Fractional Rotation, Full Potential?
Investigating Performance and Convergence of Partial RoPE

Anonymous ACL submission

Abstract

Rotary Positional Embedding (RoPE) is a common choice in transformer architectures for encoding relative positional information. Although earlier work has examined omitting RoPE in specific layers, the effect of varying the fraction of hidden dimensions that receive rotary transformations remains largely unexplored. This design choice can yield substantial memory savings, which becomes especially significant at long context lengths— we find up to 10× memory savings over the standard RoPE cache, while achieving comparable final loss. In this work, we present a systematic study examining the impact of partial RoPE on training dynamics and convergence across architectures and datasets. Our findings uncover several notable patterns: (1) applying RoPE to only a small fraction of dimensions (around 10%) achieves convergence comparable to using full RoPE; (2) these trends hold consistently across sequence lengths and datasets of varying quality and architectures, with higher-quality data resulting in lower overall loss; and (3) parallel attention architectures exhibit instability under NoPE (No Positional Encoding), which can be alleviated through minimal RoPE application or QK-Norm which converges to a higher loss. Together, these results offer practical guidance for model designers aiming to balance computational efficiency and training stability, while emphasizing the previously overlooked importance of partial RoPE.

1 Introduction

Transformers use positional encodings to capture the order of tokens within a sequence, since the other parts of the model themselves are permutation-invariant. Rotary Positional Embedding (RoPE) (Su et al., 2023) has emerged as a leading method for encoding relative positions directly in the query–key interactions in self-attention (Vaswani et al., 2017) computations. RoPE is favored in modern decoder-only language models for its simplicity, strong empirical performance, and ability to generalize to sequences longer than those seen during training. It is widely adopted across architectures, either as the sole positional encoding or alongside alternatives such as No Positional Encoding (NoPE) (Kazemnejad et al., 2023).

Despite the widespread adoption of RoPE, a fundamental design choice remains largely unexplored: the fraction of hidden dimensions within each attention head that undergoes the rotary transformation. This parameter varies considerably across major model families, highlighting a lack of consensus on best practices. Early implementations in models like GPT-J (Wang and Komatsuzaki, 2021) and GPT-NeoX (Black et al., 2022), and later Pythia (Biderman et al., 2023), adopted a partial approach, applying RoPE to only 25% of the dimensions. In contrast, the LLaMA series (Touvron et al., 2023a,b; Grattafiori et al., 2024; Meta AI, 2025) and most of the Qwen series (Bai et al., 2023; Yang et al., 2024; Qwen et al., 2025; Yang et al., 2025a) applied the transformation to all dimensions. Interestingly, the latest Qwen3-Next model moved to a 25% application, a change claimed to improve extrapolation to longer sequences (Qwen Team, 2025). Other models have explored intermediate values: NVIDIA's Nemotron-4-340B (Nvidia et al., 2024) uses 50%, and Microsoft's Phi-2 (Javaheripi et al., 2023) uses 40%.

This wide variance in implementation (ranging from 25% to 100% of dimensions receiving the rotary transformation) highlights a significant gap in the literature. To date, no study has systematically examined how partial RoPE influences model convergence, training dynamics, or efficiency. In practice, this design choice is made inconsistently across model families with no reported ablations for most models, unlike well-studied hyperparameters such as depth or number of attention heads. Moreover, it remains under-documented and not well supported in several pre-training frameworks.

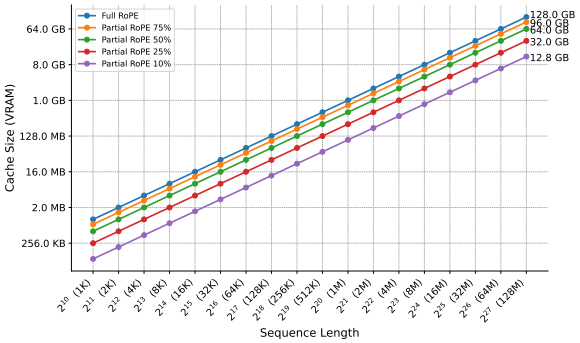


Figure 1: Memory usage of the RoPE sine/cosine cache as a function of sequence length. Partial application (e.g., 10%) drastically reduces RoPE cache size, which becomes critical for very long context windows (especially for edge devices and other resource constrained settings). The exact estimation procedure is presented in App. B.

