FAST GRADIENT COMPUTATION FOR ROPE ATTENTION IN ALMOST LINEAR TIME

Yifang Chen[*] **Jiayan Huo**[†] **Xiaoyu Li**[‡]
Yingyu Liang[§] **Zhenmei Shi**[¶] **Zhao Song**[‖]

ABSTRACT

The Rotary Position Embedding (RoPE) mechanism has become a powerful enhancement to the Transformer architecture, which enables models to capture token relationships when encoding positional information. However, the RoPE mechanisms make the computations of attention mechanisms more complicated, which makes efficient algorithms challenging. Earlier research introduced almost linear time algorithms for the forward computation under specific parameter settings of bounded entries (i.e., in time $n^{1+o(1)}$ where n is the number of input tokens), but has not addressed backward computation. In this work, we develop the first almost linear time algorithm for backward computations in the RoPE-based attention under bounded entries. Our approach builds on recent advancements in fast RoPE attention computations, utilizing a novel combination of the polynomial method and the Fast Fourier Transform. Furthermore, we show that with lower bounds derived from the Strong Exponential Time Hypothesis (SETH), the bounded entry condition is necessary for subquadratic performance.

1 INTRODUCTION

The GPT-o3 (OpenAI, 2024), Llama 3.3 (Llama Team, 2024; AI, 2024), Claude 3.5 (Anthropic, 2024b) are transformed-based Large Language Models (LLMs), have become important tools in natural language processing, which enables applications from machine translation to sentiment analysis. In the Transformer architecture, attention mechanisms, computationally intensive operations, compute token correlations within the sequence (Vaswani et al., 2017). The efficiency of attention computations, both in forward computations and backward gradient computations, directly influenced the scalability and feasibility of training LLMs, especially when the size and input context length of these LLMs continue to grow (Alman & Song, 2024a; 2023). In recent research, Rotating Position Embedding (RoPE) (Su et al., 2024) has become a popular modification to the attention mechanism, and it has enabled models to capture positional relationships between tokens with better expressiveness. The RoPE mechanism has been adopted in state-of-the-art models, such as Llama (Touvron et al., 2023a;b; Llama Team, 2024), Claude (Anthropic, 2024b), Apple's LLMs (Gunter et al., 2024; McKinzie et al., 2024), and many others, but the implementation of RoPE complicates attention computation due to the additional structure imposed by position-dependent rotations (Su et al., 2024). In recent work, Alman & Song (2024b) demonstrated an efficient algorithm for forward computation of RoPE attention in the bounded entry regime. However, backward computation, the process of calculating gradients for model optimization, has been explored less.

Backward computation introduces additional complexity because it requires the evaluation of gradients that involve non-linear transformations of the attention matrix and positional embeddings. In Alman & Song (2023), they present their algorithm to approximate forward computations of fast

[*] yifangc@uchicago.edu. University of Chicago.
[†] jiayanh@arizona.edu. University of Arizona.
[‡] xiaoyu.li2@student.unsw.edu.au. University of New South Wales.
[§] yingyul@hku.hk. The University of Hong Kong. yliang@cs.wisc.edu. University of Wisconsin-Madison.
[¶] zhmeishi@cs.wisc.edu. University of Wisconsin-Madison.
[‖] magic.linuxkde@gmail.com. The Simons Institute for the Theory of Computing at UC Berkeley.

attention with bounded entries using the polynomial methods and low-rank approximation. In Alman & Song (2024c), they propose almost linear time, i.e., $n^{1+o(1)}$ where n is the number of input tokens, an algorithm to compute backward gradients for fast attention with bounded entries. In recent work, Alman & Song (2024b) proposes an efficient algorithm to perform the forward computation of RoPE-based attention using the polynomial methods and Fast Fourier Transform. Therefore, it is natural to raise the key question:

Can backward computations for the RoPE attention match the efficiency of their forward computations in the bounded entry regime?

In this work, we aim to address the question by presenting the first efficient algorithm for backward computation in RoPE attention under the bounded entry. Our main result shows that the backward gradient computations for the RoPE attention match their forward version's efficiency. Therefore, by leveraging our algorithm in approximating backward computations in the RoPE attention with the forward algorithm from Alman & Song (2024b), we will improve the overall time complexity of RoPE attention to almost linear time with bounded entries.

To the best of our knowledge, this is the first work to characterize the fine-grained complexity of backward computations in RoPE attentions, extending prior results on forward computations in RoPE attention (Alman & Song, 2024b). Our contribution can be described as follows.

- We formulated the closed-form gradient for the RoPE attention (see Lemma 4.1) along with its exact time complexity (see Theorem 4.2).
- We derive the almost linear time backward approximation (see Theorem 5.2) for RoPE attention based on the closed-form gradient.
- We show that with lower bounds derived from the SETH, the bounded entry condition is necessary for subquadratic performance (see Theorem 6.1).

Roadmap. In Section 2, we present some relevant papers. In Section 3, we show essential components for the RoPE attention. Section 4 gives the closed form of the RoPE Attention gradient and discusses the exact time complexity to compute it. Section 5 shows the fast computation of the RoPE Attention gradient in almost linear time. Section 6 details the lower bounds of hardness. Finally, Section 7 provides conclusions and avenues for future work.

2 Related Work

Rotary Position Embedding. At a high level, RoPE gives more expressive power to the model in exchange for making the computational problem more complicated. In particular, many prior algorithms, such as the algorithm of Alman & Song (2023), no longer apply to RoPE for fundamental reasons we will discuss. RoPE was proposed by Su et al. (2024) and has been used extensively in large-scale industrial models. Examples which are known to use RoPE include the open-source models released by Meta such as Llama (Touvron et al., 2023a) (see page 3), Llama 2 (Touvron et al., 2023b) (see page 5), Llama 3 Llama Team (2024) (see page 7), and the close-source LLM Claude 3.5 (Anthropic, 2024b) released by Anthropic. Apple also incorporates RoPE into their LLM architecture (see McKinzie et al. (2024), and page 3 of Gunter et al. (2024)).

Fast Attention Computation. The attention mechanism has often been criticized for its quadratic computational complexity concerning context length, a challenge that becomes more pronounced as the sequence length grows in today's LLMs (Achiam et al., 2023; OpenAI, 2024; Llama Team, 2024; AI, 2024; Anthropic, 2024a;b). However, this issue can be addressed using polynomial kernel approximation methods (Aggarwal & Alman, 2022), which facilitate constructing the approximated attention matrix using low-rank approximations. Such techniques enable substantial improvements in computation speed, allowing a single attention layer to perform both training and inference nearly as fast as linear time (Alman & Song, 2023; 2024c). Our research further extends this efficiency to support multi-layer transformer architectures for both training and inference. In addition, these techniques can generalize to advanced attention mechanisms, such as tensor attention, while preserving the almost linear time complexity in both training and evaluation phases (Alman & Song, 2024a; Liang et al., 2024c). Beyond this, alternative theoretical methods also exist. For example,

the conv-basis approach introduced in Liang et al. (2024a) offers another avenue for speeding up attention computation.

Gradient Approximation. Using low-rank approximation to approximate the gradient is a common approach for optimizing the training of transformers by reducing the complexity in the computations, such as Liang et al. (2024b;c); Alman & Song (2024c); Hu et al. (2024). Specifically, Alman & Song (2024c) extends the low-rank approximation technique developed in Alman & Song (2023), which studies the forward computation of attention to approximate the gradient of the attention computation. In Liang et al. (2024b), they further develop the low-rank approximation technique in Alman & Song (2024c) to study multi-layer transformers, showing they can use nearly linear time to approximate the backward computations of multi-layer transformers. On the other hand, Liang et al. (2024c) generalizes the gradient approximation of Alman & Song (2024c) to another direction: they use it to study the training of the tensor version of attention computation that develops from the forward computation as in Alman & Song (2024a). Finally, Hu et al. (2024) leverages the low-rank approximation technique to study the training of Diffusion Transformers (DiTs).

3 Preliminaries on RoPE Attention

Our approximation task can be formalized as follows.

Definition 3.1 (The Approx of the gradient of RoPE Attention Loss Function, $\mathsf{ARAttLGC}(n, d, B, \epsilon)$)**.** *Let $B > 0$ and $\epsilon > 0$ denote two parameters. Given a set of matrices $W_{-(n-1)}, \cdots, W_{-1}, W_0, W_1, \cdots, W_{n-1} \in \mathbb{R}^{d \times d}$ where $\mathrm{supp}(W_i) \subset S$ for all $i \in \{-(n-1), \cdots, -1, 0, 1, \cdots, n-1\}$. Here $S \subseteq [d] \times [d]$ where $|S| = O(d)$. For $i, j \in [n]$, let $\mathsf{W} \in \mathbb{R}^{n^2 \times d^2}$ such that $\mathsf{W}_{i+(j-1)n,*} = \mathrm{vec}(W_{i-j})$. Let $X_1, X_2, Y \in \mathbb{R}^{d \times d}$. Let $\mathsf{X} := X_1 \otimes X_2 \in \mathbb{R}^{d^2 \times d^2}$. We have four $n \times d$ matrices Let A_1, A_2, A_3, E. Let $\mathsf{A} \in \mathbb{R}^{n^2 \times d^2}$ such that A equals to an $n^2 \times d^2$ matrix from $A_1 \otimes A_2$. Assume $\|A_1 X\|_\infty \leq B, \|A_2 X\|_\infty \leq B, \|A_3 Y\|_\infty \leq B, \|W\|_\infty \leq 1$. Assume that all the $\log(n)$ bits model is applied throughout all numbers in matrices. We define $\mathsf{Loss}(\mathsf{X})$ from Def. A.11. Here, we define $\frac{\mathrm{d}\mathsf{Loss}(\mathsf{X})}{\mathrm{d}\mathsf{X}}$ as the loss function gradient. The goal is to output a vector $\widetilde{g} \in \mathbb{R}^{d^4}$ such that*

$$\|\widetilde{g} - \frac{\mathrm{d}\mathsf{Loss}(\mathsf{X})}{\mathrm{d}\mathsf{X}}\|_\infty \leq \epsilon.$$

The formulation of the problem can be found in Section A.6

4 Exact Gradient Computation Time

In this section, we provide the gradient computations of RoPE attentions. In Section 4.1, we formulate the gradient in its closed form. In Section 4.2, we conduct a time complexity analysis on the exact computation of RoPE attention gradients.

4.1 Reformulate the Gradient into Its Closed Form

In this section, we present the closed-form gradient of RoPE attention.

Lemma 4.1 (Gradient Reformulation, $\frac{\mathrm{d}\mathsf{Loss}(x)}{\mathrm{d}x}$, Informal Version of Lemma C.4)**.** *For every $i \in [d^4]$, we choose the following functions*

- *The Normalized Softmax function $s(x)_{j_0} \in \mathbb{R}^n$ (see Definition B.3),*
- *The Error term $\ell(x)_{j_0, i_0} \in \mathbb{R}$ (see Definition B.5),*
- *The Loss term $\mathsf{Loss}(x)_{j_0, i_0} \in \mathbb{R}$ (see Definition B.6),*
- *We define $\beta(x)_{j_0} \in \mathbb{R}^n$ is $\underbrace{A_3 Y}_{n \times d} \underbrace{\ell(x)_{j_0,*}^\top}_{d \times 1}$*

- *We define* $\gamma(x)_{j_0} \in \mathbb{R}^n$ *is* $(\mathrm{diag}(s(x)_{j_0}) - s(x)_{j_0} s(x)_{j_0}^\top)\beta(x)_{j_0}$

Then, we get

$$\frac{\mathrm{d}\mathsf{Loss}(x)}{\mathrm{d}x} = \underbrace{\widetilde{\mathsf{A}}^\top}_{d^4 \times n^2} \mathrm{vec}(\underbrace{\gamma(x)}_{n \times n})$$

Proof. See full proof at Lemma C.4. □

4.2 Time Complexity for Computing the Gradient of RoPE Attention

In this section, we provide the time complexity of computing the exact gradient of RoPE attention.

