactly like PI, avoiding any extrapolation), and $r > \beta$ to be not interpolated at all (always extrapolated). Define the ramp function $\gamma_d$ to be

$$\gamma_d(r) = \begin{cases} 0, & \text{if } r < \alpha \\ 1, & \text{if } r > \beta \\ \dfrac{r-\alpha}{\beta-\alpha}, & \text{otherwise} \end{cases} \tag{22}$$

With the help of the ramp function, we define the new wavelength as

$$\lambda'_d = (1 - \gamma_d)s\lambda_d + \gamma_d\lambda_d. \tag{23}$$

The values of $\alpha$ and $\beta$ should be tuned in a case-by-case basis. For example, we have found experimentally that for the Llama family of models, good values for $\alpha$ and $\beta$ are $\alpha = 1$ and $\beta = 32$.

After converting $\lambda_d$ to $\theta_d$, the method can be described as:

$$g(m) = m \tag{24}$$

$$h(\theta_d) = (1 - \gamma_d)\frac{\theta_d}{s} + \gamma_d\theta_d \tag{25}$$

Using the techniques described in this section, a variant of the resulting method was released under the name "NTK-by-parts" interpolation [5]. This improved method performs better than the previous PI [7] and "NTK-aware" 3.1 interpolation methods, both with non-fine-tuned models and with fine-tuned models. As this method avoids extrapolating the dimensions that have uneven distribution in the rotational domain, it avoids all of the fine-tuning issues from previous methods.

### 3.3 Dynamic Scaling - "Dynamic NTK" Interpolation [11]

When using RoPE interpolation methods to extend the context size without fine-tuning, we would like the model to degrade gracefully at longer context sizes, rather than taking the full degradation hit across the entire context size if the scale $s$ is set to a value higher than what is needed. Recall that in Section 2.3, $s = L'/L$ in the PI where $L$ is the trained context length and $L'$ is the newly extended context length. In the "Dynamic NTK" method, we dynamically calculate the scale $s$ as follows:

$$s = \begin{cases} \frac{L'}{L}, & \text{if } \frac{L'}{L} > 1 \\ 1, & \text{otherwise} \end{cases} \tag{26}$$

Dynamically changing the scale during inference when the context size is exceeded allows all models to gracefully degrade instead of immediately breaking when hitting the trained context limit $L$.

Some care has to be taken when using Dynamic Scaling with kv-caching [6], as in some implementations, the RoPE embeddings are cached. The correct implementation should cache the kv-embeddings before applying RoPE, as the RoPE embedding of every token changes when $s$ changes.

### 3.4 Increase in Average Minimum Cosine Similarity for Long Distances - YaRN

Even if we solve the issue of local distances described in Section 3.2, larger distances must be interpolated at the threshold $\alpha$ in order to avoid extrapolation. Intuitively, this should not seem to be an issue as global distances does not require high precision to be able to distinguish token positions (i.e. the network only needs to broadly know if the token is at the beginning, middle or the end of the sequence). However, we find out that because the average minimum distances become closer with increasing number of tokens [4], it skews the attention softmax distribution to become "spikier" (i.e. decreases the average entropy of the attention softmax). In other words, as the effect of long distance decay is diminished due to interpolation, the network "pays more attention" to more tokens. This distribution shift causes a degradation in the LLM's outputs which is independent of the previous issues.

Since there is a decrease of entropy in the attention softmax distribution as we interpolate RoPE embeddings to longer context sizes, our goal is to reverse the decrease of entropy (i.e. increase the "temperature" of the attention logits). This can be done by multiplying the intermediate attention matrix by a temperature $t > 1$ before applying the softmax, but as RoPE embeddings are encoded as a rotation matrix, we can simply scale the length of the RoPE embeddings by a constant factor $\sqrt{t}$. The "length scaling" trick allows us to avoid any modifications to the attention code, which significantly simplifies integration into existing training and inference pipelines and has a time complexity of $O(1)$.