The question becomes especially relevant for long-context models: as we find, applying RoPE to only a small subset of each head (e.g., 10%) can reduce the memory footprint of the RoPE cache by an order of magnitude, as shown in Fig. 1 which becomes significant at long context windows. A systematic investigation into this parameter can therefore yield principled insights, improve memory efficiency, and guide future design choices.

2 Related Work

RoPE and Partial RoPE. RoPE (Su et al., 2023) has rapidly become a standard method for encoding relative positions in transformer models, due to their simplicity, ability to generalize to longer sequences, and compatibility with various attention mechanisms. Prior research has explored several aspects of RoPE design and integration. For example, some prior work as well as frontier models skip RoPE in certain layers/combine it with sliding attention to improve efficiency without compromising model performance (Kazemnejad et al., 2023; Cohere et al., 2025; Meta AI, 2025). In addition, the choice of rotary embedding base and scaling factors can affect the model's extrapolation ability and convergence (Men et al., 2024; Yang et al., 2025b). Other studies have focused on understanding RoPE's internal working (Barbero et al., 2025) and investigate issues with using RoPE in long context settings with improper precision (Wang et al., 2024). Despite these efforts, the specific question of how many dimensions of the hidden state should undergo rotary transformations has received little systematic attention. To our knowledge, the only discussion on partial RoPE application appears in (Black et al., 2022), which suggested, based on small-scale experiments[1], that a 25% application offered a good trade-off between performance and efficiency. These findings, though limited, have influenced design choices in multiple subsequent models.

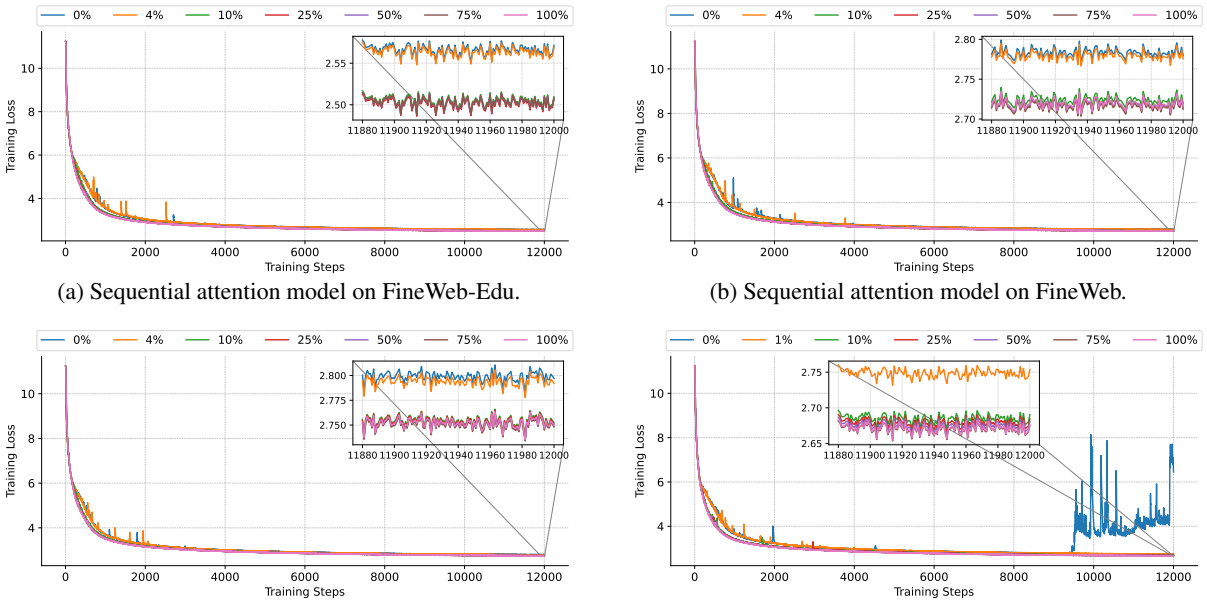
Long Context Models. There has been a continuous push to train models with increasingly long context windows. For example, Google's Gemini 1.5 series boasted a context window of upto 10 million tokens (Google, 2024), while Magic reported preliminary results for 100 million token context windows (Magic, 2024). At these extreme scales, previously negligible implementation details become critical. For instance, RoPE cache, which typically consumes minimal VRAM, can require substantial chunks of memory usage due to linear scaling with sequence length. This challenge is compounded in multi-GPU setups, where either replicating the cache on each device consumes redundant memory, or sharding it introduces communication overhead.