Theorem 4.2 (RoPE attention gradient computation time complexity)**.** *We define three* $n \times d$ *input matrices as* A_1, A_2, A_3*, and the* $n \times d$ *approximated attention computation matrix as* E*. We define several input fixed matrices as* $X_1, X_2, Y \in \mathbb{R}^{d \times d}$*. We define* $\mathsf{X} = X_1 \otimes X_2$*,* $\mathsf{A} = A_1 \otimes A_2$*. We define* $x := \mathrm{vec}(\mathsf{X})$ *and try to get the Loss function gradient. Let* $g := \frac{\mathrm{d}\mathsf{Loss}(X_1, X_2)}{\mathrm{d}x}$ *where* $\mathsf{Loss}(X_1, X_2)$ *from Def. 3.1. Then, it costs* $O(\mathcal{T}_{\mathrm{mat}}(n, d, d) + \mathcal{T}_{\mathrm{mat}}(n, d, n))$ *time to get the gradient* $g \in \mathbb{R}^{d^4}$*.*

Proof. See full proof at Theorem C.8. □

Note that $O(\mathcal{T}_{\mathrm{mat}}(n, d, d) + \mathcal{T}_{\mathrm{mat}}(n, d, n)) \geq \Omega(n^2)$. Thus, the naive RoPE attention gradient computation is a complexity obstacle in practice, as discussed in Section 1. Based on the closed formulation in Lemma 4.1, we derive our acceleration method, which will be introduced in the following section.

5 Compute RoPE Attention Gradient in Almost Linear Time

In this section, we present our main result. With the low-rank approximation, we can approximate the RoPE gradient computations in almost linear time.

In Section 5.1, we discuss the techniques we used to develop the almost linear time algorithm. In Section 5.2, we present our main results, which compute RoPE Attention gradient in almost linear time.

5.1 Technique Overview

In recent work Alman & Song (2024b), they present an almost linear-time algorithm to compute forward computations of RoPE attention as follows.

Lemma 5.1 (Theorem 1.3 in Alman & Song (2024b))**.** *Suppose* $d = O(\log n)$ *and* $B = o(\sqrt{\log n})$*. There is an* $n^{1+o(1)}$ *time algorithm to approximate* ArAttC *up to* $\epsilon = 1/\operatorname{poly}(n)$ *additive error.*

Recall that the closed form gradient of RoPE attention is $\frac{\mathrm{d}\mathsf{Loss}(x)}{\mathrm{d}x} = \widetilde{\mathsf{A}}^\top \mathrm{vec}(\gamma(x))$ from Lemma 4.1. We need to show $\gamma(x)$ can be low-rank approximated in $O(n^{1+o(1)})$ time with $1/\operatorname{poly}(n)$ error.

To low rank approximate $\gamma(x)$, we use the strategy to split $\gamma(x)$ into two terms, $\gamma_1(x)$ and $\gamma_2(x)$, and run the approximation separately. From Lemma 4.1, $\gamma(x)_{j_0} \in \mathbb{R}^n$ is $(\mathrm{diag}(s(x)_{j_0}) - s(x)_{j_0} s(x)_{j_0}^\top)\beta(x)_{j_0}$. We define $\gamma_1(x)_{j_0} = \mathrm{diag}(s(x)_{j_0})\beta(x)_{j_0}$and $\gamma_2(x)_{j_0} = s(x)_{j_0} s(x)_{j_0}^\top \beta(x)_{j_0}$; thus, we can have $\gamma(x) = \gamma_1(x) - \gamma_2(x)$.

In the definitions of $\gamma_1(x)$ and $\gamma_2(x)$ provided above, they both contain $s(x)$ and $\beta(x)$. In order to find the almost linear time algorithm of $\gamma(x)$, we need to first show that there exists $O(n^{1+o(1)})$ time complexity approximation for $s(x)$ and $\beta(x)$ with $\epsilon/\operatorname{poly}(n)$ error first. From Lemma 4.1, we have $\beta(x)_{j_0} \in \mathbb{R}^n$ is $A_3 Y \ell(x)_{j_0,*}^\top$. Based on the $\beta(x)$ definition, we need to show $\ell(x)$ can be approximated in almost linear time first.

Overall, to develop the $O(n^{1+o(1)})$ time complexity algorithm to compute RoPE gradients with $\epsilon/\operatorname{poly}(n)$ error, we need to prove the existence of almost linear time algorithms for $s(x), \ell(x), \gamma(x)$, and $\beta(x)$ with low rank approximation.

5.2 Fast Computation in Almost Linear Time

Based on Section 5.1, we have proved the almost linear time approximation of $s(x), \ell(x), \gamma(x)$, and $\beta(x)$ in Lemma D.1, D.2, D.3, D.5, and D.4. We are now ready to show our main result, which is to approximate RoPE gradient computation in almost linear time.

Theorem 5.2 (Main result, Low Rank Approximate RoPE Attention Gradient)**.** *Assuming the entries of A_1, A_2, X_1, X_2, Y, E are represented using $O(\log n)$ bits, there is an $n^{1+o(1)}$ time algorithm to solve $\mathsf{AAttLGC}(n, d = O(\log n), B = o(\sqrt{\log n}))$, from Def. 3.1, with the accuracy upper bounded by $\frac{1}{\mathrm{poly}(n)}$. To be more specific, a gradient vector $\widetilde{g} \in \mathbb{R}^{d^4}$ comes out of our algorithm where $\| \frac{\mathrm{d}\mathsf{Loss}}{\mathrm{d}x} - \widetilde{g} \|_\infty \leq \frac{1}{\mathrm{poly}(n)}$.*

Proof Sketch. By Lemma D.5 and Lemma D.4, there exist matrices $\gamma_1(x)$ and $\gamma_2(x)$ such that $\gamma(x) = \gamma_1(x) - \gamma_2(x)$. We assume these lemmas follow from the low-rank approximations in Lemmas D.1–D.3, allowing us to write $\widetilde{\gamma}_1(x) = U_3 V_3^\top$ and $\widetilde{\gamma}_2(x) = U_4 V_4^\top$ in $n^{1+o(1)}$ time. From Lemma 4.1, the reformulated gradient is $\frac{\mathrm{d}\mathsf{Loss}(x)}{\mathrm{d}x} = \widetilde{\mathsf{A}}^\top \mathrm{vec}(\gamma(x))$, and hence the total running time remains $n^{1+o(1)}$. To bound the error, we show that

$$\begin{aligned} \| \frac{\mathrm{d}\mathsf{Loss}(x)}{\mathrm{d}x} - \widetilde{g} \|_\infty &= \| \widetilde{\mathsf{A}}^\top (\mathrm{vec}(\gamma(x)) - \mathrm{vec}(\widetilde{\gamma}(x))) \|_\infty \\ &\leq \| \widetilde{\mathsf{A}}^\top \|_\infty \| \gamma(x) - \widetilde{\gamma}(x) \|_\infty \\ &\leq \epsilon / \mathrm{poly}(n), \end{aligned}$$

This completes the proof. (See full proof at Theorem D.6) □

6 Hardness

In this section, we provide the lower bound results to compute the gradient of RoPE attention. The hardness result shows that under the widely accepted SETH, the bounded entries condition is necessary for achieving subquadratic runtime.

Theorem 6.1 (Lower bound, informal version of Theorem E.1)**.** *Assuming SETH, for any $q > 0$, for the $\mathsf{ARAttLGC}(n, d = O(\log n), B = \omega(\sqrt{\log n}))$, there does not exist an algorithm which can be executed in time $O(n^{2-q})$ based on Def. 3.1.*

Proof. See the full proof at Theorem E.1. □

In Theorem 6.1, we show that under the Strong Exponential Time Hypothesis (SETH) (see Hypothesis A.4), computing the gradient of RoPE attention remains computationally hard. Specifically, for any constant $q > 0$, no algorithm can compute the gradient in time $O(n^{2-q})$ when $d = O(\log n)$ and $B = \omega(\sqrt{\log n})$. This result establishes a lower bound that fundamentally limits the efficiency of gradient computation for RoPE attention.

7 Conclusion

This paper presents the first efficient backward gradient computation, assuming bounded entries for the RoPE-based attention mechanism. We achieve almost linear time complexity by leveraging polynomial methods and the Fast Fourier Transform, making the forward and backward computations comparably efficient. Additionally, we demonstrate that conditions exist under which performance better than quadratic can be realized, consistent with the lower bounds suggested by the Strong Exponential Time Hypothesis (SETH).

These findings not only improve the computational efficiency of RoPE-based attention mechanisms but also provide a foundation for exploring sub-gradient computations in other advanced attention variants of neural networks. This work highlights the connection between algorithm design and computational complexity theory, unveiling new possibilities for the development of large transformer models. Future research could extend these results to cases involving unbounded entries and assess the real-world implications of these theoretical advancements for large language models.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Amol Aggarwal and Josh Alman. Optimal-degree polynomial approximations for exponentials and gaussian kernel density estimation. In *Proceedings of the 37th Computational Complexity Conference*, pp. 1–23, 2022.

Meta AI. Introducing meta llama 3: The most capable openly available llm to date, 2024. https://ai.meta.com/blog/meta-llama-3/.

Josh Alman and Zhao Song. Fast attention requires bounded entries. *Advances in Neural Information Processing Systems*, 36, 2023.

Josh Alman and Zhao Song. How to capture higher-order correlations? generalizing matrix softmax attention to kronecker computation. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=v0zNCwwkaV.

Josh Alman and Zhao Song. Fast rope attention: Combining the polynomial method and fast fourier transform. *manuscript*, 2024b.

Josh Alman and Zhao Song. The fine-grained complexity of gradient computation for training large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024c. URL https://openreview.net/forum?id=up4tWnwRol.

Anthropic. Claude 3.5 sonnet, 2024a. URL https://www.anthropic.com/news/claude-3-5-sonnet.

Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024b. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Markus Bläser. Fast matrix multiplication. *Theory of Computing*, pp. 1–60, 2013.

Peter Bürgisser, Michael Clausen, and Mohammad Amin Shokrollahi. Algebraic complexity theory. *Grundlehren der mathematischen Wissenschaften*, 1997.

Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as cnf-sat. *ACM Transactions on Algorithms (TALG)*, 12(3):1–24, 2016.

Google Gemini. Gemini 1.5 pro updates, 1.5 flash debut and 2 new gemma models. https://blog.google/technology/developers/gemini-gemma-developer-updates-may-2024/, 2024. Accessed: May 15.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, et al. Apple intelligence foundation language models. *arXiv preprint arXiv:2407.21075*, 2024.

Jerry Yao-Chieh Hu, Weimin Wu, Zhao Song, and Han Liu. On statistical rates and provably efficient criteria of latent diffusion transformers (dits). *arXiv preprint arXiv:2407.01079*, 2024.

Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

Yingyu Liang, Heshan Liu, Zhenmei Shi, Zhao Song, and Junze Yin. Conv-basis: A new paradigm for efficient attention inference and gradient computation in transformers. *arXiv preprint arXiv:2405.05219*, 2024a.

Yingyu Liang, Zhizhou Sha, Zhenmei Shi, Zhao Song, and Yufa Zhou. Multi-layer transformers gradient can be approximated in almost linear time. *arXiv preprint arXiv:2408.13233*, 2024b.

Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Tensor attention training: Provably efficient learning of higher-order transformers. *arXiv preprint arXiv:2405.16411*, 2024c.

AI @ Meta Llama Team. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, et al. Mm1: Methods, analysis & insights from multimodal llm pre-training. *arXiv preprint arXiv:2403.09611*, 2024.

OpenAI. Introducing openai o1-preview. `https://openai.com/index/introducing-openai-o1-preview/`, 2024. Accessed: September 12.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

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

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

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

Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the international congress of mathematicians: Rio de janeiro 2018*, pp. 3447–3487. World Scientific, 2018.

Appendix

A PRELIMINARY

In Section A.2, we talk about polynomial approximation of the exponential function. In Section A.3, we talk about the time complexity of matrix multiplications, setting up the framework for analyzing efficiency in attention computation. In Section A.4, we talk about the Strong Exponential Time Hypothesis (SETH). In Section A.5, we talk about mathematical properties and tricks, such as the tensor trick and row-wise Kronecker products, which enable efficient matrix-vector operations. In Section A.6, we show the formulation of our RoPE attention optimization task. In Section A.7, we show the construction of the Loss function.