As our RoPE interpolation scheme interpolates RoPE dimensions unevenly, it is difficult to compute an analytical solution for the temperature scale $t$ required with respect to the scale $s$. Fortunately, we have found experimentally that by minimizing perplexity, all Llama models follow somewhat closely the same fitted curve:

$$\sqrt{t} \approx 0.1 \ln(s) + 1 \tag{27}$$

The equation above was found by fitting $\sqrt{t}$ where perplexity was lowest against the scale extension by a factor $s$, without fine-tuning, on LLaMA 7b, 13b, 33b and 65b models, using the interpolation methodology described in 3.2. We have also found that this equation applies fairly well to Llama 2 models (7b, 13b and 70b), with only slight differences. It suggests that this property of increasing the entropy is common and generalizable across different models and training data.

This final modification yields our final YaRN method, where the method described in 3.2 is combined with 3.4 to be used during both training and inference. It surpasses all previous methods in both

[4] More precisely, if we select $N$ random points uniformly on a line of length $L$, the average minimum distance between any two points follow the equation $\frac{L}{N^2-1}$

fine-tuned and non-fine-tuned scenarios and does not require modification to the inference code at all. Only a modification to the algorithm that initially generates the RoPE embeddings is required. The simplicity of YaRN allows it to be implemented readily in all inference and training libraries, including compatibility with Flash Attention 2 [10].

### 3.5 Extrapolation and Transfer Learning

In Code Llama [25], a dataset with 16k context was used with a scale factor set to $s \approx 88.6$, which corresponds to a context size of 355k. They show that the network extrapolates up to 100k context without ever seeing those context sizes during training. Similar to 3.1 and [25], YaRN also supports training with a higher scale factor $s$ than the length of the dataset. Due to compute constraints, we test only $s = 32$ by further fine-tuning the $s = 16$ model for 200 steps using the same dataset with 64k context.

We show in 4.2 that the $s = 32$ model successfully extrapolates up to 128k context using only 64k context during training. Unlike previous "blind" interpolation methods, YaRN is much more efficient at transfer learning when increasing the scale $s$. This demonstrates successful transfer learning from $s = 16$ to $s = 32$ without the network needing to relearn the interpolated embeddings, as the $s = 32$ model is equivalent to the $s = 16$ model across the entire context size, despite only being trained on $s = 32$ for 200 steps.

## 4 Experiments

We show that YaRN successfully achieves context window extension of LLMs. Moreover, this result is achieved with only 400 training steps, representing approximately 0.1% of the model's original pre-training corpus, a 10x reduction from [25] and 2.5x reduction in training steps from [7], making it highly compute-efficient for training with no additional inference costs. We calculate the perplexity of long documents and score on established benchmarks to evaluate the resulting models, finding that they surpass all other context window extension methods.

We broadly followed the training and evaluation procedures as outlined in [7].

### 4.1 Training

For training, we extended the Llama 2 [33] 7B and 13B parameter models. No changes were made to the LLaMA model architecture other than the calculation of the embedding frequencies as described in 3.4 with $s = 16$ and $s = 32$.

We used a learning rate of $2 \times 10^{-5}$ with no weight decay and a linear warmup of 20 steps along with AdamW [20] $\beta_1 = 0.9$ and $\beta_2 = 0.95$. For $s = 16$ we fine-tuned for 400 steps with global batch size $64$ using PyTorch [21] Fully Sharded Data Parallelism [36] and Flash Attention 2 [10] on the PG19 dataset [23] chunked into 64k segments bookended with the BOS and EOS token. For $s = 32$ we followed the same procedure, but started from the finished $s = 16$ checkpoint and trained for an additional 200 steps.

### 4.2 Evaluation

#### 4.2.1 Long Sequence Language Modeling

To evaluate long sequence language modeling performance we use the GovReport [15] and Proof-pile [2] datasets, both of which contain many long sequence samples. For all evaluations, the test splits of both datasets were used exclusively. All perplexity evaluations were calculated using the sliding window method from [22] with $S = 256$.