3 Experimental Setup

We pretrain several models from scratch to examine how the fraction of each attention head's hidden state receiving rotary positional embeddings affects model performance. We evaluate several fractions, 0% (NoPE), 10%, 25%, 50%, 75%, and 100% (full RoPE), to observe their effect on loss convergence. Additionally, we include experiments with the minimal feasible application, corresponding to just two channels per head, which represents approximately 1% for Pythia (head dimension 256) and 4% for LLaMA (head dimension 64).

We evaluate two transformer architectures: a sequential attention design, implemented using LLaMA-3.2-1B (Meta, 2024), and a parallel attention design, implemented using Pythia-1B (Biderman et al., 2023). The primary training is conducted on the FineWeb-Edu dataset (Lozhkov et al., 2024), with additional experiments on FineWeb (Penedo et al., 2024) to assess effect of dataset quality. We use the officially released 100B-token subset of each dataset and process all inputs with the Pythia tokenizer. Additional details are presented in App. A.

[1]On inspecting the public logs we were able to infer the authors used the GPT2-Small architecture (124M parameters) and trained on only 82M tokens, in contrast we test multiple architectures at the 1B scale and over 100B tokens



(a) Sequential attention model on FineWeb-Edu.

(b) Sequential attention model on FineWeb.

(c) Sequential attention model on FineWeb (seq. length 1024).

(d) Parallel attention model on FineWeb-Edu.

Figure 2: Training loss trajectories for various Partial RoPE configurations across sequential and parallel attention architectures on FineWeb and FineWeb-Edu datasets. Each subfigure specifies the architecture and dataset used.

4 Partial RoPE Analysis

RQ1: How does the fraction of hidden dimensions receiving RoPE influence model training dynamics? As shown in Fig. 2a, configurations using 10% or more RoPE exhibit nearly identical convergence behavior. By the end of training, two distinct convergence groups emerge: models without positional embeddings or with RoPE applied to only 2 channels (4%) converge to consistently higher final losses, while those with 10% or more RoPE achieve similar and lower final losses. This indicates that applying RoPE to even a modest fraction of hidden dimensions is sufficient to match the convergence performance of full RoPE.

RQ2: How does the quality of pre-training data affect the optimal partial RoPE configuration? To explore this, we repeated the experiments using FineWeb, a dataset of lower quality compared to FineWeb-Edu. FineWeb-Edu is derived from FineWeb by applying an educational quality classifier that filters for content with higher educational quality. The impact of data quality is evident in the higher final loss values across runs (with differences of at least 0.2 points from Figs. 2a and 2b). Nonetheless, similar convergence patterns are observed across both datasets.

RQ3: How does the training sequence length influence the optimal Partial RoPE configuration? To assess the impact of sequence length on the behavior of Partial RoPE, we replicate the experiments from RQ1 using training sequences of 1024 tokens. As shown in Fig. 2c, the model exhibits similar convergence bands across configurations, suggesting that the observed trends are consistent across different sequence lengths.

RQ4: How consistent are the effects of partial RoPE across different transformer block designs (sequential vs. parallel attention)? To investigate this, we trained models following the Pythia-1B architecture with parallel transformer blocks and observed two notable trends. The NoPE configuration (0%) fails to converge (Fig. 2d), a phenomenon we analyze further in Section 5. Excluding the NoPE run, two distinct convergence bands emerge, similar to those seen in sequential attention models, indicating that the overall patterns of partial RoPE performance are largely consistent across different transformer block designs.

Takeaways Across our experiments, distinct patterns emerge for partial RoPE. Gains from increasing the fraction of hidden dimensions that receive RoPE plateau beyond minimal application (10%). Second, these trends are robust across datasets of differing quality: even lower-quality pre-training data exhibits similar convergence patterns, though with higher absolute loss values. Finally, the effects of partial RoPE largely persist across different sequence lengths and transformer block designs, with both sequential and parallel attention models displaying similar convergence bands. Together, these findings suggest that a modest fraction of RoPE

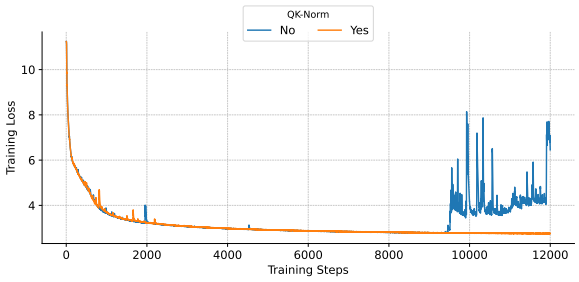


Figure 3: Training loss trajectories for parallel attention models trained on FineWeb-Edu with and without QK-Norm.