A.1 NOTATION

For $n \in \mathbb{Z}^+ \cup \{0\}$, for set $\{1, 2, \cdots, n\}$, we denote the set by using the notation $[n]$. Here, we define the concept of nearly linear time when the time is $O(n \log n)$. We introduce the concept of almost linear time when time is $O(n^{1+o(1)})$. Given a as any vector, we say the diagonal matrix of c is $\mathrm{diag}(c)$ where c_i means the i, i-th entry in the matrix $\mathrm{diag}(c)$. For any matrix, we denote the support of the matrix using the notation supp, that is, the set of entries where the matrix is nonzero. $B^\top$ is defined as $(B^\top)_{i,j} := B_{j,i}$. Suppose there are two vectors c, d of the same length. We denote the entry-wise multiplication using the notation $c \circ d$; that is, the i-th entry in that vector is $c_i d_i$. To denote the Frobenius norm, for any matrix B, we denote it as $\|B\|_F := \sqrt{\sum_{i,j} B_{i,j}^2}$; to denote the maximum norm of matrix B, we use $\|B\|_\infty := \max_{i,j} |B_{i,j}|$. Suppose there are two matrices C, D of the same dimensions. We represent the Hadamard product or the entry-wise multiplication by using the notation $C \circ D$, that is, (i, j)-th entry of the matrix is $C_{i,j} \cdot D_{i,j}$. Let $C \in \mathbb{R}^{n_0 \times m_0}$ and $D \in \mathbb{R}^{n_1 \times m_1}$. We define $C \otimes D$ is an $n_0 n_1 \times m_0 m_1$ matrix, where $(C \otimes D)_{(j_0-1)n_1+j_1,(i_0-1)m_2+i_1}$ is equal to $C_{j_0,i_0} D_{j_1,i_1}$ for any $j_0 \in [n_0], i_0 \in [m_0], j_1 \in [n_1], i_1 \in [m_1]$.

A.2 POLYNOMIAL APPROXIMATION OF EXPONENTIAL

Here, we will explain a technical tool for controlling the error dependence of our approximate algorithm. In particular, we will use the following optimal-degree polynomial approximation of the exponential function.

Lemma A.1 (Aggarwal & Alman (2022))**.** *Let $B > 1$ and suppose ϵ in $(0, 0.1)$. We can have P, which has input as a scalar and output as a scalar of degree g. g is defined as $\Theta\left(\max\left\{\log(1/\epsilon)/(\log(\log(1/\epsilon)/B)), B\right\}\right)$ such that for all $x \in [0, B]$, we can get*

$$|P(x) - \exp(x)| < \epsilon.$$

Because P's coefficients are rational values with numerators and denominators represented using integers of $\mathrm{poly}(g)$-bit size and these coefficients can be determined in $\mathrm{poly}(g)$ time, we can calculate P in an efficient way.

A.3 TIME COMPLEXITY OF MULTIPLICATIONS

Matrix multiplication is a fundamental operation in many algorithms, and understanding its time complexity is essential for analyzing computational efficiency. Here, we introduce the time complexity of matrix multiplications.

Definition A.2. *We suppose n_1, n_2, n_3, denote any three positive integers. We define $A \in \mathbb{R}^{n_1 \times n_2}$ and $B \in \mathbb{R}^{n_2 \times n_3}$. It costs $\mathcal{T}_{\mathrm{mat}}(n_1, n_2, n_3)$ time to perform AB.*

To further analyze the structure of matrix multiplication time complexity, we rely on a well-known fact from prior research Bürgisser et al. (1997); Bläser (2013). This fact provides equivalences between different permutations of matrix dimensions.

Fact A.3. *We suppose n_1, n_2, n_3, denote any three positive integers. $\mathcal{T}_{\mathrm{mat}}(n_1, n_2, n_3) = O(\mathcal{T}_{\mathrm{mat}}(n_1, n_3, n_2)) = O(\mathcal{T}_{\mathrm{mat}}(n_2, n_1, n_3)) = O(\mathcal{T}_{\mathrm{mat}}(n_2, n_3, n_1)) = O(\mathcal{T}_{\mathrm{mat}}(n_3, n_1, n_2)) = O(\mathcal{T}_{\mathrm{mat}}(n_3, n_2, n_1))$.*

A.4 SETH HYPOTHESIS

Now, we introduce a fundamental theoretical assumption underpinning many of the results presented in this paper: the Strong Exponential Time Hypothesis (SETH). This hypothesis serves as a cornerstone for establishing the hardness of various computational problems.

Our results are built on the common conjecture. Impagliazzo & Paturi (2001) introduce the Strong Exponential Time Hypothesis (SETH) as a stronger form of the $\mathsf{P} \neq \mathsf{NP}$ conjecture. It suggests that our current best SAT algorithms are optimal and is a popular conjecture for proving fine-grained lower bounds for a wide variety of algorithmic problems Cygan et al. (2016); Williams (2018).

Hypothesis A.4 (SETH). *$\forall \epsilon > 0$, $\exists k \in \mathbb{Z}^+$ and k greater or equal to 3 such that, even when utilizing randomized algorithms, within the time of $O(2^{(1-\epsilon)n})$, we cannot solve k-SAT problems with n variables.*

A.5 BASIC FACTS

In this section, we present several basic facts that are used throughout the paper to develop the proof of our main results. These fundamental properties enable efficient computations of vectors and matrices products in our paper.

Here, we first introduce the facts about row-wise Kronecker products.

Fact A.5 (Row-wise Kronecker product)**.** *Let $U_1, V_1 \in \mathbb{R}^{n \times k_1}$. Let $U_2, V_2 \in \mathbb{R}^{n \times k_2}$. Then we have*

$$(U_1 V_1^\top) \circ (U_2 V_2^\top) = (U_1 \oslash U_2)(V_1 \oslash V_2)^\top$$

Here, given $U_1 \in \mathbb{R}^{n \times k_1}$ and $U_2 \in \mathbb{R}^{n \times k_2}$, we define the row-wise Kronecker product as $U_1 \oslash U_2 \in \mathbb{R}^{n \times k_1 k_2}$. That is, $(U_1 \oslash U_2)_{i, l_1 + (l_2 - 1) k_1} := (U_1)_{i,l_1} U_{i,l_2}$ for all $i \in [n]$, $l_1 \in [k_1]$ and $l_2 \in [k_2]$

To simplify the computation of certain matrix operations, we can use a technique known as the tensor trick, which reformulates matrix products into operations involving vectorized representations and Kronecker products.

Fact A.6 (Tensor trick)**.** *Let $X \in \mathbb{R}^{d \times d}$. Let $x \in \mathbb{R}^{d^2}$ be the vectorization of X. Let there be two $n \times d$ matrices A_1, A_2, and we define $\mathsf{A} = A_1 \otimes A_2$. Then, we can get $\mathrm{vec}(A_1 X A_2^\top) = \mathsf{A}x$.*

Given the above tensor trick fact, we can derive additional properties that extend its applicability to exponential operations on matrices. These properties can help us compute the exponential of matrix products efficiently. The properties are presented below.

Fact A.7. *Let there be two $n \times d$ matrices A_1, A_2, and we define $\mathsf{A} = A_1 \otimes A_2$. Let $X \in \mathbb{R}^{d \times d}$. Let $\mathsf{A}_{j_0} \in \mathbb{R}^{n \times d^2}$ be a block of A. We introduce $x \in \mathbb{R}^{d^2}$ as the vectorization of X. Thus, we get*

- $(\exp(A_1 X A_2^\top)_{j_0,*})^\top = \exp(\mathsf{A}_{j_0} x)$
- $\mathrm{vec}(\exp(A_1 X A_2^\top)) = \exp(\mathsf{A}x)$,

For the j_0-th row of $\exp(A_1 X A_2^\top) \in \mathbb{R}^{n \times n}$, we use the notation $\exp(A_1 X A_2^\top)_{j_0,*}$.

Proof. From Lemma and Def. A.6, we are able to prove this fact. We omit the details here since the proof is straightforward. □

Fact A.8. *We suppose there are three vectors of n dimension x, y, z. Thus, we get*

- $\langle x \circ y, z \rangle = x^\top \mathrm{diag}(y) z$.
- $\langle x, y \rangle = \langle x \circ y, \mathbf{1}_n \rangle$.

Next, we introduce some important properties of inner products that help us to reshape the equations in the proofs.

Fact A.9 (Inner Products)**.** *We suppose $n \in \mathbb{Z}^+$, and we suppose the n dimension vectors a, b, c and a scalar d. Then, we have*

- $\langle a, b \rangle = \langle a \circ b, \mathbf{1}_n \rangle$.
- $\langle da, b \rangle = d\langle a, b \rangle = \langle a, db \rangle = d\langle b, a \rangle$.
- $\langle a + c, b \rangle = \langle a, b \rangle + \langle c, b \rangle$.
- $\langle a, b \rangle = a^\top b$.
- $\langle a \circ c, b \rangle = \langle a, b \circ c \rangle$.
- $\langle a \circ b, c \rangle = b^\top \operatorname{diag}(a) c$

A.6 Problem Definition

Let n be the number of input tokens, and let d be the hidden/feature dimensions. We state the generalization of the standard RoPE attention from Alman & Song (2024b).

Definition A.10 (A General Approximate RoPE Attention Computation, ARAttC, Definition 1.1 in Alman & Song (2024b))**.** *Let $B > 0$ and $\epsilon > 0$ denote two parameters. Given a set of matrices $W_{-(n-1)}, \cdots, W_{-1}, W_0, W_1, \cdots, W_{n-1} \in \mathbb{R}^{d \times d}$ where $\operatorname{supp}(W_i) \subset S$ for all $i \in \{-(n-1), \cdots, -1, 0, 1, \cdots, n-1\}$. Here $S \subseteq [d] \times [d]$ where $|S| = O(d)$. Given three $n \times d$ matrices Q, K, V with the guarantee that $\|Q\|_\infty, \|K\|_\infty, \|V\|_\infty \leq B$ and $\|W\|_\infty \leq 1$. We define matrix $A \in \mathbb{R}^{n \times n}$ as, for $i, j \in [n]$,*

$$A_{i,j} := \exp(Q_{i,*} W_{i-j} K_{j,*}^\top / d).$$

We define $D := \operatorname{diag}(A \mathbf{1}_n)$. The goal of General Approximate RoPE Attention Computation is to output a matrix $T \in \mathbb{R}^{n \times d}$ such that $\|T - \mathsf{ARAttC}\|_\infty \leq \epsilon$ is small, where $\mathsf{ARAttC} := D^{-1} A V$. For matrix M, we use $\|M\|_\infty := \max_{i,j} |M_{i,j}|$. Note that the $1/d$ factor inside $\exp$ in the definition of A is a normalization factor.

Our focus is to find weights to fit the attention to a desired output. Let $Q := A_1 X_1$, $K := A_2 X_2$, and $V := A_3 Y$. We use X_1, X_2, and X_3 to represent the weights W_Q, W_K and W_V, respectively. We use A_1, A_2, and A_3 to replace the input matrix to handle the more general settings such as cross attention. Then, the attention matrix is as follows.

$$\begin{aligned} A(X_1, X_2)_{i,j} :=& \exp((A_1 X_1)_{i,*} W_{i-j} (A_2 X_2)_{j,*}^\top / d) \\ =& \exp(A_{1,i,*} X_1 W_{i-j} X_2^\top A_{2,j,*}^\top / d). \end{aligned}$$

We define $w_{i-j} := \operatorname{vec}(W_{i-j}) \in \mathbb{R}^{d^2}$ and define W such that $\mathsf{W}_{j_0,*}$ is an $1 \times d^2$ block and $\mathsf{W}_{i+(j-1)n} := w_{i-j}^\top$. Here, let $\mathsf{A} := A_1 \otimes A_2 \in \mathbb{R}^{n^2 \times d^2}$ and $\mathsf{X} := X_1 \otimes X_2 \in \mathbb{R}^{d^2 \times d^2}$. We can show that

$$\begin{aligned} & A_{1,i,*} X_1 W_{i-j} X_2^\top \mathsf{A}_{2,j,*}^\top \\ =& (A_{1,i,*} \otimes A_{2,j,*})(X_1 \otimes X_2) \operatorname{vec}(W_{i-j}) \\ =& \mathsf{A}_{i+(j-1)n,*} \mathsf{X} w_{i-j}, \end{aligned}$$

where the first step uses the tensor trick, and the second step uses the definitions of w_{i-j}, A, and X. Thus we can reformulate the attention matrix A as, for $i, j \in [n]$

$$A(\mathsf{X})_{i,j} = \exp(\underbrace{\mathsf{A}_{i+(j-1)n,*}}_{1 \times d^2} \underbrace{\mathsf{X}}_{d^2 \times d^2} \underbrace{w_{i-j}/d}_{d^2 \times 1}).$$

