Extending Context Size of Pre-Trained Large Language Model

Rotational Positional Embedding

LLM Course (MVA)

Paul Caucheteux, Sacha Hakim, Quentin Mayedpour et Tom Welch

ENS-Paris-Saclay

Mars 2025

Sommaire

Sommaire

Positional Embedding

Position embedding (PE) is the cornerstone of transformers :

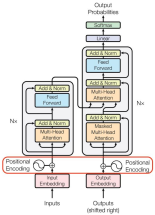


Figure – Classical Positional encoding : Sinusoidal [Vas+17].

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \qquad (1)$$

Problems : Information is injected too early and get mixed with embeddings : hard to distinguish position. Absolute positional encoding : bad understanding of dependencies between tokens, bad generalization...

RPE and RoPE idea

Much work following the paper of Vaswani et al. [Vas+17] to improve PE. Notably RPE, first introduced by Shaw et al. (2018) [SUV18] :

- Idea : Incorporate relative positional information during the computation of $q_i^T k_j$ and $z_i = \sum_{j=1}^{N} \text{sim}(i,j) v_j$.
- Better understanding of dependencies and better generalization to unseen positions (by clipping relative positions).
- Various versions of RPE followed, always focusing on modifying the the computation of $q_i^T k_j$.

Incompatibility with the structure of the scalar product. Can we find f_Q, f_K such that $\langle f_Q(x_m), f_K(x_n) \rangle = g(x_m, x_n, n-m)$?

RoPE (2D)

In 2D, a solution is given (in complex formulation) by :

$$f_Q(x_m) = (W^Q x_i)e^{i\theta m}$$
$$f_K(x_n) = (W^Q x_n)e^{i\theta n},$$

for a fixed $\theta \in \mathbb{R}^*$.

Interpretation : Apply a rotation to each query/key by an angle proportional to their position index. This introduces relative position information directly in the attention computation, while preserving scalar product formulation.

RoPE General Case

For an even embedding dimension d, we divide the embeddings into $\frac{d}{2}$ blocks. Then, the solution in 2D naturally generalizes in d dimensions. We can write $f_K(x_m, m) = \text{Rot}^{d}_{\theta,m} W^K x_m$ with :

$$\text{Rot}^{d}_{\theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

Interpretation : This time, we apply a rotation to each block of 2 dimensions to the queries/keys, by a proportional angle to the position index.

Pros : RPE incorporated during the computation of keys and values. Compatibility with linear transformers ($O(N)$ instead of $O(N^2)$). Very interpretable. Choosing $\theta_i = 10000^{-2i/d}$ allows for a long-term decay property.

Extending the Context Window in LLMs

Problem : Most LLMs are pre-trained with a fixed context window (e.g., LLaMA2 with 4096 tokens).

Limitation : This constraint is incompatible with long-form tasks :

- Conducting long conversations, summarizing long documents ...

Goal : Can we extend the context window *without* re-training from scratch ?

Naive solution : fine-tune an existing pre-trained model with a longer context window
$\rightarrow$ Empirically slow and ineffective. (catastrophic values)

Solution : We want to extend the context from L to L'. We can interpolate between position that the model already know
Define the *extension ratio* as :

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

Proposed Improvements to RoPE

We simplify RoPE embeddings as :

$$\left[\cos\left(\frac{m}{\lambda_s \beta^i}\right),\ \sin\left(\frac{m}{\lambda_s \beta^i}\right)\right] \quad \text{with } i = 0, \ldots, \frac{d}{2} - 1,\ \beta = \theta^{\frac{2}{d}},\ \lambda_s = 1$$

Several interpolation strategies have been proposed to extend the context window :

- **Linear Positional Interpolation (PI)** [Che+23] : $\lambda_s = s$
  Linearly rescales all RoPE frequencies. However, this leads to *crowded position encodings*, making it hard to distinguish nearby tokens.
- **NTK Interpolation** [23] : $\lambda_s = s^i$
  Distributes interpolation across dimensions. Lower-index (high-frequency) dimensions receive less scaling, preserving short-range information.
- **YaRN** [Pen+23] : Adapt the interpolation on the frequency :
  - *High frequency* (first dimensions) : $\lambda_s = 1$ (no interpolation)
  - *Medium frequency* : $\lambda_s = s^i$ (NTK)
  - *Low frequency* (last dimensions) : $\lambda_s = s$ (PI)

Sommaire

Key Findings from LongRoPE

Finding 1 : RoPE dimensions exhibit non-uniformities that are not properly handled by existing interpolation methods.
$\rightarrow$ Introduce a specialized rescale factor λ_i for each dimension.