(around 10% or higher) is sufficient to achieve training dynamics comparable to full RoPE, and that these observations generalize across data quality and architectural variations.

5 Analyzing Loss Spikes in Parallel Architectures with NoPE

Having established that partial RoPE can achieve comparable convergence, we now turn to analyzing the unrecoverable loss spikes observed in the NoPE configuration under the parallel transformer block architecture. These loss spikes were not observed by Kazemnejad et al. (2023) as they primarily focused on smaller synthetic tasks, where both the model and dataset scales were significantly lower than in our setting, and notably did not involve a parallel architecture. In contrast, our experiments reveal that these loss spikes emerge only after several tens of billions of training tokens, and exclusively in parallel architectures. Given the renewed interest in parallel transformer architectures driven by their efficiency advantages (Cohere et al., 2025), we highlight this phenomenon as an important consideration for future model developers and researchers exploring position-agnostic transformer variants.

To identify the root causes and mitigate these spikes, we systematically investigate the following:

Effect of Random Seeds: We first considered that the spikes could result from an unlucky ordering of training data/model initialization. However, repeating the experiments with multiple random seeds (which change model initialization and training order) consistently produced the same spikes (Fig. 4a), ruling out randomness as the cause.

Effect of Learning Rate: Next, we examined whether the learning rate might be responsible. We tested learning rates an order of magnitude higher and lower than the default (4×10^{-3} & 4×10^{-5}). With a smaller learning rate, training converged to a higher final loss and still exhibited spikes, whereas a larger learning rate led to early divergence (Fig. 4b).

Effect of QK-Norm: Finally, we applied QK-Norm (Henry et al., 2020), a normalization technique known to stabilize training. With QK-Norm, the loss spikes disappear (Fig. 3), suggesting that the underlying cause may be excessively large or spiky gradients and parameter magnitudes (Taylor, 2024; OLMo et al., 2025), both of which this normalization mitigates. Following its application, the NoPE configuration converged to a higher loss band, consistent with patterns observed in other RQs. *Based on this, we recommend that model trainers adopt QK-Norm as a precautionary measure against loss spikes in such configurations or use partial RoPE to prevent the same.*

6 Conclusion

In this work, we presented a systematic empirical study on the impact of partial application of Rotary Positional Embeddings across hidden dimensions in large-scale transformer models. Our experiments demonstrate that even modest fractions of RoPE (10% of dimensions) are sufficient to achieve convergence and final loss comparable to full RoPE, while extremely low fractions or NoPE configurations lead to slower training and, in parallel block architectures, can induce pronounced loss spikes. We further show that these trends are robust across datasets of varying quality and hold for both sequential and parallel transformer architectures that we tested. Additionally, we find that stabilization techniques such as QK-Norm can mitigate the loss spikes observed under NoPE for parallel architectures, however partial RoPE is a far more effective way to do the same.

Collectively, our findings provide actionable guidance for model designers: partial RoPE can significantly reduce computational overhead especially at long context windows without sacrificing convergence, and careful consideration of transformer block design and normalization strategies can prevent instability in extreme configurations. This work highlights the previously underexplored role of partial RoPE in model optimization and lays the groundwork for future studies on efficient positional encoding strategies in large language models.

Limitations

As with any study involving pretraining, the design space is vast and each experiment incurs substantial computational cost. While it is infeasible to explore all possible combinations of architectures, model sizes, and datasets, we carefully select configurations that follow established best practices and evaluate them at scales consistent with prior work, allowing us to make claims we expect to generalize to larger settings.

We leave several directions for future work such as combining partial RoPE with NoPE for additional efficiency gains, studying scaling laws for partial RoPE, and exploring the interaction of partial RoPE with length extrapolation methods due to the substantial computational resources and training time each of these investigations would require.

References

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Jason Phang, Shivanshu Purohit, Hailey Schoelkopf, Dashiell Stander, Tri Songz, Curt Tigges, Benjamin Thérien, and 2 others. 2023. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Federico Barbero, Alex Vitvitskyi, Christos Perivolaropoulos, Razvan Pascanu, and Petar Veličković. 2025. Round and round we go! what makes rotary positional encodings useful? *Preprint*, arXiv:2410.06205.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of the ACL Workshop on Challenges & Perspectives in Creating Large Language Models*.