Using the tensor trick again, we have

$$\begin{aligned} A(\mathsf{X})_{i,j} =& \exp(\underbrace{(\mathsf{A}_{i+(j-1)n,*} \otimes w_{i-j}^\top)}_{1 \times d^4} \underbrace{\operatorname{vec}(\mathsf{X})/d}_{d^4 \times 1}) \\ =& \exp(\underbrace{(\mathsf{A}_{i+(j-1)n,*} \otimes \mathsf{W}_{i+(j-1)n,*})}_{1 \times d^4} \underbrace{\operatorname{vec}(\mathsf{X})/d}_{d^4 \times 1}). \end{aligned}$$

Hence, by definition of row-wise Kronecker product, we have

$$\mathrm{vec}(A(\mathsf{X})) = \exp(\underbrace{(\mathsf{A} \oslash \mathsf{W})}_{n^2 \times d^4} \underbrace{\mathrm{vec}(\mathsf{X})/d}_{d^4 \times 1}).$$

We define the matrix $D(\mathsf{X}) \in \mathbb{R}^{n \times n}$ as

$$D(\mathsf{X}) = \mathrm{diag}(\underbrace{A(\mathsf{X})}_{n \times n} \underbrace{\mathbf{1}_n}_{n \times 1}).$$

Then, the optimization problem in the context of RoPE attention computation is described as follows:

Definition A.11 (Optimize RoPE Attention)**.** *Let $B > 0$ and $\epsilon > 0$ denote two parameters. Given a set of matrices $W_{-(n-1)}, \cdots, W_{-1}, W_0, W_1, \cdots, W_{n-1} \in \mathbb{R}^{d \times d}$ where $\mathrm{supp}(W_i) \subset S$ for all $i \in \{-(n-1), \cdots, -1, 0, 1, \cdots, n-1\}$. Here $S \subseteq [d] \times [d]$ where $|S| = O(d)$. For $i, j \in [n]$, let $\mathsf{W} \in \mathbb{R}^{n^2 \times d^2}$ such that $\mathsf{W}_{i+(j-1)n,*} = \mathrm{vec}(W_{i-j})$. Here, we suppose four $n \times d$ matrices A_1, A_2, A_3, E, and we have three $d \times d$ matrices X_1, X_2, Y. Let $\mathsf{X} := X_1 \otimes X_2 \in \mathbb{R}^{d^2 \times d^2}$. We define the matrix $A(\mathsf{X}) \in \mathbb{R}^{n \times n}$ as the matrix representation of $\exp(\underbrace{(\mathsf{A} \oslash \mathsf{W})}_{n^2 \times d^4} \underbrace{\mathrm{vec}(\mathsf{X})/d}_{d^4 \times 1})$ and the $n \times n$ matrix $D(\mathsf{X}) := \mathrm{diag}(\underbrace{A(\mathsf{X})}_{n \times n} \underbrace{\mathbf{1}_n}_{n \times 1})$. The RoPE attention optimization problem $\min_{\mathsf{X} \in \mathbb{R}^{d^2 \times d^2}} \mathsf{Loss}(\mathsf{X})$ is formulated as follows:*

$$\min_{\mathsf{X} \in \mathbb{R}^{d^2 \times d^2}} 0.5 \| D(\mathsf{X})^{-1} A(\mathsf{X}) A_3 Y - E \|_F^2.$$

Note that we are able to get the gradient computation of Loss with respect to X_1 or X_2 based on the chain rule because

$$\begin{aligned} \frac{\mathrm{d}\mathsf{Loss}(X_1, X_2)}{\mathrm{d}X_1} &= \frac{\mathrm{d}\mathsf{Loss}(\mathsf{X})}{\mathrm{d}\mathsf{X}} \frac{\mathrm{d}\mathsf{X}}{\mathrm{d}X_1} \\ &= \frac{\mathrm{d}\mathsf{Loss}(\mathsf{X})}{\mathrm{d}\mathsf{X}} \frac{\mathrm{d}(X_1 \otimes X_2)}{\mathrm{d}X_1} \\ &= \frac{\mathrm{d}\mathsf{Loss}(\mathsf{X})}{\mathrm{d}\mathsf{X}} (I_{d \times d} \otimes X_2). \end{aligned}$$

Our approximation task can be formalized as follows.

Definition A.12 (The Approx of the gradient of RoPE Attention Loss Function, $\mathsf{ARAttLGC}(n, d, B, \epsilon)$)**.** *Let $B > 0$ and $\epsilon > 0$ denote two parameters. Given a set of matrices $W_{-(n-1)}, \cdots, W_{-1}, W_0, W_1, \cdots, W_{n-1} \in \mathbb{R}^{d \times d}$ where $\mathrm{supp}(W_i) \subset S$ for all $i \in \{-(n-1), \cdots, -1, 0, 1, \cdots, n-1\}$. Here $S \subseteq [d] \times [d]$ where $|S| = O(d)$. For $i, j \in [n]$, let $\mathsf{W} \in \mathbb{R}^{n^2 \times d^2}$ such that $\mathsf{W}_{i+(j-1)n,*} = \mathrm{vec}(W_{i-j})$. Let $X_1, X_2, Y \in \mathbb{R}^{d \times d}$. Let $\mathsf{X} := X_1 \otimes X_2 \in \mathbb{R}^{d^2 \times d^2}$. We have four $n \times d$ matrices Let A_1, A_2, A_3, E. Let $\mathsf{A} \in \mathbb{R}^{n^2 \times d^2}$ such that A equals to an $n^2 \times d^2$ matrix from $A_1 \otimes A_2$. Assume $\|A_1 X\|_\infty \leq B, \|A_2 X\|_\infty \leq B, \|A_3 Y\|_\infty \leq B, \|W\|_\infty \leq 1$. Assume that all the $\log(n)$ bits model is applied throughout all numbers in matrices. We define $\mathsf{Loss}(\mathsf{X})$ from Def. A.11. Here, we define $\frac{\mathrm{d}\mathsf{Loss}(\mathsf{X})}{\mathrm{d}\mathsf{X}}$ as the loss function gradient. Then, our target is to output a vector $\widetilde{g} \in \mathbb{R}^{d^4}$ satisfying:*

$$\| \widetilde{g} - \frac{\mathrm{d}\mathsf{Loss}(\mathsf{X})}{\mathrm{d}\mathsf{X}} \|_\infty \leq \epsilon.$$

A.7 Reformulation of the Loss Function

In this section, we are able to reformulate and simplify the loss function based on the definitions provided in Section B. This reformulation provides a structured representation of the loss in terms of its components, using the tensor trick to simplify computations and facilitate analysis.

The following lemma formalizes this reformulation, consolidating the expressions for the loss function and connecting its components:

Lemma A.13 (Loss Function Formulation)**.** *Given three $n \times d$ input sequence matrices A_1, A_2, and A_3, we define $\mathsf{A} = A_1 \otimes A_2 \in \mathbb{R}^{n^2 \times d^2}$ and $\mathsf{X} = X_1 \otimes X_2 \in \mathbb{R}^{d^2 \times d^2}$, where $\otimes$ denotes Kronecker product. Given W is a $n^2 \times d^2$ matrix, we define $\widetilde{A} = A \oslash W$, where $\oslash$ is the row-wise Kronecker product from Fact A.5. Let $j_0 \in [n]$, we define $\widetilde{A}_{j_0} \in \mathbb{R}^{n \times d^2}$ be a block of size $n \times d^2$ from $\widetilde{A}$. Let $E \in \mathbb{R}^{n \times d}$ be a matrix, for $j_0 \in [n]$ and $i_0 \in [d]$, we define E_{j_0,i_0} as the (j_0, i_0)-th entry of the matrix E. We use* Loss *function from Definition A.11. Based on Def. B.6, for j_0 in the set $[n]$ and i_0 in the set $[d]$, we get $\mathsf{Loss}(\mathsf{X})_{j_0,i_0}$. Then, we have*

$$\mathsf{Loss}(\mathsf{X}) = \sum_{j_0 \in [n]} \sum_{i_0 \in [d]} \mathsf{Loss}(x)_{j_0,i_0}.$$

Proof. We present the reformulation of the Loss Function using the tensor trick as follows.

$$\begin{aligned}
\mathsf{Loss}(\mathsf{X}) = & 0.5 \| D(\mathsf{X})^{-1} A(\mathsf{X}) A_3 Y - E \|_F^2 \\
= & \sum_{j_0=1}^{n} \sum_{i_0=1}^{d} 0.5 \cdot (\langle \langle \exp(\widetilde{\mathsf{A}}_{j_0} x), \mathbf{1}_n \rangle^{-1} \\
& \exp(\widetilde{\mathsf{A}}_{j_0} x), A_3 Y_{*,i_0} \rangle - E_{j_0,i_0})^2 \\
= & \sum_{j_0=1}^{n} \sum_{i_0=1}^{d} 0.5 (\langle s(x)_{j_0}, v(y)_{i_0} \rangle - E_{j_0,i_0})^2 \\
= & \sum_{j_0=1}^{n} \sum_{i_0=1}^{d} \mathsf{Loss}(x)_{j_0,i_0}
\end{aligned}$$

where the 1st equality is based on Def. A.11, the definition of Frobenius norm derives the 2nd equality, the 3rd equality is due to Def. B.3 and Def. B.4, and the 4th step is based on Def. B.6. □

B KEY DEFINITIONS OF ROPE ATTENTION

In this section, we decompose RoPE attention into its individual components, each representing a specific function or operation within the attention mechanism. These definitions provide a structured framework for understanding and analyzing the properties of RoPE attention in subsequent sections.

We denote the d^4-dimensional vector $x \in \mathbb{R}^{d^4}$ as the vectorization of a $d^2 \times d^2$ matrix X. We divide the RoPE attention to the following components to simplify our calculations and notation.

First, we define $u(x)$ for the softmax operation.

Definition B.1 (Softmax $u(x)$)**.** *We suppose there are two $n^2 \times d^2$ matrices A, W. We define $\widetilde{\mathsf{A}}$ as $\mathsf{A} \oslash \mathsf{W}$, which is an $n^2 \times d^4$ matrix. We use $\widetilde{\mathsf{A}}_{j_0}$ to denote the an $n \times d^4$ subblock of $\widetilde{\mathsf{A}}$, given that the total counts of subblocks is n. The function is defined as $u(x)_{j_0}$ maps a d^4 dimensional vector to an n-dimensional vector with every $j_0 \in [n]$ such that*

$$u(x)_{j_0} := \exp(\widetilde{\mathsf{A}}_{j_0} x).$$

Next, we define $\alpha(x)$ for the diagonal matrix.

Definition B.2 (Diagonal matrix $\alpha(x)$)**.** *We suppose two $n^2 \times d^2$ matrices A, W. Suppose that $\widetilde{\mathsf{A}} := \mathsf{A} \oslash \mathsf{W} \in \mathbb{R}^{n^2 \times d^4}$. We use $\widetilde{\mathsf{A}}_{j_0}$ to denote the an $n \times d^4$ subblock of $\widetilde{\mathsf{A}}$, given the counts of total subblocks is n. The function is defined as $\alpha(x)_{j_0}$ maps from a d^4-dimensional vector to a scalar with every $j_0 \in [n]$ such that*

$$\alpha(x)_{j_0} := \langle \exp(\widetilde{\mathsf{A}}_{j_0} x), \mathbf{1}_n \rangle.$$

We define $s(x)$ for the normalized softmax $(D^{-1} \cdot \mathrm{softmax})$.

Definition B.3 (Normalized softmax $s(x)$)**.** *From Def. B.1, it defines $u(\cdot)_{j_0}$, and we have $\alpha(\cdot)_{j_0}$based on Def. B.2. The function $s(x)_{j_0}$ maps a d^4-dimensional vector to an n-dimensional vector given every $j_0 \in [n]$ such that $s(x)_{j_0} := \alpha(x)_{j_0}^{-1} u(x)_{j_0}$.*

Lastly, we define $v(y)$ for the value matrix in the attention component.

Definition B.4 (Value matrix $v(y)$)**.** *Let $A_3 \in \mathbb{R}^{n \times d}$ be a matrix. We define $v(y)_{i_0}$ as the i_0-th column of $v(y)$. We define the function $v(y)_{i_0}$ maps a d^2-dimensional vector to an n-dimensional vector, given each i_0 in the set $[d]$, such that $v(y)_{i_0} := A_3 Y_{*,i_0}$ where $y \in \mathbb{R}^{d^2}$ is the vectorization of $n \times n$ matrix Y.*

Given the definitions of the RoPE attention components, we can now define the loss functions, which quantify the difference between the computed and target values in the context of RoPE attention.