Firstly, we evaluated how the model performed as the context window increases. We selected 10 random samples from Proof-pile that were at least 128k tokens in length and evaluated the calculated the perplexity of each of these samples when truncated at 2k steps from a sequence length of 2k tokens through 128k tokens. Figure 1 shows the results compared to other context window extension methods. In particular, we compare against the a publicly available PI model trained at 32k from Together.ai [31] and "NTK-aware" Code Llama [25]. These results are summarized in Table 1.

| Model Size | Model Name | Context Window | Extension Method | Evaluation Context Window Size | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 8192 | 32768 | 65536 | 98304 | 131072 |
| 7B | Together | 32k | PI | 3.50 | 2.64 | $> 10^2$ | $> 10^3$ | $> 10^4$ |
| 7B | Code Llama | 100k | NTK | 3.71 | 2.74 | 2.55 | 2.54 | 2.71 |
| 7B | YaRN ($s = 16$) | 64k | YaRN | 3.51 | 2.65 | 2.42 | $> 10^1$ | $> 10^1$ |
| 7B | YaRN ($s = 32$) | 128k | YaRN | 3.56 | 2.70 | 2.45 | 2.36 | 2.37 |
| 13B | Code Llama | 100k | NTK | 3.54 | 2.63 | 2.41 | 2.37 | 2.54 |
| 13B | YaRN ($s = 16$) | 64k | YaRN | 3.25 | 2.50 | 2.29 | $> 10^1$ | $> 10^1$ |
| 13B | YaRN ($s = 32$) | 128k | YaRN | 3.29 | 2.53 | 2.31 | 2.23 | 2.24 |

Table 1: Sliding window perplexity ($S = 256$) of ten 128k Proof-pile documents truncated to evaluation context window size

Table 2 shows the final perplexity on 50 untruncated GovReport documents at least 16k tokens in length evaluated at a 32k context window.

| Model Size | Model Name | Context Window | Extension Method | Perplexity |
|---|---|---|---|---|
| 7B | Together | 32k | PI | 3.67 |
| 7B | Code Llama | 100k | NTK | 4.44 |
| 7B | YaRN ($s = 16$) | 64k | YaRN | 3.59 |
| 7B | YaRN ($s = 32$) | 128k | YaRN | 3.64 |
| 13B | Code Llama | 100k | NTK | 4.22 |
| 13B | YaRN ($s = 16$) | 64k | YaRN | 3.35 |
| 13B | YaRN ($s = 32$) | 128k | YaRN | 3.39 |

Table 2: Sliding window perplexity ($S = 256$) of 50 long GovReport documents with a fixed context window size of 32k

We observe that the model exhibits strong performance across the entire context window, outperforming all other context window extension methods. Of particular note is the YaRN ($s = 32$) models, which show continued declining perplexity through 128k, despite the fine-tuning data being limited to 64k tokens in length. This demonstrates that the model is able to generalize to unseen context lengths.

#### 4.2.2 Standardized Benchmarks

The Hugging Face Open LLM Leaderboard [16] compares a multitude of LLMs across a standardized set of four public benchmarks. Specifically, 25-shot ARC-Challenge [9], 10-shot HellaSwag [35], 5-shot MMLU [14], and 0-shot TruthfulQA [19].

To test degradation of model performance under context extension, we evaluated our models using this suite and compared it to established scores for the Llama 2 baselines as well as publicly available PI and "NTK-aware" models. The results are summarized in Table 3.

We observe that there is minimal performance degradation between the YaRN models and their respective Llama 2 baselines. We also observe that there was on average a 0.49% drop in scores between the YaRN $s = 16$ and $s = 32$ models. From this we conclude that the the iterative extension from 64k to 128k results in negligible performance loss.

## 5 Conclusion

In conclusion, we have shown that YaRN improves upon all existing RoPE interpolation methods and can act as a drop-in replacement to PI, with no downsides and minimal implementation effort. The fine-tuned models preserve their original abilities on multiple benchmarks while being able to attend to a very large context size. Furthermore, YaRN allows efficient extrapolation with fine-tuning on shorter datasets and can take advantage of transfer learning for faster convergence, both of

| Model Size | Model Name | Context Window | Extension Method | ARC-c | Hellaswag | MMLU | TruthfulQA |
|---|---|---|---|---|---|---|---|
| 7B | Llama 2 | 4k | None | 53.1 | 77.8 | 43.8 | 39.0 |
| 7B | Together | 32k | PI | 47.6 | 76.1 | 43.3 | 39.2 |
| 7B | Code Llama | 100k | NTK | 39.9 | 60.8 | 31.1 | 37.8 |
| 7B | YaRN ($s = 16$) | 64k | YaRN | 52.3 | 78.8 | 42.5 | 38.2 |
| 7B | YaRN ($s = 32$) | 128k | YaRN | 52.1 | 78.4 | 41.7 | 37.3 |
| 13B | Llama 2 | 4k | None | 59.4 | 82.1 | 55.8 | 37.4 |
| 13B | Code Llama | 100k | NTK | 40.9 | 63.4 | 32.8 | 43.8 |
| 13B | YaRN ($s = 16$) | 64k | YaRN | 58.1 | 82.3 | 52.8 | 37.8 |
| 13B | YaRN ($s = 32$) | 128k | YaRN | 58.0 | 82.2 | 51.9 | 37.3 |