Team Cohere, :, Aakanksha, Arash Ahmadian, Marwan Ahmed, Jay Alammar, Milad Alizadeh, Yazeed Alnumay, Sophia Althammer, Arkady Arkhangorodsky, Viraat Aryabumi, Dennis Aumiller, Raphaël Avalos, Zahara Aviv, Sammie Bae, Saurabh Baji, Alexandre Barbet, Max Bartolo, Björn Bebensee, and 211 others. 2025. Command a: An enterprise-ready large language model. *Preprint*, arXiv:2504.00698.

Google DeepMind Google, Alphabet. 2024. Our next-generation model: Gemini 1.5. https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. 2020. Query-key normalization for transformers. *Preprint*, arXiv:2010.04245.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, and 1 others. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2023. The impact of positional encoding on length generalization in transformers. *Preprint*, arXiv:2305.19466.

Ilya Loshchilov and Frank Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts. *Preprint*, arXiv:1608.03983.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

Magic. 2024. 100m token context windows. https://magic.dev/blog/100m-token-context-windows.

Xin Men, Mingyu Xu, Bingning Wang, Qingyu Zhang, Hongyu Lin, Xianpei Han, and Weipeng Chen. 2024. Base of rope bounds context length. *Preprint*, arXiv:2405.14591.

Meta. 2024. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

Meta AI. 2025. Llama 4: Multimodal intelligence. https://ai.meta.com/blog/llama-4-multimodal-intelligence/. Accessed: 2025-10-03.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. *Preprint*, arXiv:1710.03740.

Nvidia, :, Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H. Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, Sirshak Das, Ayush Dattagupta, Olivier Delalleau, Leon Derczynski, Yi Dong, Daniel Egert, Ellie Evans, and 64 others. 2024. Nemotron-4 340b technical report. *Preprint*, arXiv:2406.11704.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, and 21 others. 2025. 2 olmo 2 furious. *Preprint*, arXiv:2501.00656.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Qwen Team. 2025. Qwen3-next: Towards ultimate training & inference efficiency. https://qwen.ai/blog?from=research.latest-advancements-list&id=4074cca80393150c248e508aa62983f9cb7d27cd. Accessed: 2025-10-05.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Preprint*, arXiv:2104.09864.

Ross Taylor. 2024. Qk norm and the curious case of logit drift. https://rossjtaylor.com/blog/qk-norm-and-the-curious-case-of-logit-drift/. Accessed: 2025-10-03.

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. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ben Wang and Aran Komatsuzaki. 2021. Gpt-j-6b: A 6 billion parameter autoregressive language model.

Haonan Wang, Qian Liu, Chao Du, Tongyao Zhu, Cunxiao Du, Kenji Kawaguchi, and Tianyu Pang. 2024. When precision meets position: Bfloat16 breaks down rope in long-context training. *arXiv preprint arXiv:2411.13476*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Bowen Yang, Bharat Venkitesh, Dwarak Talupuru, Hangyu Lin, David Cairuz, Phil Blunsom, and Acyr Locatelli. 2025b. Rope to nope and back again: A new hybrid attention strategy. *Preprint*, arXiv:2501.18795.

A Training Details

All experiments are conducted on a cluster of 4 nodes, each equipped with 8 H200 GPUs. For most runs we use a per-GPU micro-batch size of 64 with 2 gradient accumulation steps, resulting in an effective global batch size of 4096 sequences, each containing 2048 tokens. For RQ3 with sequence length 1024, we double the micro-batch size to 128 to keep the training steps identical across runs. Training is implemented using the GPT-NeoX framework (Andonian et al., 2023), selected for its strong support for partial RoPE and our existing familiarity with its ecosystem. Optimization is performed using AdamW (Loshchilov and Hutter, 2019) with an initial learning rate of

4×10^{-4}, a 5% warmup phase, and cosine learning rate decay (Loshchilov and Hutter, 2017), without any dropout. All models are trained in mixed precision (Micikevicius et al., 2018) following common conventions, where forward and backward computations are performed in BF16 while gradient accumulation and inter-GPU reductions are carried out in FP32. The training setup employs pure data parallelism across all GPUs, with no additional forms of model or pipeline parallelism. Each run took between 12-13 hours for completion.