We first introduce the error $\ell(x)_{j_0,i_0}$ between the exact RoPE attention computation $\langle s(x)_{j_0}, v(y)_{i_0} \rangle$ and approximated RoPE computation E_{j_0,i_0}.

Definition B.5 (RoPE attention error $\ell(x)$)**.** *From Def. B.3, with every j_0 in the set $[n]$, it gives $s(x)_{j_0}$ as an n-dimensional normalized vector, and we define $v(y)_{i_0}$ based on Def. B.4 given that each $i_0 \in [d]$. Defining a function $\ell(x)_{j_0,i_0}$ maps a d^4-dimensional vector to a scalar with each $j_0 \in [n]$ and each $i_0 \in [d]$ such that*

$$\ell(x)_{j_0,i_0} := \langle s(x)_{j_0}, v(y)_{i_0} \rangle - E_{j_0,i_0}.$$

Here E_{j_0,i_0} is the (j_0, i_0)-th coordinate of $E \in \mathbb{R}^{n \times d}$ for each j_0 in the set $[n]$ and i_0 in the set $[d]$, that is $\ell(x) = s(x)v(y) - E$.

Then, we define the Loss term.

Definition B.6 (Loss term $\mathsf{Loss}(x)$)**.** *Here we let $\mathsf{Loss}(x)_{j_0,i_0} := 0.5\ell(x)^2_{j_0,i_0}$ with every j_0 in the set $[n]$ and i_0 in the set $[d]$.*

C RoPE Attention Gradient Calculation

In this section, we analyze the time complexity of exact gradient computation. In Section C.1, we reformulate the closed form of the gradient. In Section C.2, we show the time complexity for $s(x)$ and $v(y)$. In Section C.3, we show the time complexity for $\ell(x)$. In Section C.4, we show the time complexity for $\beta(x)$ and $\gamma(x)$. In Section C.5, we show the total time complexity for computing the gradient of RoPE attention.

In this section, we compute the entry-wise gradient of the RoPE attention loss function from Lemma A.13

Lemma C.1. *If we have for every $i \in [d^4]$,*

- *The column function $u(x)_{j_0} \in \mathbb{R}^n$ (Definitions B.1),*
- *$\alpha(x)_{j_0}$ is a real number (Def. B.2),*
- *$s(x)_{j_0}$ is an arbitrary element in $\mathbb{R}^n$ (Def. B.3),*
- *$\ell(x)_{j_0,i_0}$ is a real number (Def. B.5), and*
- *$\mathsf{Loss}(x)_{j_0,i_0}$ is a real number (Def. B.6).*

Then, we have $\forall j_0 \in [n]$, $\forall i_0 \in [d]$,

- **Part 1.**

$$\frac{\mathrm{d}\widetilde{\mathsf{A}}_{j_0} x}{\mathrm{d}x_i} = \underbrace{(\widetilde{\mathsf{A}}_{j_0})_{*,i}}_{n \times 1}.$$

- **Part 2.**

$$\frac{\mathrm{d}u(x)_{j_0}}{\mathrm{d}x_i} = u(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}.$$

- **Part 3.**

$$\frac{\mathrm{d}\alpha(x)_{j_0}}{\mathrm{d}x_i} = \langle (\widetilde{\mathsf{A}}_{j_0})_{*,i}, u(x)_{j_0} \rangle.$$

- **Part 4.**

$$\frac{\mathrm{d}s(x)_{j_0}}{\mathrm{d}x_i} = -s(x)_{j_0} \langle (\widetilde{\mathsf{A}}_{j_0})_{*,i}, s(x)_{j_0} \rangle + s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}$$

- **Part 5.**

$$\frac{\mathrm{d}\langle s(x)_{j_0}, v(y)_{i_0} \rangle}{\mathrm{d}x_i} = \langle -s(x)_{j_0} \langle (\widetilde{\mathsf{A}}_{j_0})_{*,i}, s(x)_{j_0} \rangle + s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}, A_3 Y_{*,i_0} \rangle.$$

- **Part 6.**

$$\frac{\mathrm{d}\ell(x)_{j_0,i_0}}{\mathrm{d}x_i} = \langle -s(x)_{j_0} \langle (\widetilde{\mathsf{A}}_{j_0})_{*,i}, s(x)_{j_0} \rangle + s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}, A_3 Y_{*,i_0} \rangle.$$

- **Part 7.**

$$\frac{\mathrm{d}\mathsf{Loss}(x)_{j_0,i_0}}{\mathrm{d}x_i} = \ell(x)_{j_0,i_0} \langle -s(x)_{j_0} \langle (\widetilde{\mathsf{A}}_{j_0})_{*,i}, s(x)_{j_0} \rangle + s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}, A_3 Y_{*,i_0} \rangle.$$

Proof. To show **Part 1**,

$$\begin{aligned}
\frac{\mathrm{d}\widetilde{\mathsf{A}}_{j_0} x}{\mathrm{d}x_i} &= \widetilde{\mathsf{A}}_{j_0} \frac{\mathrm{d}x}{\mathrm{d}x_i} \\
&= \underbrace{\widetilde{\mathsf{A}}_{j_0}}_{n \times d^4} \underbrace{e_i}_{d^4 \times 1} \\
&= (\widetilde{\mathsf{A}}_{j_0})_{*,i},
\end{aligned}$$

and we note that the 1st and 2nd equalities are by the basic derivative rule and the 3rd equality is due to the basis vector definition.

To show **Part 2**,

$$\begin{aligned}
\frac{\mathrm{d}u(x)_{j_0}}{\mathrm{d}x_i} &= \frac{\mathrm{d}\exp(\widetilde{\mathsf{A}}_{j_0} x)}{\mathrm{d}x_i} \\
&= \exp(\widetilde{\mathsf{A}}_{j_0} x) \circ \frac{\mathrm{d}\widetilde{\mathsf{A}}_{j_0} x}{\mathrm{d}x_i} \\
&= \exp(\widetilde{\mathsf{A}}_{j_0} x) \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i} \\
&= u(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i},
\end{aligned}$$

and we note that the 1st equality is by Def. B.1, the 2nd equality is by chain rule, the 3rd equality is due to **Part 1**, and the 4th equality is because of Def. B.1.

To show **Part 3**,

$$\begin{aligned}
\frac{\mathrm{d}\alpha(x)_{j_0}}{\mathrm{d}x_i} &= \frac{\mathrm{d}\langle \exp(\widetilde{\mathsf{A}}_{j_0} x), \mathbf{1}_n \rangle}{\mathrm{d}x_i} \\
&= \langle \frac{\mathrm{d}\exp(\widetilde{\mathsf{A}}_{j_0} x)}{\mathrm{d}x_i}, \mathbf{1}_n \rangle + \langle \exp(\widetilde{\mathsf{A}}_{j_0} x), \frac{\mathrm{d}\mathbf{1}_n}{\mathrm{d}x_i} \rangle \\
&= \langle \frac{\mathrm{d}\exp(\widetilde{\mathsf{A}}_{j_0} x)}{\mathrm{d}x_i}, \mathbf{1}_n \rangle \\
&= \langle u(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}, \mathbf{1}_n \rangle \\
&= \langle (\widetilde{\mathsf{A}}_{j_0})_{*,i}, u(x)_{j_0} \rangle,
\end{aligned}$$

and we note that the 1st equality is by Def. B.2, the 2nd equality is by product rule, the 3rd equality is due to $\frac{\mathrm{d}\mathbf{1}_n}{\mathrm{d}x_i} = \mathbf{0}_n$, the 4th equality is because of Def. B.1, and 5th equality derives from basic algebra.

To show **Part 4**,

$$\begin{aligned}
\frac{\mathrm{d}s(x)_{j_0}}{\mathrm{d}x_i} =& \frac{\mathrm{d}(\alpha(x)_{j_0}^{-1} u(x)_{j_0})}{\mathrm{d}x_i} \\
=& \frac{\mathrm{d}\alpha(x)_{j_0}^{-1}}{\mathrm{d}x_i} u(x)_{j_0} + \alpha(x)_{j_0}^{-1} \frac{\mathrm{d}u(x)_{j_0}}{\mathrm{d}x_i} \\
=& -\alpha(x)_{j_0}^{-2} \frac{\mathrm{d}\alpha(x)_{j_0}}{\mathrm{d}x_i} u(x)_{j_0} + \alpha(x)_{j_0}^{-1} \frac{\mathrm{d}u(x)_{j_0}}{\mathrm{d}x_i} \\
=& -\alpha(x)_{j_0}^{-2} \langle \underbrace{(\widetilde{\mathsf{A}}_{j_0})_{*,i}}_{n \times 1}, \underbrace{u(x)_{j_0}}_{n \times 1} \rangle u(x)_{j_0} + \alpha(x)_{j_0}^{-1} (\underbrace{u(x)_{j_0}}_{n \times 1} \circ \underbrace{(\widetilde{\mathsf{A}}_{j_0})_{*,i}}_{n \times 1}) \\
=& -\alpha(x)_{j_0}^{-1} s(x)_{j_0} \langle \underbrace{(\widetilde{\mathsf{A}}_{j_0})_{*,i}}_{n \times 1}, \underbrace{u(x)_{j_0}}_{n \times 1} \rangle + s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i} \\
=& -s(x)_{j_0} \langle \underbrace{(\widetilde{\mathsf{A}}_{j_0})_{*,i}}_{n \times 1}, \underbrace{s(x)_{j_0}}_{n \times 1} \rangle + s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i},
\end{aligned}$$

and we note that the 1st equality is by Def. B.3, the 2nd equality is by product rule, the 3rd equality is due to chain rule, the 4th equality is because of previous parts, the 5th and 6th equalities derive from Def. B.3.

To show **Part 5**,

$$\begin{aligned}
\frac{\mathrm{d}\langle s(x)_{j_0}, v(y)_{i_0} \rangle}{\mathrm{d}x_i} =& \langle \frac{\mathrm{d}s(x)_{j_0}}{\mathrm{d}x_i}, v(y)_{i_0} \rangle + \langle s(x)_{j_0}, \frac{\mathrm{d}v(y)_{i_0}}{\mathrm{d}x_i} \rangle \\
=& \langle \frac{\mathrm{d}s(x)_{j_0}}{\mathrm{d}x_i}, v(y)_{i_0} \rangle \\
=& \langle -s(x)_{j_0} \langle (\widetilde{\mathsf{A}}_{j_0})_{*,i}, s(x)_{j_0} \rangle + s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}, A_3 Y_{*,i_0} \rangle,
\end{aligned}$$

and we note that the 1st equality is due to the product rule, the 2nd equality is by $\frac{\mathrm{d}v(y)_{i_0}}{\mathrm{d}x_i} = \mathbf{0}_n$, and the 3rd equality is due to the previous part.

To show **Part 6**,

$$\begin{aligned}
\frac{\mathrm{d}\ell(x)_{j_0,i_0}}{\mathrm{d}x_i} =& \frac{\mathrm{d}(\langle s(x)_{j_0}, v(y)_{i_0} \rangle - E_{j_0,i_0})}{\mathrm{d}x_i} \\
=& \frac{\mathrm{d}\langle s(x)_{j_0}, v(y)_{i_0} \rangle}{\mathrm{d}x_i} \\
=& \langle -s(x)_{j_0} \langle \underbrace{(\widetilde{\mathsf{A}}_{j_0})_{*,i}}_{n \times 1}, \underbrace{s(x)_{j_0}}_{n \times 1} \rangle + s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}, A_3 Y_{*,i_0} \rangle,
\end{aligned}$$

and we note that the 1st equality is by Def. B.5, the 2nd equality is by $\frac{\mathrm{d}E_{j_0,i_0}}{\mathrm{d}x_i} = \mathbf{0}_n$, and the 3rd equality is due to the previous part.

To show **Part 7**,

$$\begin{aligned}
\frac{\mathrm{d}\mathsf{Loss}(x)_{j_0,i_0}}{\mathrm{d}x_i} =& 0.5 \frac{\mathrm{d}(\ell(x)_{j_0,i_0})^2}{\mathrm{d}x_i} \\
=& \ell(x)_{j_0,i_0} \cdot \frac{\mathrm{d}\ell(x)_{j_0,i_0}}{\mathrm{d}x_i} \\
=& \ell(x)_{j_0,i_0} \langle -s(x)_{j_0} \langle \underbrace{(\widetilde{\mathsf{A}}_{j_0})_{*,i}}_{n \times 1}, \underbrace{s(x)_{j_0}}_{n \times 1} \rangle + s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}, A_3 Y_{*,i_0} \rangle,
\end{aligned}$$

and we note that the 1st equality is by Def. B.6, the 2nd equality is by chain rule and the 3rd equality is due to the previous part. □

C.1 Reformulate the Gradient into Its Closed Form

In this section, we reformulate the entry-wise gradient of the RoPE loss function from Lemma 4.1 into its matrix form.