Table 3: Performance of context window extensions methods on the Hugging Face Open LLM benchmark suite compared with original Llama 2 baselines

which are crucial under compute-constrained scenarios. Finally, we have shown the effectiveness of extrapolation with YaRN where it is able to "train short, and test long".

## References


[1] Introducing Qwen-7B: Open foundation and human-aligned models (of the state-of-the-arts). URL https://github.com/QwenLM/Qwen-7B/blob/main/tech_memo.md.

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

[3] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, M. Pieler, U. S. Prashanth, S. Purohit, L. Reynolds, J. Tow, B. Wang, and S. Weinbach. GPT-NeoX-20B: An open-source autoregressive language model, 2022. arXiv: 2204.06745.

[4] bloc97. NTK-Aware Scaled RoPE allows LLaMA models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation., 2023. URL https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/.

[5] bloc97. Add NTK-Aware interpolation "by parts" correction, 2023. URL https://github.com/jquesnelle/scaled-rope/pull/1.

[6] C. Chen. Transformer Inference Arithmetic, 2022. URL https://kipp.ly/blog/transformer-inference-arithmetic/.

[7] S. Chen, S. Wong, L. Chen, and Y. Tian. Extending context window of large language models via positional interpolation, 2023. arXiv: 2306.15595.

[8] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. PaLM: Scaling language modeling with pathways, 2022. arXiv: 2204.02311.

[9] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try ARC, the AI2 Reasoning Challenge, 2018.

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

[11] emozilla. Dynamically Scaled RoPE further increases performance of long context LLaMA with zero fine-tuning, 2023. URL https://www.reddit.com/r/LocalLLaMA/comments/14mrgpr/dynamically_scaled_rope_further_increases/.

[12] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning, 2017. arXiv: 1705.03122.

[13] C. Han, Q. Wang, W. Xiong, Y. Chen, H. Ji, and S. Wang. LM-Infinite: Simple on-the-fly length generalization for large language models, 2023.

[14] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[15] L. Huang, S. Cao, N. Parulian, H. Ji, and L. 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. Association for Computational Linguistics, June 2021.

[16] Hugging Face. Open LLM Leaderboard, 2023. URL https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard.

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

[18] A. Kazemnejad, I. Padhi, K. N. Ramamurthy, P. Das, and S. Reddy. The impact of positional encoding on length generalization in transformers, 2023. arXiv: 2305.19466.

[19] S. Lin, J. Hilton, and O. 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, May 2022.

[20] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019.

[22] O. Press, N. Smith, and M. Lewis. Train Short, Test Long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022.

[23] J. W. Rae, A. Potapenko, S. M. Jayakumar, C. Hillier, and T. P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

[24] A. Roberts, C. Raffel, K. Lee, M. Matena, N. Shazeer, P. J. Liu, S. Narang, W. Li, and Y. Zhou. Exploring the limits of transfer learning with a unified text-to-text transformer. Technical report, Google, 2019.

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

[26] P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[27] J. Su. Rectified rotary position embeddings. https://github.com/bojone/rerope, 2023.

[28] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu. RoFormer: Enhanced transformer with rotary position embedding, 2022. arXiv: 2104.09864.

[29] Y. Sun, L. Dong, B. Patra, S. Ma, S. Huang, A. Benhaim, V. Chaudhary, X. Song, and F. Wei. A length-extrapolatable transformer, 2022.

[30] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[31] Together.ai. LLaMA-2-7B-32K, 2023. URL `https://huggingface.co/togethercomputer/LLaMA-2-7B-32K`.

[32] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and efficient foundation language models, 2023. arXiv: 2302.13971.

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

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

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

[36] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, A. Desmaison, C. Balioglu, B. Nguyen, G. Chauhan, Y. Hao, and S. Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel, 2023.