Finding 2 : The first token positions are crucial, as they often get higher attention.
$\rightarrow$ Preserve the first $\hat{n}$ positions without interpolation, and search for the optimal $\hat{n}$.

Solution : LongRoPE [Din+24] proposes an efficient search algorithm to jointly optimize $\{\lambda_i\}$ and $\hat{n}$, addressing both findings.

Note : With a pre-trained model, interpolation methods can be applied :

- Directly in a non-fine-tuning scenario.
- As an initialization for more effective fine-tuning.

Finding 3 : Both settings will benefit from Long RoPE.

Formulation of the problem and Algorithm

The optimization problem is then formulated as follows :

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

where $\text{RoPE(n)} = \left[\cdots, \cos\left(\frac{n}{\mathbb{I}(\lambda_i,\hat{n})\beta^i}\right), \sin\left(\frac{n}{\mathbb{I}(\lambda_i,\hat{n})\beta^i}\right), \cdots\right], \quad i = 0, \ldots, \frac{d}{2} - 1;\ n \in [0, |\mathbf{x}|]$, and $\mathbb{I}(\lambda_i, \hat{n}) = \begin{cases} 1 & \text{if } n < \hat{n}, \\ \lambda_i & \text{if } n \geq \hat{n} \end{cases}$

Key points of the algorithm :

- Start from 3 initial configurations (NTK, PI, and YaRN), then randomly mutate any λ_s^i.
- Impose the constraint $\lambda_i < \lambda_{i+1}$ to reduce the search space and align with the NTK intuition.
- Evaluate each configuration by computing the model's perplexity on example inputs, and retain the best-performing one.

Method

To obtain the final model, a **progressive fine-tuning strategy** is used :

- Extend the pre-trained model to $L' = 256k$ using LongRoPE search.
- Fine-tune at 256k context length.
- Extend to $L' = 2048k$ ($512\times$) without further fine-tuning using LongRoPE search.

To mitigate performance loss on short context lengths : perform an inverse LongRoPE search down to 8k and dynamically adjust during inference.

To assess performance : use either perplexity or estimate the effective context length by inserting a key token and checking how far the model can retrieve the information (Passkey retrieval).

Sommaire

Implementation

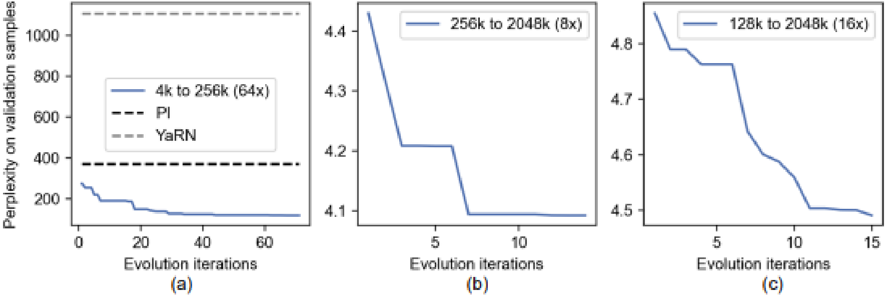
How to select λ_i and $\hat{n}$? Evolution-based search [Guo+20]