We first begin with reformulating the gradient with respect to the entire vector x.

Lemma C.2 (Gradient Reformulation, $\frac{\mathrm{d}\mathsf{Loss}(x)_{j_0,i_0}}{\mathrm{d}x}$)**.** *If we have for every $i \in [d^4]$,*

- *The column function $u(x)_{j_0} \in \mathbb{R}^n$ (Definitions B.1),*
- *$\alpha(x)_{j_0}$ is a real number (Def. B.2),*
- *$s(x)_{j_0}$ is an arbitrary element in $\mathbb{R}^n$ (Def. B.3),*
- *$\ell(x)_{j_0,i_0}$ is a real number (Def. B.5), and*
- *$\mathsf{Loss}(x)_{j_0,i_0}$ is a real number (Def. B.6).*

Then, we have

$$\frac{\mathrm{d}\mathsf{Loss}(x)_{j_0,i_0}}{\mathrm{d}x} = \ell(x)_{j_0,i_0} \widetilde{\mathsf{A}}_{j_0}^\top (\mathrm{diag}(s(x)_{j_0}) A_3 Y_{*,i_0} - s(x)_{j_0} s(x)_{j_0}^\top A_3 Y_{*,i_0}).$$

Proof.

$$\begin{aligned}
\frac{\mathrm{d}\mathsf{Loss}(x)_{j_0,i_0}}{\mathrm{d}x_i} &= \ell(x)_{j_0,i_0} \langle -s(x)_{j_0} \langle (\widetilde{\mathsf{A}}_{j_0})_{*,i}, s(x)_{j_0} \rangle + s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}, A_3 Y_{*,i_0} \rangle \\
&= \ell(x)_{j_0,i_0} \langle s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}, A_3 Y_{*,i_0} \rangle - \ell(x)_{j_0,i_0} \langle s(x)_{j_0} \langle (\widetilde{\mathsf{A}}_{j_0})_{*,i}, s(x)_{j_0} \rangle, A_3 Y_{*,i_0} \rangle \\
&= \ell(x)_{j_0,i_0} \langle s(x)_{j_0} \circ (\widetilde{\mathsf{A}}_{j_0})_{*,i}, A_3 Y_{*,i_0} \rangle - \ell(x)_{j_0,i_0} \langle (\widetilde{\mathsf{A}}_{j_0})_{*,i}, s(x)_{j_0} \rangle \langle s(x)_{j_0}, A_3 Y_{*,i_0} \rangle \\
&= \ell(x)_{j_0,i_0} (\widetilde{\mathsf{A}}_{j_0}^\top)_{*,i} \,\mathrm{diag}(s(x)_{j_0}) A_3 Y_{*,i_0} - \ell(x)_{j_0,i_0} (\widetilde{\mathsf{A}}_{j_0}^\top)_{*,i} s(x)_{j_0} s(x)_{j_0}^\top A_3 Y_{*,i_0} \\
&= \ell(x)_{j_0,i_0} (\widetilde{\mathsf{A}}_{j_0}^\top)_{*,i} (\mathrm{diag}(s(x)_{j_0}) - s(x)_{j_0} s(x)_{j_0}^\top) A_3 Y_{*,i_0}.
\end{aligned}$$

where the 1st step follows from Lemma 4.1, and all other steps follow from Fact A.9.

Then, the gradient can be reformulated as follows.

$$\frac{\mathrm{d}\mathsf{Loss}(x)_{j_0,i_0}}{\mathrm{d}x} = \ell(x)_{j_0,i_0} \widetilde{\mathsf{A}}_{j_0}^\top (\mathrm{diag}(s(x)_{j_0}) A_3 Y_{*,i_0} - s(x)_{j_0} s(x)_{j_0}^\top A_3 Y_{*,i_0}).$$

Thus, we complete the proof. □

Next, we show our reformulation of the gradient by dropping the index i_0 from $\mathsf{Loss}(x)_{j_0,i_0}$

Lemma C.3 (Gradient Reformulation, $\frac{\mathrm{d}\mathsf{Loss}(x)_{j_0}}{\mathrm{d}x}$)**.** *If we have for every $i \in [d^4]$,*

- *The column function $u(x)_{j_0} \in \mathbb{R}^n$ (Definitions B.1),*
- *$\alpha(x)_{j_0}$ is a real number (Def. B.2),*
- *$s(x)_{j_0}$ is an arbitrary element in $\mathbb{R}^n$ (Def. B.3),*
- *$\ell(x)_{j_0,i_0}$ is a real number (Def. B.5),*
- *$\mathsf{Loss}(x)_{j_0,i_0}$ is a real number (Def. B.6), and*
- *$\beta(x)_{j_0} \in \mathbb{R}^n$ is $\underbrace{A_3 Y}_{n \times d} \underbrace{\ell(x)_{j_0,*}^\top}_{d \times 1}$.*

Then, we get

$$\begin{aligned}
\frac{\mathrm{d}\mathsf{Loss}(x)_{j_0}}{\mathrm{d}x} &= \widetilde{\mathsf{A}}_{j_0}^\top (s(x)_{j_0} \circ A_3 \beta(x)_{j_0}) \\
&\quad - \widetilde{\mathsf{A}}_{j_0}^\top s(x)_{j_0} \langle s(x)_{j_0}, A_3 \beta(x)_{j_0} \rangle.
\end{aligned}$$

Proof. We can get

$$\begin{aligned}
&\frac{\mathrm{d}\mathsf{Loss}(x)_{j_0}}{\mathrm{d}x} \\
=& \sum_{i_0 \in [d]} \frac{\mathrm{d}\mathsf{Loss}(x)_{j_0,i_0}}{\mathrm{d}x} \\
=& \sum_{i_0 \in [d]} \widetilde{\mathsf{A}}_{j_0}^\top (\mathrm{diag}(s(x)_{j_0}) - s(x)_{j_0} s(x)_{j_0}^\top) \ell(x)_{j_0,i_0} A_3 Y_{*,i_0} \\
=& \widetilde{\mathsf{A}}_{j_0}^\top (\mathrm{diag}(s(x)_{j_0}) - s(x)_{j_0} s(x)_{j_0}^\top) \beta(x)_{j_0},
\end{aligned}$$

and we note that the first equality is because Lemma A.13, the 2nd equality is due to basic algebra, and the 3rd equality comes from the lemma statement.

Thus, we complete this proof. □

Finally, we reformulate the gradient into its matrix form.

Lemma C.4 (Gradient Reformulation, $\frac{\mathrm{d}\mathsf{Loss}(x)}{\mathrm{d}x}$, Formal version of Lemma 4.1)**.** *If we have for every $i \in [d^4]$,*

- *The column function $u(x)_{j_0} \in \mathbb{R}^n$ (Definitions B.1),*
- *$\alpha(x)_{j_0}$ is a real number (Def. B.2),*
- *$s(x)_{j_0}$ is an arbitrary element in $\mathbb{R}^n$ (Def. B.3),*
- *$\ell(x)_{j_0,i_0}$ is a real number (Def. B.5),*
- *$\mathsf{Loss}(x)_{j_0,i_0}$ is a real number (Def. B.6),*
- *$\beta(x)_{j_0} \in \mathbb{R}^n$ is $\underbrace{A_3 Y}_{n \times d} \underbrace{\ell(x)_{j_0,*}^\top}_{d \times 1}$, and*
- *$\gamma(x)_{j_0} \in \mathbb{R}^n$ is $(\mathrm{diag}(s(x)_{j_0}) - s(x)_{j_0} s(x)_{j_0}^\top) \beta(x)_{j_0}$*

Then, we get

$$\frac{\mathrm{d}\mathsf{Loss}(x)}{\mathrm{d}x} = \underbrace{\widetilde{\mathsf{A}}^\top}_{d^4 \times n^2} \mathrm{vec}(\underbrace{\gamma(x)}_{n \times n})$$

Proof. We show that

$$\begin{aligned}
\frac{\mathrm{d}\mathsf{Loss}(x)}{\mathrm{d}x} =& \sum_{j_0 \in [n]} \frac{\mathrm{d}\mathsf{Loss}(x)_{j_0}}{\mathrm{d}x} \\
=& \sum_{j_0 \in [n]} \widetilde{\mathsf{A}}_{j_0}^\top \gamma(x)_{j_0} \\
=& \widetilde{\mathsf{A}}^\top \mathrm{vec}(\gamma(x)),
\end{aligned}$$

where we note that the 1st equality is because of Lemma A.13, the 2nd equality is based on the lemma statement, and the 3rd equality derives from basic concepts of vectorization.

Thus, we complete the proof. □

C.2 Time Complexity for Computing $s(x)$ and $v(y)$ Functions

In this section, we use the vector and matrix multiplication time complexity from Definition A.2 and Fact A.3 to analyze the complexity of computing $s(x)$ and $v(y)$.

Lemma C.5. *Pick $s(x)$ and $v(y)$ from Def. B.3 and Def. B.4, then it costs $O(\mathcal{T}_{\mathrm{mat}}(n,d,d) + \mathcal{T}_{\mathrm{mat}}(n,d,n))$ time to get $s(x)$, and it costs $\mathcal{T}_{\mathrm{mat}}(n,d,d)$ time to get $v(y)$.*

Proof. We first show the time complexity of $s(x)$.

Let $A \in \mathbb{R}^{n \times n}$ be the RoPE attention matrix. Let $D = A\mathbf{1}_n$. Then

$$s(x) = D^{-1}A.$$

Then, we need $\mathcal{T}_{\mathrm{mat}}(n,d,d) + \mathcal{T}_{\mathrm{mat}}(n,d,n)$ time to get A.

Next, we need $O(n^2)$ time to get D.

Now, we need $O(n^2)$ time to get $D^{-1}A$.

Therefore, they cost time $O(\mathcal{T}_{\mathrm{mat}}(n,d,d) + \mathcal{T}_{\mathrm{mat}}(n,d,n))$ time .

We show the time complexity of $v(y)$.

To get $v(y) = A_3 Y$, it costs time $\mathcal{T}_{\mathrm{mat}}(n,d,d)$.

Thus, we complete the proof. □

C.3 Time Complexity for Computing $\ell(x)$ Functions

In this section, we use the vector and matrix multiplication time complexity from Definition A.2 and Fact A.3 to analyze the complexity of computing $\ell(x)$.

Lemma C.6. *We have $\ell(x)$ from Def. B.5, then it costs $\mathcal{T}_{\mathrm{mat}}(n,n,d) + O(nd)$ to calculate $\ell(x)$.*

Proof. We show the time complexity of $\ell(x)$, where $\ell(x) = s(x)v(y) - E$.

It costs $\mathcal{T}_{\mathrm{mat}}(n,n,d)$ time to get $s(x)v(y)$.

Then, it requires $O(nd)$ time to get $s(x)v(y) - E$.

Therefore, they cost time $\mathcal{T}_{\mathrm{mat}}(n,n,d) + O(nd)$.

Thus, we complete the proof. □

C.4 Time Complexity for Computing $\beta(x)$ and $\gamma(x)$ Functions

In this section, we use the vector and matrix multiplication time complexity from Definition A.2 and Fact A.3 to analyze the complexity of computing $\beta(x)$ and $\gamma(x)$.

Lemma C.7. *Let $\beta(x) \in \mathbb{R}^{n \times n}$ be defined as $\beta(x) := \ell(x)v(y)^\top$ and $\gamma(x)$ be defined as $\gamma(x)_{j_0} := (\mathrm{diag}(s(x)_{j_0} - s(x)_{j_0} s(x)_{j_0}^\top \beta(x)_{j_0} \in \mathbb{R}^n$, given that $s(x) \in \mathbb{R}^{n \times n}$ then $\beta(x)$ can be computed in time of $O(\mathcal{T}_{\mathrm{mat}}(n,n,d))$ and $\gamma(x)$ can be computed in time of $O(n^2)$.*

Proof. Here we present $\beta(x)$ as follows: $\beta(x) = \ell(x)v(y)^\top$.