B Computation of RoPE Cache Size

Rotary Positional Embeddings (RoPE) enhance transformer models by encoding positional information without learnable parameters. Instead of storing explicit embedding vectors for each position, RoPE applies a rotational transformation to query and key vectors on-the-fly. To optimize this process, the sine and cosine values required for these rotations are pre-computed and stored in a cache upon model initialization.

The VRAM required to store the RoPE cache is a direct function of three key parameters: the maximum supported sequence length, the dimensionality of each attention head, and the numerical precision used for storage. The total size can be calculated using the following formula:

$$S_{cache} = L_{max} \times D_{head} \times P_{bytes} \quad (1)$$

Where:

- S_{cache} is the total cache size in bytes.
- L_{max} represents the maximum sequence length (or context window) the model is configured to handle.
- D_{head} is the dimensionality of the vector for a single attention head.
- P_{bytes} represents the number of bytes needed to store a single numerical value, determined by the chosen precision (e.g., 4 bytes for 32-bit floating point, FP32). We use FP32 in our calculations, as prior work has shown that lower precisions often lead to instability and divergence during long-context training, issues that are mitigated when using FP32 (Wang et al., 2024).

This linear relationship demonstrates that the cache size scales directly with both the maximum sequence length and the head dimension, a critical consideration when designing models for long-context applications.

For Fig. 1, we use an attention head dimension of 256, which corresponds to the head dimensionality of Pythia-1B. The memory requirements would increase further for larger head dimensions (in practice most current models seem to stick to 128). Our calculations also exclude any effects of memory fragmentation and consider only the raw storage needed for the RoPE sine/cosine cache.

C References for Partial RoPE Usage

Table 1 lists various models that utilize partial RoPE, along with links referencing this configuration in their model settings.

D Additional Results

Figures 4a & 4b showcase the persistence of loss spikes across seeds and learning rates.

E LLM Usage

We use LLMs for help with grammatical corrections/writing of the paper and also as coding assistants. In both cases we independently verify the outputs.

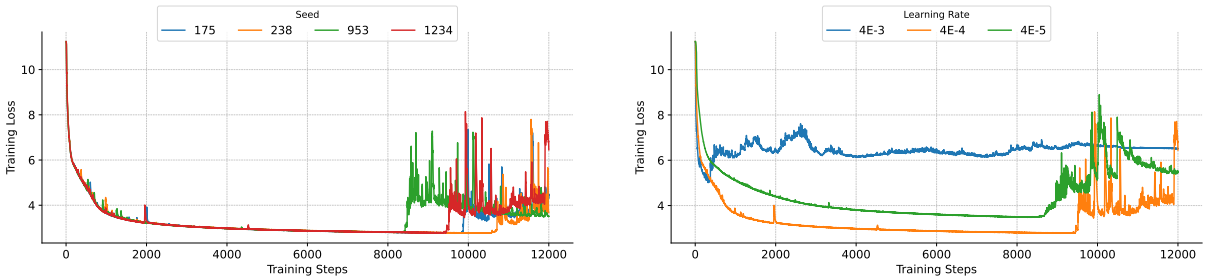
F Artifact Release/Usage

We will publically release all our artifacts (models, optimizer states, intermediate checkpoints, config files, etc.) publically upon acceptance.

Our usage of artifacts such as the pretraining dataset and frameworks is consistent with their intended usage.

Model	URL
EleutherAI/gpt-j-6b	www.hf.co/EleutherAI/gpt-j-6b/blob/main/config.json#L21
EleutherAI/gpt-neox-20b	www.hf.co/EleutherAI/gpt-neox-20b/blob/main/config.json#L19
EleutherAI/pythia-12b	www.hf.co/EleutherAI/pythia-12b/blob/main/config.json#L17
microsoft/phi-2	www.hf.co/microsoft/phi-2/blob/main/config.json#L20
nvidia/Nemotron-4-340B-Base	www.hf.co/nvidia/Nemotron-4-340B-Base/blob/main/model_config.yaml#L34
Qwen/Qwen3-Next-80B-A3B-Instruct	www.hf.co/Qwen/Qwen3-Next-80B-A3B-Instruct/blob/main/config.json#L31

Table 1: Examples of widely used open-weight/open-source models employing **Partial RoPE**, with references indicating this choice—parameter names vary across models due to differences in training frameworks. Note: For Pythia, we show one family member, though all members use the same 25% partial RoPE.



(a) Different random seeds. (b) Different learning rates.

Figure 4: Training loss trajectories for parallel attention models trained on FineWeb-Edu under different experimental settings. The left panel varies random seeds, while the right panel varies learning rates.