- Algorithm to find the best architecture from a (really) large panel of choice
- Biology based : select the best parameters, crossover them, mutate them

```
def evolutionnary_search(model, data, extension_ratio, population_size,
                         num_mutations, num_crossovers, max_iterations,):
    search_space = init_search_space()
    population = initialize_population(population_size, search_space, model.d_model)

    for _ in range(max_iterations):

        # Get the score (perplexity) for each configurations (population)
        perplexities = evaluate_population(model, data, population)

        # Select the top-performing individuals as parents
        indices = np.argsort(perplexities)[:population_size//2]
        parents = [population[i] for i in indices]

        # Create new population through mutation and crossover
        mutated = mutate(parents, num_mutations, model.d_model) # randomly change attributes
        crossed = crossover(parents, num_crossovers, model.d_model) # cross attributes
        population = mutated + crossed  # we combine the two list
    # select the best
    best_individual = min(population, key=lambda x: evaluate_individual(model, data, x))
    return best_individual["lambda_i"], best_individual["n_hat"]
```

Implementation

- No fine-tuning doesn't mean fast to compute !
- If the weights are not updated, we still need to call the model (a lot) during this phase
- Takes about 1 hour for a search with non "dummy" parameters
- But according to the authors, the evolution-search converges "quickly"



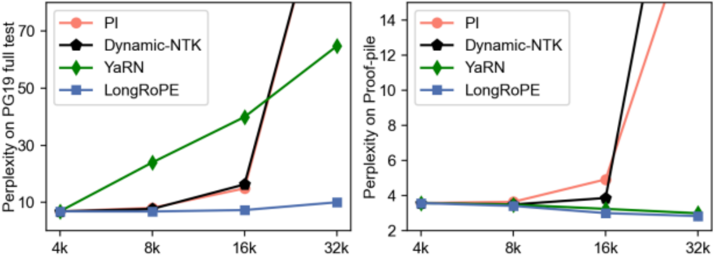
Implementation : Dataset

How to create a (really) large dataset

- Take entire books ? Long context indeed but for most words in the book, the model does not need to know what the authors said in the beginning
- Construct a dataset ? For instance create a sentence containing a key, add a lot of text and then ask to return the key.
  - Not relevant in practice, and really long to construct
- Important task : design really large dataset to effectively assess the performance of an LLM on long context

Results

- LongRope beats all the other methods on all dataset and contexts lengths
- However, the model is only evaluated on next token prediction with large datasets
- In fact, LongRope seems to be an effective way to keep the LLM stable when we increase the context lengths, but can it really understand and use informations from really long texts ?



Sommaire

Conclusion et Discussion

-RoPE :

- Relative Position Embedding
- Flexibility of sequence length
- Better performance on long-text tasks

-LongRope :

- Frequency scaling
- Improved extrapolation
- Efficiency scalability
- **2048k tokens**

Bibliographie I

[23] *LocalLLaMA. Dynamically scaled RoPE further increases performance of long context LLaMA with zero fine-tuning*. Accessed : 2025-03-20. 2023.

[Che+23] Shouyuan Chen et al. *Extending Context Window of Large Language Models via Positional Interpolation*. 2023. arXiv : 2306.15595 [cs.CL]. URL : https://arxiv.org/abs/2306.15595.

[Din+24] Yiran Ding et al. *LongRoPE : Extending LLM Context Window Beyond 2 Million Tokens*. 2024. arXiv : 2402.13753 [cs.CL]. URL : https://arxiv.org/abs/2402.13753.

Bibliographie II

[Guo+20] Zichao GUO et al. *Single Path One-Shot Neural Architecture Search with Uniform Sampling*. 2020. arXiv : 1904.00420 [cs.CV]. URL : https://arxiv.org/abs/1904.00420.

[Pen+23] Bowen PENG et al. *YaRN : Efficient Context Window Extension of Large Language Models*. 2023. arXiv : 2309.00071 [cs.CL]. URL : https://arxiv.org/abs/2309.00071.

[SUV18] Peter SHAW, Jakob USZKOREIT et Ashish VASWANI. *Self-Attention with Relative Position Representations*. 2018. arXiv : 1803.02155 [cs.CL]. URL : https://arxiv.org/abs/1803.02155.

Bibliographie III

[Vas+17] Ashish VASWANI et al. *Attention Is All You Need*. 2017. arXiv : 1706.03762 [cs.CL]. URL : https://arxiv.org/abs/1706.03762.