It costs $\mathcal{T}_{\mathrm{mat}}(n,d,n)$ time to get $\ell(x)v(y)^\top$, which equals to $O(\mathcal{T}_{\mathrm{mat}}(n,n,d))$.

Next, we show the time complexity for $\gamma(x)_{j_0} = (\mathrm{diag}(s(x)_{j_0} - s(x)_{j_0} s(x)_{j_0}^\top)\beta(x)_{j_0}$ It costs $O(n)$ time to get $\gamma(x)_{j_0}$. The reason is that $s(x)_{j_0} s(x)_{j_0}^\top$ is a rank one matrix and $\mathrm{diag}(s(x)_{j_0})$ is a diagonal matrix

Given $j_0 \in [n]$, the time for $\gamma(x)$ is $O(n^2)$ and we finish the proof. □

C.5 Time Complexity for Computing the Gradient of RoPE Attention

Theorem C.8 (RoPE attention gradient computation time complexity, Restatement of Theorem 4.2)**.** *We define three $n \times d$ input sequence matrices as A_1, A_2, A_3, and the $n \times d$ approximated attention computation matrix as E. We define several input fixed matrices as $X_1, X_2, Y \in \mathbb{R}^{d \times d}$. We define $\mathsf{X} = X_1 \otimes X_2$, $\mathsf{A} = A_1 \otimes A_2$. We define $x := \mathrm{vec}(\mathsf{X})$ and try to get the Loss function gradient. Let*

$g := \frac{\mathrm{d}\mathsf{Loss}(X_1, X_2)}{\mathrm{d}x}$ *where* $\mathsf{Loss}(X_1, X_2)$ *from Def. 3.1. Then, it costs* $O(\mathcal{T}_{\mathrm{mat}}(n,d,d)+\mathcal{T}_{\mathrm{mat}}(n,d,n))$ *time to get the gradient* $g \in \mathbb{R}^{d^4}$.

Proof. We show the time complexity of g as follows.

1. We need time $O(\mathcal{T}_{\mathrm{mat}}(n,d,d) + \mathcal{T}_{\mathrm{mat}}(n,d,n))$ for $s(x), v(y)$ from Lemma C.2.
2. We need time $O(\mathcal{T}_{\mathrm{mat}}(n,n,d) + \mathcal{T}_{\mathrm{mat}}(n,d,d))$ for $\ell(x)$ from Lemma C.3.
3. We need time $O(\mathcal{T}_{\mathrm{mat}}(n,n,d))$ for $\beta(x)$ from Lemma C.7.
4. We need time $O(n^2)$ for $\gamma(x)$ from Lemma C.7.

Therefore, it costs $O(\mathcal{T}_{\mathrm{mat}}(n,d,d) + \mathcal{T}_{\mathrm{mat}}(n,d,n))$ time overall for the gradient computation.

Thus, we complete the proof. □

D Low Rank Approximation of RoPE Attention

This section presents the fast running time using the low-rank approximations where the low-rank matrices are generated from the polynomial method (see Lemma A.1).

D.1 Approximate s Using Low Rank Approximation

In this section, we use the low-rank approximation technique to approximate $s(x)$

Lemma D.1 (Low Rank Approximate $s(x)$)**.** *For any* $B = o(\sqrt{\log n})$*, let* k_1 *equals to* $n^{o(1)}$ *such that: Suppose we have two* $n \times d$ *matrices* A_1, A_2*,* $X_1, X_2 \in \mathbb{R}^{d \times d}$ *and* $\mathsf{X} = X_1 \otimes X_2 \in \mathbb{R}^{d^2 \times d^2}$*. Assume we can use* $O(\log n)$ *bits to write every entry from* $s(x)$*. It holds that* $\max\{\|A_1 X_1\|_\infty, \|A_2 X_2\|_\infty\} \leq B$*, then there are three matrices* $U_1, V_1, W_1 \in \mathbb{R}^{n \times k_1}$ *such that* $\|U_1 V_1^\top - s(x)\|_\infty \leq \epsilon / \operatorname{poly}(n)$*. Here* $s(x) = D^{-1}A \in \mathbb{R}^{n \times n}$ *where* A *is defined as the matrix representation of* $\exp((\mathsf{A} \odot \mathsf{W}) \operatorname{vec}(X))$*, and* $D = \operatorname{diag}(A/d)\mathbf{1}_n$*. Moreover, these matrices* U_1, V_1 *can be created explicitly in* $n^{1+o(1)}$ *time.*

Proof. By definition of $A(\mathsf{X})$, we have

$$\operatorname{vec}(A(\mathsf{X})) = \exp(\mathsf{A} \odot \mathsf{W}) \operatorname{vec}(X).$$

Hence, using the tensor trick, we have

$$\begin{aligned} A(\mathsf{X})_{i,j} =& \exp((\mathsf{A}_{i+(j-1)n} \otimes \mathsf{W}_{i+(j-1)n}) \operatorname{vec}(\mathsf{X})/d) \\ =& \exp((\mathsf{A}_{i+(j-1)n} \otimes w_{i-j}^\top) \operatorname{vec}(\mathsf{X})/d). \end{aligned}$$

We define $w_{i-j} := \operatorname{vec}(W_{i-j}) \in \mathbb{R}^{d^2}$ and define W such that W_{j_0} is an $1 \times d^2$ block and $\mathsf{W}_{i+(j-1)n} := w_{i-j}^\top$. We also define $\mathsf{A} := A_1 \otimes A_2 \in \mathbb{R}^{n^2 \times d^2}$ and $\mathsf{X} := X_1 \otimes X_2 \in \mathbb{R}^{d^2 \times d^2}$. We use A_{j_0} to denote the a $1 \times d^2$ subblock of A.

We can reformulate the attention matrix A as, for $i, j \in [n]$

$$A(\mathsf{X})_{i,j} = \exp(\underbrace{\mathsf{A}_{i+(j-1)n}}_{1 \times d^2} \underbrace{\mathsf{X}}_{d^2 \times d^2} \underbrace{w_{i-j}/d}_{d^2 \times 1}).$$

Thus, we can show that

$$\begin{aligned} & \mathsf{A}_{i+(j-1)n} \mathsf{X} w_{i-j}, \\ =& (A_{1,i,*} \otimes A_{2,j,*})(X_1 \otimes X_2) \operatorname{vec}(W_{i-j}) \\ =& A_{1,i,*} X_1 W_{i-j} X_2^\top A_{2,j,*}^\top \end{aligned}$$

where 1st equality uses definitions of w_{i-j}, A, and X, and the second step uses the tensor trick. We complete our proof after applying Lemma 5.1. □

D.2 APPROXIMATE ℓ USING LOW RANK APPROXIMATION

In this section, we use the low-rank approximation technique to approximate $\ell(x)$

Lemma D.2 (Low Rank Approximate $\ell(x)$)**.** *Let d equal $O(\log n)$. Suppose we can use $O(\log n)$ bits to write every entry in $E, v(y) \in \mathbb{R}^{n \times d}$. Define the $\ell(x) \in \mathbb{R}^{n \times d}$ as specified in Def. B.5. Then, we have $U_1, V_1 \in \mathbb{R}^{n \times k_1}$ such that $\|U_1 V_1^\top v(y) - E - \ell(x)\|_\infty \leq \epsilon / \operatorname{poly}(n)$.*

Proof. Here, we present the bound as follows.

$$\begin{aligned} \|U_1 V_1^\top v(y) - E - \ell(x)\|_\infty &= \|U_1 V_1^\top v(y) - s(x) v(y)\|_\infty \\ &= \|v(y)\|_\infty \cdot \|U_1 V_1^\top - s(x)\|_\infty \\ &\leq \epsilon / \operatorname{poly}(n), \end{aligned}$$

where the 1st is because of Def. B.5, 2nd step is based on the distributive law, and 3rd step is due to Lemma D.1. □

D.3 APPROXIMATE β USING LOW RANK APPROXIMATION

In this section, we use the low-rank approximation technique to approximate $\beta(x)$

Lemma D.3 (Low Rank Approximate $\beta(x)$)**.** *Let $k_2 = n^{o(1)}$. We define $\ell(x) \in \mathbb{R}^{n \times d}$ based on Def. B.5, and $v(y) \in \mathbb{R}^{n \times d}$ based on Def. B.4. We suppose $\beta(x)$ is equal to $v(y)\ell(x)^\top$, which is an $n \times n$ matrix. Let $U_2, V_2 \in \mathbb{R}^{n \times k_2}$ such that $\|U_2 V_2^\top - \beta(x)\|_\infty \leq \epsilon / \operatorname{poly}(n)$. In $n^{1+o(1)}$ time, we can get U_2, V_2.*

Proof. Let $\widetilde{\beta}(x) \approx \beta(x)$

By Lemma D.2, $U_1 V_1^\top v(y) - E$ approximately equals to $\ell(x)$.

Then we define $\widetilde{\beta}(x) = v(y)(U_1 V_1^\top v(y) - E)^\top$.

We can use the low-rank technique to represent $\widetilde{\beta}(x) = v(y) v(y)^\top V_1 U_1^\top - v(y) E^\top$.

Also, $v(y)^\top V_1$ can be computed at first because it takes $n^{1+o(1)}$ time.

Given that all low-rank matrices, we have $U_2, V_2 \in \mathbb{R}^{n \times k_2}$ where $k_2 = \max\{d, k\} + d = n^{o(1)}$.

Here, we present the proof for obtaining the bound.

$$\begin{aligned} \|\widetilde{\beta}(x) - \beta(x)\|_\infty &= \|v(y)(U_1 V_1^\top v(y)) - E)^\top - v(y)\ell(x)^\top\|_\infty \\ &\leq \|U_1 V_1^\top v(y)) - E - \ell(x)\|_\infty \cdot \|v(y)\|_\infty \cdot d \\ &\leq \epsilon / \operatorname{poly}(n) \end{aligned}$$

where the first step is based on the definition of $\beta(x)$ and $\widetilde{\beta}(x)$, the second step is due to the distributive law, and the third step derives from Lemma D.2.

Thus, we complete the proof. □

D.4 APPROXIMATE γ USING LOW RANK APPROXIMATION

In this section, we use the low-rank approximation technique to approximate $\gamma(x)$. Specifically, we apply the polynomial methods to $\gamma_1(x)$ and $\gamma_2(x)$ where $\gamma(x) = \gamma_1(x) - \gamma_2(x)$.

First, we show the low-rank approximation of $\gamma_1(x)$.

Lemma D.4 (Low Rank Approximate $\gamma_1(x)$)**.** *Let $k_1 = n^{o(1)}$. Let $k_2 = n^{o(1)}$. We suppose $\gamma_1(x)$ is $\operatorname{diag}(s(x))\beta(x)$, and U_1, V_1 be two $n \times k_1$ matrices, in which $\|U_1 V_1^\top - f(x)\|_\infty \leq \frac{\epsilon}{\operatorname{poly}(n)}$. We suppose two $n \times k_2$ matrices U_2, V_2 in which $\|U_2 V_2^\top - \beta(x)\|_\infty \leq \frac{\epsilon}{\operatorname{poly}(n)}$. Then we have two $n \times k_3$ matrices in which $\|U_3 V_3^\top - \gamma_1(x)\|_\infty \leq \epsilon / \operatorname{poly}(n)$. We can construct U_3, V_3 in $n^{1+o(1)}$ time.*

Proof. Let $U_3 = U_1 \oslash U_2$ and $V_3 = V_1 \oslash V_2$, and we can use $n^{1+o(1)}$ time to get them.

Let $\widetilde{s}(x) = U_1 V_1^\top$ and $\widetilde{\beta}(x) = U_2 V_2^\top$.

Then, we have the following by Fact A.5.

$$\begin{aligned}
\|U_3 V_3^\top - \gamma_1(x)\|_\infty &\leq \|U_3 V_3^\top - \mathrm{diag}(s(x))\beta(x)\|_\infty \\
&= \|(U_1 \oslash U_2)(V_1 \oslash V_2)^\top - \mathrm{diag}(s(x))\beta(x)\|_\infty \\
&= \|\,\mathrm{diag}(U_1 V_1^\top)(U_2 V_2^\top) - \mathrm{diag}(s(x))\beta(x)\|_\infty \\
&= \|\,\mathrm{diag}(\widetilde{s}(x))\widetilde{\beta}(x) - \mathrm{diag}(s(x))\beta(x)\|_\infty \\
&= \|\,\mathrm{diag}(\widetilde{s}(x))\widetilde{\beta}(x) - \mathrm{diag}(\widetilde{s}(x))\beta(x) + \mathrm{diag}(\widetilde{s}(x))\beta(x) - \mathrm{diag}(s(x))\beta(x)\|_\infty \\
&\leq \|\,\mathrm{diag}(\widetilde{s}(x))\widetilde{\beta}(x) - \mathrm{diag}(\widetilde{s}(x))\beta(x)\|_\infty + \|\,\mathrm{diag}(\widetilde{s}(x))\beta(x) - \mathrm{diag}(s(x))\beta(x)\|_\infty \\
&\leq \frac{\epsilon}{\mathrm{poly}(n)}
\end{aligned}$$

where the first inequality is because of the def. of $\gamma_1(x)$, the second equality is due to the def. of U_3, V_3, the third equality is based on Fact A.5, the fourth equality is due to the def. of $\widetilde{s}(x)$ and $\widetilde{\beta}(x)$, the fifth equality is due to simple arithmetic, the sixth inequality is because of the triangle inequality, and the seventh inequality derives from Lemma D.1 and Lemma D.3. □

Next, we show the low-rank approximation of $\gamma_2(x)$.

Lemma D.5 (Low Rank Approximate $\gamma_2(x)$)**.** *Let $k_1 = n^{o(1)}$. Let $k_2 = n^{o(1)}$. Let $k_4 = n^{o(1)}$. Let $\gamma_2(x) \in \mathbb{R}^{n \times n}$ where for j_0 in set $[n]$, j_0 represents j_0-th column, $\gamma_2(x)_{j_0} = s(x)_{j_0} s(x)_{j_0}^\top \beta(x)_{j_0}$. We suppose $U_1, V_1 \in \mathbb{R}^{n \times k_1}$ in which $\|U_1 V_1^\top - s(x)\|_\infty \leq \frac{\epsilon}{\mathrm{poly}(n)}$. We suppose two $n \times k_2$ matrices U_2, V_2 in which $\|U_2 V_2^\top - \beta(x)\|_\infty \leq \frac{\epsilon}{\mathrm{poly}(n)}$. Then, we have $U_4, V_4 \in \mathbb{R}^{n \times k_4}$ such that $\|U_4 V_4^\top - \gamma_2(x)\|_\infty \leq \epsilon / \mathrm{poly}(n)$. We can get U_4, V_4 in $n^{1+o(1)}$ time.*

Proof. Let $\rho(x) \in \mathbb{R}^n$ be $\rho(x)_{j_0} := s(x)_{j_0} \beta(x)_{j_0}$.

We define $\widetilde{\rho}(x) \approx \rho(x)$.

Let $(U_1 V_1)_{j_0,*}^\top \approx s(x)_{j_0}$ and $(U_2 V_2)_{j_0,*}^\top \approx \beta(x)_{j_0}$.

Then, we define $\widetilde{\rho}(x)_{j_0}$ as the inner product of $\widetilde{s}(x)_{j_0}$ and $\widetilde{\beta}(x)_{j_0}$, and by Fact A.9, we have $\widetilde{\rho}(x)_{j_0} = (U_1 V_1)_{j_0,*} \cdot (U_2 V_2)_{j_0,*}^\top$

Then, it costs $n^{1+o(1)}$ time if we compute $V_1 V_2^\top$ first.

Now, we show

$$\begin{aligned}
\widetilde{\rho}(x)_{j_0} &= (U_1 V_1)_{j_0,*} \cdot (U_2 V_2)_{j_0,*}^\top \\
&= \underbrace{(U_1)_{j_0,*}}_{1 \times k_1} \underbrace{V_1 V_2^\top}_{k_1 \times k_2} \underbrace{(U_2)_{j_0,*}^\top}_{k_2 \times 1}
\end{aligned}$$

Once the $V_1 V_2^\top$ are pre-computed, the above step only takes $O(k_1 k_2)$ time. Given that $j_0 \in [n]$, we can have the total time $O(n k_1 k_2) = n^{1+o(1)}$.

We suppose $\widetilde{s}(x)$ approximates $s(x)$ and set is equal to $U_1 V_1^\top$. Then, we are able to approximate $\gamma_2(x)$ using $\widetilde{s}(x)$ and $\widetilde{\rho}(x)$ as follows.

We suppose $\widetilde{\gamma}_2(x)$ equals to $\widetilde{s}(x)\,\mathrm{diag}(\widetilde{\rho}(x))$. U_4 and V_4 can be obtained since we can use the low-rank approximation technique to represent $\widetilde{s}(x)$ and $\mathrm{diag}(\widetilde{\rho}(x))$ is a diagonal matrix. Basically $U_4 = U_1$ and $V_4 = \mathrm{diag}(\widetilde{\rho}(x)) V_1$.

Now, we need to control the error. We have

$$\begin{aligned}
\|U_4 V_4^\top - \gamma_2(x)\|_\infty &= \|\widetilde{\gamma}_2(x) - \gamma_2(x)\|_\infty \\
&= \max_{j_0 \in [n]} \|\widetilde{s}(x)_{j_0} \widetilde{\rho}(x)_{j_0} - s(x)_{j_0} \rho(x)_{j_0}\|_\infty
\end{aligned}$$

$$
\begin{aligned}
&= \max_{j_0 \in [n]} \|\widetilde{s}(x)_{j_0} \widetilde{\rho}(x)_{j_0} - \widetilde{s}(x)_{j_0} \rho(x)_{j_0} + \widetilde{s}(x)_{j_0} \rho(x)_{j_0} - s(x)_{j_0} \rho(x)_{j_0} \|_\infty \\
&\leq \max_{j_0 \in [n]} \|\widetilde{s}(x)_{j_0} \widetilde{\rho}(x)_{j_0} - \widetilde{s}(x)_{j_0} \rho(x)_{j_0} \|_\infty + \|\widetilde{s}(x)_{j_0} \rho(x)_{j_0} - s(x)_{j_0} \rho(x)_{j_0} \|_\infty \\
&\leq \max_{j_0 \in [n]} \|\widetilde{s}(x)_{j_0}\|_\infty \cdot \|\widetilde{\rho}(x)_{j_0} - \rho(x)_{j_0}\|_\infty + \|\widetilde{s}(x)_{j_0} - s(x)_{j_0}\|_\infty \cdot \|\rho(x)_{j_0}\|_\infty \\
&\leq \epsilon / \operatorname{poly}(n)
\end{aligned}
$$

where the 1st equality is based on the def. of $\widetilde{\gamma}_2(x)$, the 2nd equality is due to def. of $\widetilde{\gamma}_2(x)$ and $\gamma_2(x)$, the 3rd equality is due to simple mathematical properties, the 4th step is due to the triangle inequalities, and the 5th step is due to the distributive law.

Thus, we complete the proof. □

D.5 Fast Computation in Almost Linear Time

In this section, we present our main result. With the low-rank approximation, we can approximate the RoPE gradient computations in almost linear time.

Theorem D.6 (Main result, Low Rank Approximate RoPE Attention Gradient, Restatement of Theorem 5.2)**.** *Assuming the entries of A_1, A_2, X_1, X_2, Y, E are represented using $O(\log n)$ bits, there is an $n^{1+o(1)}$ time algorithm to solve $\mathsf{AAttLGC}(n, d = O(\log n), B = o(\sqrt{\log n}))$, from Def. 3.1, with the accuracy upper bounded by $\frac{1}{\operatorname{poly}(n)}$. To be more specific, a gradient vector $\widetilde{g} \in \mathbb{R}^{d^4}$ comes out of our algorithm where $\|\frac{\mathrm{d}\mathsf{Loss}}{\mathrm{d}x} - \widetilde{g}\|_\infty \leq \frac{1}{\operatorname{poly}(n)}$.*

Proof. By Lemma D.5 and Lemma D.4, There are matrices $\gamma(x), \gamma_1(x) \in \mathbb{R}^{n \times n}$ and $\gamma_2(x)$, we have

$$\gamma(x) = \gamma_1(x) - \gamma_2(x).$$

We assume Lemma D.4 and Lemma D.5 are true from Lemma D.1 to Lemma D.3. Thus, we can have the following based on Lemma D.4 Lemma D.5.

We can use low-rank approximation technique to represent $\widetilde{\gamma}_1(x) = U_3 V_3^\top$ and $\widetilde{\gamma}_2(x) = U_3 V_3^\top$ as the approximation to $\gamma_1(x)$ and $\gamma_2(x)$ respectively.

The cost is $n^{1+o(1)}$ time for every Lemma in Lemmas D.1, D.2, D.3, D.4 and D.5.

We have the reformulated gradient from Lemma C.4 as follows.

$$\frac{\mathrm{d}\mathsf{Loss}(x)}{\mathrm{d}x} = \underbrace{\widetilde{\mathsf{A}}^\top}_{d^4 \times n^2} \operatorname{vec}(\underbrace{\gamma(x)}_{n \times n})$$

Therefore, $n^{1+o(1)}$ is the total running time.

We show that

$$
\begin{aligned}
\|\frac{\mathrm{d}\mathsf{Loss}(X)}{\mathrm{d}x} - \widetilde{g}\|_\infty &= \|\underbrace{\widetilde{\mathsf{A}}^\top}_{d^4 \times n^2} \operatorname{vec}(\underbrace{\gamma(x)}_{n \times n}) - \underbrace{\widetilde{\mathsf{A}}^\top}_{d^4 \times n^2} \operatorname{vec}(\underbrace{\widetilde{\gamma}(x)}_{n \times n})\|_\infty \\
&= \|\underbrace{\widetilde{\mathsf{A}}^\top}_{d^4 \times n^2} (\operatorname{vec}(\underbrace{\gamma(x)}_{n \times n}) - \operatorname{vec}(\underbrace{\widetilde{\gamma}(x)}_{n \times n}))\|_\infty \\
&= \|\underbrace{\widetilde{\mathsf{A}}^\top}_{d^4 \times n^2}\|_\infty \|\operatorname{vec}(\underbrace{\gamma(x)}_{n \times n}) - \operatorname{vec}(\underbrace{\widetilde{\gamma}(x)}_{n \times n})\|_\infty \\
&= \|\underbrace{\widetilde{\mathsf{A}}^\top}_{d^4 \times n^2}\|_\infty \|\gamma(x) - \widetilde{\gamma}(x)\|_\infty \\
&\leq \epsilon / \operatorname{poly}(n).
\end{aligned}
$$

where the first equality is based on Lemma C.4, the second equality is due to the distributive law, the third equality derives from the definition of ℓ_∞ norm, the fourth equality is due to the def. of vectorization, and the fifth inequality derives from the Lemmas in Lemma D.4 and Lemma D.5.

We choose $\epsilon = \frac{1}{\mathrm{poly}(n)}$.

Thus, we have finished our proof. □

Remark D.7. *The assumption in Theorem 5.2 is practical. In practice, especially in recent long context tasks, the n is large, e.g., $n = 2 \times 10^6$ for Google's Gemini 1.5 Pro Gemini (2024), while the model training uses a half-precision floating-point format, e.g., the bit number is 16. Furthermore, our assumption is "tight", where if we slightly weaken the assumption, there is no algorithm that can solve the RoPE attention gradient computation in truly sub-quadratic complexity (Theorem 6.1).*

Our Theorem 5.2 accurately approximates ($\epsilon = 1/\mathrm{poly}(n)$) the RoPE attention gradient computation in almost linear time $n^{1+o(1)}$ under practical assumptions (see the above Remark D.7). Thus, our methods solve the last puzzle of RoPE attention acceleration. Combined with previous work on RoPE attention inference (see Lemma 5.1), this may make RoPE attention practical as we overcome the theoretical quadratic time complexity barrier both in inference and training.

E HARDNESS

In this section, we provide the lower bound results to compute the gradient of RoPE attention.

Theorem E.1 (Lower bound)**.** *Assuming* SETH*, for any $q > 0$, for the* ARAttLGC$(n, d = O(\log n), B = \omega(\sqrt{\log n})$*, there does not exist an algorithm which can be executed in time $O(n^{2-q})$ based on Def. 3.1.*

Proof. We pick all of the $W_{-(n-1)}, \dots, W_{n-1} \in \mathbb{R}^{d \times d}$ as an identity matrix I_d. Therefore, the gradient computation of RoPE attention can be treated as the gradient computation of classic attention. Thus, our lower bound result can derive from Alman & Song (2024c). □